UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE CIENCIAS MATEMÁTICAS
Departamento de Estadistica e Investigacion Operativa

TESIS DOCTORAL

Complejidad algorítmica : Cuestiones y aplicaciones Notables

MEMORIA PARA OPTAR AL GRADO DE DOCTOR
PRESENTADA POR

Ernesto Azorín Mínguez

Madrid, 2015

Ernesto Azorín Mínguez

COMPLEJIDAD ALGORITMICA: CUESTIONES Y APLICACIONES NOTABLES

Departamento de Estadística e Investigación Operativa
Facultad de Ciencias Matemáticas
Universidad Complutense de Madrid
1983

Colección Tesis Doctorales. Nº 195/83

Edita e imprime la Editorial de la Universidad Complutense de Madrid. Servicio de Reprografía
Noviciado, 3 Madrid-8
Madrid, 1983
Xerox 9200 XB 480
Depósito Legal: M-28028-1983

COMPLEJIDAD ALGORITMICA : CUESTIONES Y APLICACIONES NOTABLES .

Ernesto Azorín Mínguez

Tesis para optar al grado de Doctor en Ciencias Matemáticas, realizada bajo la dirección del Dr. Francisco José Cano Sevilla.

Universidad Complutense de Madrid
Facultad de Matemáticas
Sección de Estadística e Investigación Operativa.

Madrid, Junio de 1982.

I N D I C E

PREAMBULO

Permítaseme agradecer al Profesor Sixto Rios, director del Departamento de Estadística e Investigación Operativa de la Facultad de Matemáticas de la Universidad Complutense, el honor que me hace al presidir el tribunal que ha de juzgar esta tesis. A él le debo además el haberme acogido en el Departamento que dirige desde el inicio de mis actividades docentes en Madrid.

He de destacar mi reconocimiento al Profesor Francisco José Cano, Catedrático de Investigación Operativa y director de este trabajo, por sus numerosos y siempre oportunos consejos.

Asímismo, agradezco a los profesores Ildefonso Yáñez, Ramiro Melendreras y Ricardo Vélez por el interés que han demostrado por juzgar esta monografía.

He de recordar con gratitud a los profesores Rafael Portaencasa, Rector de la Universidad Politécnica de Madrid, y Antonio Insúa, Decano de la Facultad de Informática de dicha Universidad por la fe que pusieron en mi persona y por su apoyo continuo durante la realización de este trabajo.

Es mi deseo hacer una mención explícita de los profesores de los Departamentos de Estadística e Investigación Operativa de las Facultades de Matemáticas e Informática de Madrid, por su amistad y compañerismo.

No puedo finalizar sin recordar a mi familia. A mis padres, sin cuyo estímulo constante este trabajo tal vez no habría

llegado jamás a ver la luz. A mi esposa, Denise, con quien he compartido todas las preocupaciones cotidianas. A mis hijos, Mª José y Sebastián por su comprensión al ceder el tiempo que a ellos correspondía haber dedicado.

Es de justicia destacar la gran labor desarrollada por Mª Jesús Arranz, en la árida tarea de mecanografiar un manuscrito con profusión de fórmulas matemáticas.

Madrid, Junio de 1982

INTRODUCCION

El interés del autor por la Matemática de la Informática explica que los orígenes reales de este trabajo deban situarse en - la Teoría de Algoritmos y en la Lógica de Umbral. En este tipo de - lógica, una función de umbral es una función

$f : \{0,1\}^n \longrightarrow \{0,1\}$ definida por

$$f((x_1, x_2, \ldots, x_n)) = \begin{cases} 1 & \text{si} \quad \sum_{i=1}^{n} a_i x_i \geq s \qquad (1) \\ 0 & \text{si} \quad \sum_{i=1}^{n} a_i x_i < s \qquad (2) \end{cases}$$

para $a_1, a_2, \ldots, a_n, s$ números reales dados, y donde suele denominarse a s el umbral de f. En ciertas aplicaciones y en particular a lo largo de esta tesis estos parámetros son simplemente enteros - positivos. (Hu [1965] y Muroga [1971] son referencias estándar en Lógica de Umbral).

Por otro lado, a la luz de la Teoría de Algoritmos son posibles tres puntos de vista para caracterizar un problema como "problema resuelto" :

a) Si se ha probado un teorema de existencia de una solución del problema.
b) Si se conoce un método capaz de construir una solución.
c) Si dicho método converge en tiempo útil (por ejemplo en un tiempo que no supere a la duración estimada de - la vida humana.

Sí las dos primeras caracterizaciones han dividido a los

matemáticos respectivamente en no constructivistas y constructivistas, originándose a veces ásperas controversias entre ambas escuelas, razones hay para suponer que la tercera, mucho más reciente -- que las otras dos origine disensiones aún más profundas. Pero nuestro propósito no es polémico; únicamente deseamos hacer notar que - la última caracterización mencionada motiva los estudios recientes- sobre la complejidad (o de su inverso, la eficiencia), de los algoritmos.

El punto de partida del presente trabajo fue el intentar caracterizar la complejidad computacional de la determinación de - subconjuntos I de $\{1, 2, \ldots, n\}$ de cardinalidad minimal con respecto a (1) es decir, tales que $[(x_i = 1) \iff i \in I] \implies [f(x_1, x_2, \ldots, x_n) = 1]$ y $\forall J \subset I$, $[(x_i = 1) \iff i \in J] \implies [f(x_1, x_2, \ldots, x_n) = 0]$. A lo largo del estudio de esta -- cuestión fuimos encontrando múltiples problemas que comprenden relaciones semejantes a las de la definición (1), (2). Así por ejemplo, en Teoría de Grafos, Chvátal y Hammer (1973) introducen los llamados grafos de umbral, concepto desarrollado posteriormente por Benzaken y Hammer (1977); en Investigación Operativa, el principio de decisión de la mayoría, en problemas de decisión colectiva, permite también utilizar modelos del tipo (1), (2); y así en una larga lista que no cabría citar aquí, y cuya amplitud nos sugirió tratar el problema inicial independientemente del contexto de las aplicaciones.

Esta tesis consta de 4 capítulos desarrollados todos bajo la óptica de la Complejidad Computacional.

En el Capítulo 0, presentamos una breve introducción hist6

rica de este área, así como algunas definiciones y convenciones comunmente aceptadas. Nuestro objetivo no ha sido en este capítulo el de introducir una serie de conceptos mediante el formalismo matemático que les es propio, distinto del utilizado en los capítulos siguientes, sino más bien el de proveer al lector de una base mínima, pero suficiente para comprender los desarrollos subsiguientes. Las referencias que se citan aportan dicho formalismo al lector interesado y, en su mayor parte, constituyen fuentes de gran utilidad para el investigador.

Nuestro estudio personal comienza con el Capítulo I, en el cual desarrollamos tres algoritmos para determinar un conjunto I para el problema definido anteriormente en esta introducción que además tenga cardinalidad absoluta mínima.

El primero de estos algoritmos es intuitivo, el segundo necesita ordenar totalmente y en forma previa a $\{a_1, a_2, \ldots, a_n\}$ de mayor a menor. El análisis del tercero de los algoritmos propuestos revela que, para ciertos valores de n puede no ser necesario ordenar totalmente $\{a_1, a_2, \ldots, a_n\}$, dependiendo esta posibilidad del orden inicial de presentación de dichos elementos. Si bien no probamos que este algoritmo sea optimal, su complejidad es en todo caso muy aceptable.

El Capitulo II empieza con una generalización del problema anterior: la determinación de todos los conjuntos I minimales por inclusión. Estudiamos la complejidad intrinseca de este problema, y probamos que es NP-completo, por lo que resulta inverosímil que admita algoritmos eficientes. Tras adaptar un algoritmo existente para un problema similar de inecuaciones pseudo-booleanas, desa-

rrollamos una modificación que permite aprovechar con ventaja la posible existencia de elemntos iguales en $\{a_1, a_2, \ldots, a_n\}$.

Los resultados del capitulo II nos hicieron ver el interés de caracterizar la complejidad media del problema estudiado. La dificultad y el escaso desarrollo hasta la fecha de este tipo de estudio, nos llevaron en el Capítulo III a estudiar herramientas de la Estadística que puedan ser de utilidad para este fin. En particular, encontramos que, dada una población de tamaño n, el problema de determinar la proporción de muestra de tamaño k, con $k \leq n$, cuya media sea superior o igual a la media de una muestra dada sugiere un enfoque para resolver el problema del Capítulo II. En este último capítulo, no es menor el interés que presenta el intento de sistematizar el estudio de la complejidad de diversos problemas estadísticos clásicos.

En nuestra opinión, el trabajo que hemos realizado abre diversas líneas de investigación, entre las que destacaremos el estudio de la caracterización de la complejidad de los problemas introducidos si se permite el cálculo en paralelo, y el análisis de la complejidad de métodos numéricos en problemas estadísticos, especialmente en problemas de estimación.

También es de interés el investigar la posibilidad de conciliar con las ideas aquí desarrolladas otros puntos de vista, como por ejemplo los presentados por Fine (1973) sobre la complejidad del reconocimiento de secuencias aleatorias o por Valdés y Azorín (1980) sobre la complejidad de algunos problemas de Inferencia Secuencial.

Finalmente, creemos que la Lógica de Umbral sigue siendo fuente de otros problemas cuya complejidad es interesante estudiar.

C A P I T U L O 0

PRELIMINARES .

0.1 Introducción Historica

El análisis de algoritmos es un área de investigación que adquiere importancia durante las últimas dos décadas, como una de las consecuencias de la concepción y desarrollo de los ordenadores digitales. Como precursor puede citarse a C.L. Dodgson (1883) más conocido como Lewis Carroll, por su estudio sobre ordenación de n elementos, aplicado a torneos de tenis.

Tras el intento de formalización de la noción intuitiva de algoritmo por Turing (1936) (mediante el concepto de función efectivamente calculable), Kleene (1936) (mediante el concepto de función recursiva), y tras la famosa tesis de Church (1936) que afirma que toda función intuitivamente calculable es efectivamente calculable, trabajos posteriores demostraron la equivalencia entre las nociones de calculabilidad efectiva y recursividad, así como de otras diversas formalizaciones amplias del concepto de algoritmo.

Surgió entonces en forma natural el problema de encontrar medidas del tiempo necesario para calcular una función dada (tiempo necesario para que un algoritmo dado resuelva un problema dado) y del espacio de cálculo que se necesita. Hopcroft y Ullman (1979), Blum (1967), Borodin (1972), entre otros, dieron grandes pasos en este sentido estudiando un concepto abstracto de complejidad. Paralelamente empiezan a estudiarse algoritmos prácticos para problemas concretos. Con frecuencia, diversos algoritmos distintos permiten resolver un problema o clase de problemas y sólo tras el análisis cuidadoso del algoritmo, del problema y de la utilización del problema se podrá elegir el "mejor" algoritmo. El objeto del análisis de algoritmos consiste entonces en conocer los méritos relativos de

los algoritmos para poder elegir. Estos análisis pueden conducir - también al desarrollo de nuevos algoritmos aún más eficientes. En esta línea, uno de los descubrimientos más importantes ha sido el de la clase de problemas NP-completos, para los que no solo no se conocen algoritmos eficientes sino que cualquier algoritmo eficiente para algún problema de la clase permitirá construir automáticamente algoritmos eficientes para el resto de los problemas de la clase.

0.2 Noción de algoritmo

Para todos los efectos nos bastará la siguiente definición informal de algoritmo.

Definición (Kronsjö [1979]):

Un algoritmo es un procedimiento que consiste en un conjunto finito de reglas no ambiguas que especifican una sucesión finita de operaciones cuyo resultado es la solución de un problema.

Esta definición lleva implícita los siguientes aspectos esenciales :

a) Determinismo.-

Las reglas deben indicar sin ambigüedad lo que hay que - hacer en cada paso.

b) Finitud.-

El algoritmo debe conducir a la solución del problema en un número finito de pasos.

c) Entrada y salida.-

La entrada es el conjunto de datos que definen el problema a resolver y la salida su solución, ambas codificadas en la forma que resulte conveniente.

0.3 Eficiencia y complejidad

La noción de algoritmo eficiente comunmente aceptada corresponde al criterio propuesto originalmente por Edmonds (1965) - que considera como tales a los algoritmos que trabajan en tiempo - limitado superiormente por un polinomio en el tamaño del problema (es decir, en el número de datos que definen al problema). En contraposición, los algoritmos "malos" tienen necesidad de un tiempo exponencial al menos para un caso del problema. Por abuso del lenguaje se habla respectivamente de algoritmos de complejidad polinomial y no polinomial, o simplemente de algoritmos polinomiales y - no polinomiales. La noción de complejidad corresponde por tanto al recíproco de la eficiencia. La idea esencial que justifica esta convención es que permite diferenciar claramente el tamaño máximo de - los problemas que pueden resolverse en tiempo útil, lo que es de - gran interés dado que en la mayor parte de los casos de interés -- práctico se trabaja con problemas de tamaño relativamente grande.

Esta convención tiene sin embargo un inconveniente teórico y otro práctico. El primero se refiere a la dudosa conveniencia de calificar de eficiente a un algoritmo cuya complejidad sea del - orden de n^{1000}, por ejemplo. Felizmente en, los problemas de interés práctico que pueden resolverse mediante algoritmos polinomiales, el orden del polinomio suele ser inferior a 5. El inconveniente práctico

co es que un algoritmo no polinomial puede tener un comportamiento eficiente en todos menos un conjunto reducido de casos del problema que intenta resolver. Probablemente el ejemplo más conocido al respecto es el del algoritmo Simplex de Programación Lineal. Este algoritmo realiza en la práctica del orden de 1,5m cambios de base, donde m es el número de restricciones del problema. Sin embargo -- Klee y Minty (1972) desarrollaron un ejemplo "de laboratorio" para el cual el algoritmo Simplex ya no es eficiente. Por otro lado, -- Khachiyan (1979) obtiene un algoritmo polinomial pero no lineal para este problema, por lo que no se compara favorablemente con el - Simplex (Lovász [1980]). En vista de esta dificultad, conviene utilizar con cautela el criterio de Edmonds, e intentar, cuando sea - posible, determinar la complejidad media del algoritmo; la dificultad teórica de este último enfoque estriba en que no siempre es fácil conocer la distribución de los distintos casos de un problema dado.

En forma análoga se puede estudiar también una noción de la complejidad según el espacio (papel, memoria de ordenador, etc.) utilizado por un algoritmo. Obras básicas que desarrollan estos -- conceptos son las de Aho et al. (1974), Savage (1976) y Kronsjö -- (1979).

0.4 Problemas NP-completos

Definición 0.4.1.-

Informalmente, un problema es NP-completo si

a) está en NP

b) su pertenencia a la clase P implicaría que P = NP

donde P = clase de problemas que pueden ser resueltos mediante algoritmos polinomiales y NP = clase de problemas que pueden ser resuelton mediante algoritmos no deterministas polinomiales [1]. Para una formalización de estos conceptos ver Cook (1971) y Karp (1972). Se considera en la práctica como problemas intratables computacionalmente a los problemas NP-completos.

La importancia de esta clase de problemas deriva

a) del gran número de problemas clásicos que contiene (una lista bastante completa de estos problemas es la dada por Garey y Johnson [1979]),
b) de que no ha sido posible hasta la fecha encontrar algoritmos polinomiales para ninguno de ellos,
c) de que tampoco ha sido posible probar la inexistencia de tales algoritmos,
d) de que todos estos problemas son reducibles entre si por medio de una transformación de complejidad polinomial, por lo que el desarrollo de un algoritmo polinomial para cualquiera de ellos implicaría automáticamente que todos estos problemas están en P.

La cuestión $P \stackrel{?}{=} NP$ está abierta. La conjetura es que $P \neq NP$, pero se estima que se necesitan técnicas matemáticas nuevas para probarla.

0.5 Nuevas vías de investigación

Si a la noción de algoritmo introducida en 0.2 le agregamos la posibilidad de decidir aleatoriamente entre varias acciones posibles en cada paso, se obtiene el concepto nuevo de algorit

mo probabilístico (ver por ejemplo Rabin [1976]), que ha resultado útil para algunos problemas (Rabin [1980]) y que constituye un área de investigación activa. El problema teórico más interesante que se suscita es el de averiguar si estos algoritmos son, en algún sentido, más potentes que los clásicos. Con este fin, Gill (1977) y Adelman y Manders (1977) proponen diversas definiciones plausibles para algoritmos probabilísticos polinomiales, lo que permite a su vez definir la clase de problemas RP . Intuitivamente, un problema está en RP si puede ser resuelto mediante algún algoritmo probabilístico de complejidad polinomial. Si bien a priori esta clase podría parecer más amplia que la clase P, el averiguar si RP = P es una cuestión abierta. Más aún (Rackoff [1982]) algunos resultados de la Teoría de la NP-completitud hacen pensar que, tal como sucede con el problema $P \stackrel{?}{=} NP$, serán necesarias herramientas matemáticas nuevas para obtener una respuesta concreta.

NOTAS A PIE DE PAGINA DEL CAPITULO 0

(1) Un algoritmo no determinista puede considerarse como aquél que en cada paso puede elegir entre un conjunto de acciones posi-- bles.

C A P I T U L O I

DETERMINACION DE UN SUBCONJUNTO DE NUMEROS ENTEROS CON CIERTAS PROPIEDADES.

I.1 Introducción

Sea A una sucesión finita de enteros positivos, (no necesariamente distintos)

$$s \in \mathbb{N}$$
$$S = \left\{ B : B \subseteq A \quad y \quad \sum_{x \in B} x \geqslant s \right\} \tag{1.1}$$

Consideramos el problema de determinar una subsucesión I de S minimal y de cardinalidad mínima.

Ejemplo:

$$A = \{ 2, 3, 5, 1, 8, 3 \} \; ; \quad s = 11$$

Las soluciones de este problema son los conjuntos

$$\{3,8\} \quad y \quad \{5,8\}$$

El número de soluciones del problema estará comprendido entre 0 (si $\sum_{x \in A} x < s$) y $|A|$ (si para todo x en A : $x \geqslant s$).

En lo que sigue, nos limitaremos al caso en que exista (al menos) una solución. (Basta comparar $\sum_{x \in A} x$ con s para resolver el problema de existencia).

Examinaremos tres algoritmos para resolver este problema, y analizaremos sus complejidades. Parece natural que la medida de complejidad escogida sea el número total de comparaciones y sumas bina

rias realizadas:

$c_{AS}(n)$ = número de comparaciones binarias +
+ número de sumas binarias realizadas por el algoritmo donde $n = |A|$.

I.2 Un algoritmo intuitivo

El algoritmo más intuitivo que se pueda imaginar consiste en recorrer S en el sentido determinado por la relación de inclusión.

Algoritmo AS_1 :

En la etapa i-ésima: se consideran una a una las sucesiones $B \subseteq A$ de cardinalidad i; si $\sum_{x \in B} x \geq s$ entonces I = B y FIN.
si para todo B, $\sum_{x \in B} x < s$, continuar a la etapa i+1.

Es evidente que I, encontrada de esta manera, es una solución del problema.

Estudiemos ahora la complejidad del algoritmo AS_1

Proposición I.2.1

La complejidad del algoritmo AS_1 es de $O(2^n)$

Prueba:

El número de comparaciones binarias está acotado por

$$\sum_{j=1}^{n} \binom{n}{j} = 2^n - 1 \qquad (2.1)$$

En cuanto al número de sumas binarias, éste será a lo sumo

$$\sum_{j=2}^{n} \binom{n}{j} = 2^n - (n+1) \qquad (2.2)$$

En consecuencia $c_{AS_1}(n) \leq 2^{n-1} - (n+2)$ (2.3)

c.q.d.

I.3 Utilización de una relación de orden

Es posible obtener un resultado sensiblemente mejor si se ordenan previamente en forma decreciente los elementos de A.

Algoritmo AS_2:

Preparación: Ordenar totalmente A mediante la relación "$\geqslant$".

(Si k elementos $x_{i_1}, x_{i_2}, \ldots, x_{i_k} \in A$ son iguales, se ordenarán arbitrariamente).

Sea $A_{\geqslant}$ el resultado obtenido.

En la etapa i-ésima: Si $\sum_{j=1}^{i} x_j \geqslant s$ entonces $I = \{x_j\}_{j=1}^{i}$ y FIN

En caso contrario, continuar a la etapa siguiente.

Proposición I.3.1

La complejidad del algoritmo AS_2 está acotada superiormente - por $O(n \log_2 n)$.

Prueba:

El método más eficiente conocido para ordenar totalmente una - sucesión de n elementos es el de "fusión-inserción" (merge-insertion) debido a Ford y Johnson [1959], que precisa, a lo sumo, de (1)

$\sum_{i=1}^{n} \lceil \log_2 \frac{3i}{4} \rceil$ comparaciones binarias (Hadian [1969]). Por lo tanto no son necesarias más de $\sum_{i=1}^{n} \lceil \log_2 \frac{3i}{4} \rceil + n$ comparaciones binarias para determinar I. Por otro lado, el número máximo de sumas binarias a realizar es solo de n-1.

En consecuencia:

$$c_{AS_2}(n) = O\left(\sum_{i=1}^{n} \lceil \log_2 \frac{3i}{4} \rceil + 2n - 1\right)$$

$$= O\left(\sum_{i=1}^{n} \log_2 \frac{3i}{4} + 2n\right)$$

$$= O\left(\sum_{i=1}^{n} \log_2 i + n\left(\log_2 \frac{3}{4} + 2\right)\right)$$

$$= O(\log_2 n! + \alpha n), \text{ con } \alpha = \log_2 \frac{3}{4} + 2$$

Mediante la aproximación de Stirling, $n! \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n}$, resulta:

$$c_{AS_2}(n) = O\left(n \log_2 \frac{n}{e} + \frac{1}{2} \log_2 (2\pi n) + \alpha n\right)$$

$$= O(n \log_2 n) \qquad (3.1)$$

c.q.d.

I.4 Desarrollo de un algoritmo computacionalmente eficiente

¿Puede optimizarse el algoritmo anterior si A se ordena solo parcialmente?. Intentaremos dar respuesta a esta cuestión en esta sección y en la siguiente.

Con este fin, consideremos un tercer algoritmo, definido por:

Algoritmo AS_3 :

Etapa i-ésima: determinan el i-ésimo elemento, x_i, de A (según la relación $\geqslant$).

Si $\sum_{j=1}^{i} x_j \geqslant s$ entonces $I = \{x_j\}_{j=1}^{i}$ y FIN.

En caso contrario, continuar a la etapa siguiente.

Podría pensarse en utilizar un algoritmo de selección en cada etapa. Ahora bien, si observamos que el i-ésimo elemento de A (según $\geqslant$) solo es útil si los i-1 elementos que le preceden resultan insuficientes, será más interesante recurrir a un algoritmo que respete la estructura de orden parcial obtenida. Nos proponemos aquí utilizar las ideas de "fusión-inserción" para ordenar parcialmente A, y analizar en detalle la complejidad de AS_3 .

Definición I.4.1

Llamaremos $Y_i(n)$ al número de comparaciones binarias necesarias para determinar el i-ésimo elemento (según $\geqslant$) de una sucesión de n, tras haber determinado los i-1 elementos precedentes, mediante el algoritmo de fusión-inserción.

Como en cada etapa de AS_3 se realiza una comparación suplementaria (con s) y puesto que a partir de la segunda etapa se realiza una suma de dos elementos, la complejidad de AS_3 estará acotada por:

$$c_{AS_3}(n) \leqslant \sum_{i=1}^{n} Y_i(n) + 2n - 1$$

Para facilitar la comprensión del cálculo de $Y_i(n)$, explicaremos aquí, brevemente, el algoritmo de fusión-inserción.

Algoritmo de fusión-inserción (Ford y Johnson [1959]).

Sea A la sucesión $x_1, x_2, \ldots, x_n$.

1) Comparar dos a dos $\left\lfloor \frac{n}{2} \right\rfloor$ elementos de A, dejando de lado un elemento si n es impar. (Fig. 1).

2) Ordenar de mayor a menor los $\left\lfloor \frac{n}{2} \right\rfloor$ elementos mayores, mediante este mismo algoritmo.
Se obtiene así una subsucesión A' de A, totalmente ordenada, — que llamaremos cadena principal. (Fig. 2).

3) Insertar los $\left\lceil \frac{n}{2} \right\rceil$ elementos restantes en la cadena principal.

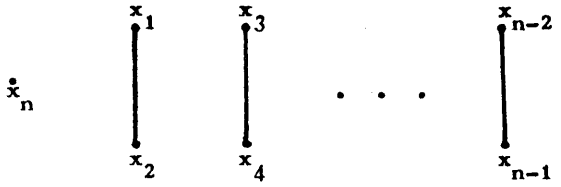


Fig. 1 Por convención, los elementos que figuren en el extremo superior de un segmento serán los elementos mayores obtenidos en el paso 1]. Es decir, $x_1 \geqslant x_2$, $x_3 \geqslant x_4$, ... , $x_{n-2} \geqslant x_{n-1}$.
Este esquema corresponde al caso n impar. Si n es par, ningún elemento queda aislado.

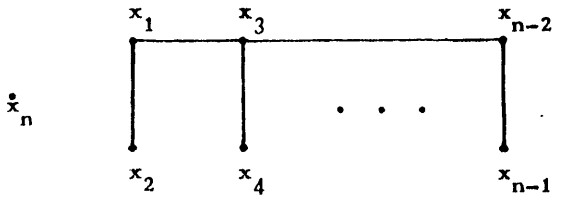


Fig. 2 En este esquema, el segmento horizontal une los $\lfloor \frac{n}{2} \rfloor$ elementos ordenados decrecientemente en el paso 2 , y numerados de modo que $x_1 \geqslant x_3 \geqslant \cdots \geqslant x_{n-2} \geqslant x_{n-1}$. Estos elementos forman la cadena principal.

En lo que concierne al paso 3] del algoritmo de fusión-inserción se observa que:

Teorema I.4.1.- (Ford y Johnson)

La inserción de un elemento en la cadena principal se realiza de la manera más eficiente posible cuando la cadena tiene $2^k - 1$ elementos $(k > 1)$.

Prueba:

Consideremos una cadena de n elementos, y sea x un elemento que se desea insertar en ella. Para ello bastan $\lfloor \log_2 n \rfloor + 1$ comparaciones (en efecto, tras comparar x con el elemento $\lfloor \frac{n}{2} \rfloor$ -ésimo de la cadena, x debe insertarse en una cadena que tenga como máximo $\lfloor \frac{n}{2} \rfloor$ elementos).

Ahora bien, si establecemos la partición de $\mathbb{Z}^+$ en clases - en las que la función $f(n) = \lfloor \log_2 n \rfloor$ es constante:

$\{2^{k-1}, 2^{k-1} + 1, \ldots, 2^k - 1\}$; $k > 1$,

resulta que la cadena más larga en la que puede insertarse un elemento con k comparaciones como máximo, es una cadena de $2^k - 1$ elementos.

c.q.d.

Observación:

Esta idea, utilizada iterativamente, da lugar al método de inserción dicotómica para ordenar una sucesión según " $\geq$ " (ver - por ejemplo Louit [1971]), cuya complejidad es igual a $\sum_{i=1}^{n} \lceil \log_2 i \rceil$.

El resultado anterior nos permite directamente determinar - la manera óptima de insertar elementos en la cadena principal:

Corolario I.4.1.-

El orden óptimo de inserción de elementos en la cadena principal es (ver Fig. 3): $x'_3, x'_2, x'_5, x'_4, x'_{11}, x'_{10}, \ldots$

Si llamamos $\bar{k}$ a la clase de elementos que precisan exactamente de k comparaciones para insertarse en la cadena principal -- tendremos:

$$\bar{1} = \{x'_1\}$$
$$\bar{2} = \{x'_3, x'_2\}$$
$$\bar{3} = \{x'_5, x'_4\}$$
$$\bar{4} = \{x'_{11}, x'_{10}, \ldots, x'_6\}$$
$$\vdots$$
$$\bar{k} = \{x'_{t_k}, x'_{t_k-1}, \ldots, x'_{t_{k-1}+1}\}$$

donde $t_k = \dfrac{2^{k+1} + (-1)^k}{3}$, $\quad k \geq 0$ (4.1)

El desarrollo para obtener este relación puede verse, por ejemplo, en Knuth [1973] . La definición de t_0 se ha establecido por convención.

La notación introducida permite enunciar el siguiente

Corolario I.4.2.-

Para $k > 1$, la clase $\bar{k}$ se inserta en la cadena principal después de haberlo hecho la clase $\bar{k-1}$.

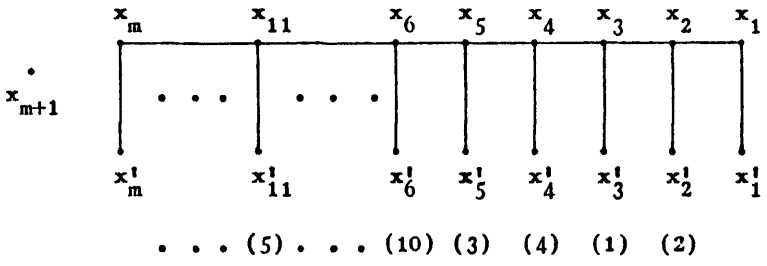


Fig. 3 Los enteros bajo los x'_j indican el orden de inserción de los x'_j en la cadena principal $x_m, \ldots, x_2, x_1, x'_1$

Esto completa la descripción del algoritmo. En lo que respecta a su complejidad U(n), se tiene, por recurrencia:

$$U(1) = 0$$

$$U(n) = \left\lfloor \frac{n}{2} \right\rfloor + U\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + G\left(\left\lceil \frac{n}{2} \right\rceil\right) \qquad (4.2)$$

donde cada término representa la contribución de los pasos 1], 2], y 3] respectivamente, y

$$G(m) = \sum_{i=1}^{k-1} i\,(t_i - t_{i-1}) + k\,(m - t_{k-1}) \qquad (4.3)$$

$$\text{para } t_{k-1} \leq m \leq t_k$$

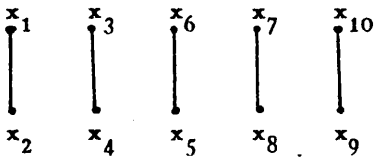
Ilustraremos lo anterior mediante el siguiente

Ejemplo:

Sea n=10 y la sucesión de enteros positivos $\{x_i\}_{i=1}^{10}$.

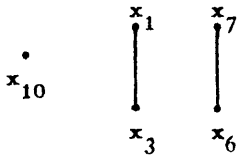
Se forman los pares (x_1, x_2); (x_3, x_4); ... Supongamos que tras ha-

ber comparado los dos elementos de cada par entre ellos (paso 1])
se obtiene el resultado:



Se forman nuevos pares con los elementos mayores, por ejemplo (x_1, x_3); (x_6, x_7) y se deja x_{10} solo.

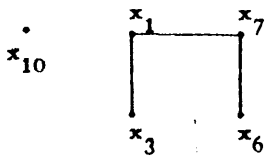
Sean



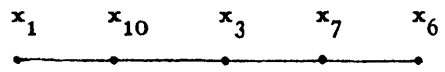
y

las configuraciones obtenidas tras dos nuevas aplicaciones del paso 1] .

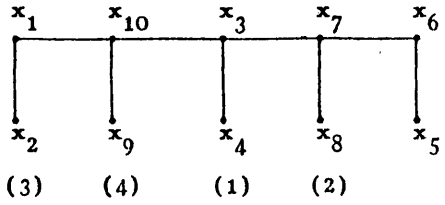
Se tiene, pues, la estructura



Estamos ahora en el paso 3] . Introduciendo x_{10} y x_3 - en la cadena principal, supongamos que se obtiene:



Colocando los elementos restantes, que habíamos omitido por claridad, en sus posiciones respectivas:



Finalmente, tras la inserción de x_4, x_8, x_2 y x_9 (en este - orden) en la cadena principal, los diez elementos quedan ordenados - decrecientemente.

El número de comparaciones binarias realizadas será, a lo - sumo, $U(10) = 22$.

Nos enfrentaremos ahora al cálculo de las funciones $Y_i(n)$ - de la definición I.4.1

Teorema I.4.2.-

$$Y_1(n) = \begin{cases} \lfloor \frac{n}{2} \rfloor + Y_1(\lfloor \frac{n}{2} \rfloor) & \text{para n par} \\ \lfloor \frac{n}{2} \rfloor + U(\lfloor \frac{n}{2} \rfloor) + H(\lceil \frac{n}{2} \rceil) & \text{para n impar} \end{cases} \qquad (4.4)$$

$$\text{donde } H(m) = \sum_{i=1}^{k-1} i\,(t_i - t_{i-1}) + k \qquad (4.5)$$

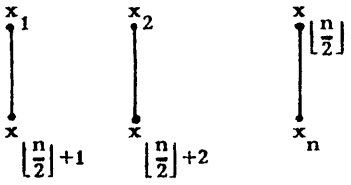
$$\text{para } 1 \leq t_{k-1} < m \leq t_k$$

$$H(1) = 0$$

Prueba:

a) n par.

Finalizado el paso 1] del algoritmo de fusión-inserción, se habrán efectuado $\lfloor \frac{n}{2} \rfloor$ comparaciones binarias y se habrá obtenido la estructura:

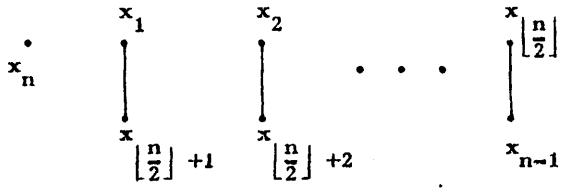


El mayor elemento pertenecerá forzosamente al conjunto $\{x_1, x_2 \ldots, x_{\lfloor \frac{n}{2} \rfloor}\}$. Como hasta aquí no se tiene información alguna sobre la posición relativa de los elementos de este conjunto, el coste de la obtención del mayor de ellos será $Y_1(\lfloor \frac{n}{2} \rfloor)$, y por lo tanto $Y_1(n) = \lfloor \frac{n}{2} \rfloor + Y_1(\lfloor \frac{n}{2} \rfloor)$.

b) n impar.

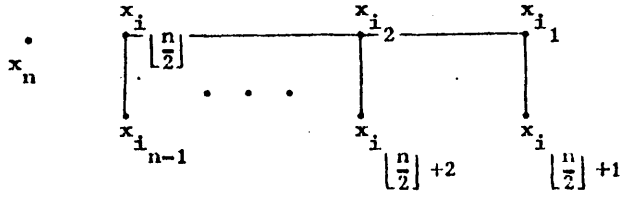
Al igual que en el caso anterior, el paso 1] del algoritmo de fusión-inserción implica $\lfloor \frac{n}{2} \rfloor$ comparaciones, pero un elemento no habrá sido comparado con ningún otro; sea x_n este elemento.

Se tiene entonces:



El mayor elemento se encuentra en el conjunto $\{x_n, x_1, x_2, \ldots, x_{\lfloor \frac{n}{2} \rfloor}\}$

En el paso 2] del algoritmo se ordena el conjunto $\{x_1, x_2, \ldots, x_{\lfloor \frac{n}{2} \rfloor}\}$, obteniéndose, tras U $(\lfloor \frac{n}{2} \rfloor)$ comparaciones la estructura:



El conjunto de elementos entre los que se encuentra el mayor se ha reducido así a $\{x_n, x_{i_{\lfloor \frac{n}{2} \rfloor}}\}$. Ahora, si x_n pertenece a la clase $\bar{k}$, su inserción en la cadena principal solo tendrá lugar tras la inserción de las k-1 clases precedentes (Corolario I.4.2.), para - lo cual son necesarias $\sum_{i=1}^{k-1} i\,(t_i - t_{i-1}) + k$ comparaciones adicionales. De esta manera, en total se habrán realizado

$$\lfloor \tfrac{n}{2} \rfloor + U(\lfloor \tfrac{n}{2} \rfloor) + \sum_{i=1}^{k-1} i\,(t_i - t_{i-1}) + k \quad \text{comparaciones binarias.}$$

El lector habrá observado que la expresión anterior no está

definida para n = 1 puesto que su clase $\bar{k}$ tampoco lo está. Como dicho caso es trivial podríamos conformarnos con una fórmula de $Y_1(n)$ para $n > 1$. Pero la dificultad es solo aparente: podemos considerar que en este caso el elemento solitario pertenece ya a la cadena principal (y constituye su único elemento). Todo esto se resume difiniendo

$$H(m) = \sum_{i=1}^{k-1} i\,(t_i - t_{i-1}) + k \qquad \text{si} \quad t_{k-1} < m \leq t_k$$

y $\quad H(1) = 0$

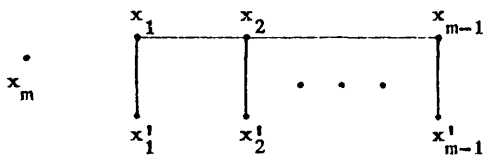
con lo cual:

$$Y_1(n) = \left\lfloor \frac{n}{2} \right\rfloor + U\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + H\left(\left\lceil \frac{n}{2} \right\rceil\right)$$

c.q.d.

Observaciones:

1.- Intuitivamente, H(m) representa el número de comparaciones necesarias para encontrar el mayor elemento en sucesiones parcialmente ordenadas en la forma:



con $1 \leq t_{k-1} < m \leq t_k$

2.- Se verifica trivialmente que

$$H(m) = G(m) - k\,(m - t_{k-1} - 1)\ ; \quad t_{k-1} < m \leq t_k \qquad (4.6)$$

$$H(t_k + 1) = G(t_k + 1)\ ; \quad k \geq 0 \qquad (4.7)$$

Desarrollaremos ahora una expresión para $Y_1(n)$ que no separe los casos n par y n impar.

Con este fin, observamos que en el proceso de obtención del mayor elemento de A hay una primera parte en la que se establece una partición de A en parejas de elementos -salvo un elemento aislado, eventualmente-, se comparan entre sí los elementos de cada pareja y se repite todo con los elementos mayores, continuando así hasta que sea imposible la formación de nuevas parejas. Esquemáticamente, podemos visualizar esto por medio de una "torre invertida", que llamaremos "torre de comparaciones" (Fig. 4) en la que cada vértice representa un elemento, las aristas representan una relación de orden sobre parejas de elementos (siendo el elemento superior el mayor) y las filas corresponden a etapas sucesivas del proceso. Es claro que tras la última etapa, el número de comparaciones realizadas será de:

$$\sum_{i=1}^{L(n)} \left\lfloor \frac{n}{2^i} \right\rfloor \qquad (4.8)$$

. con $L(n) = \lfloor \log_2 n \rfloor$

Llamaremos "niveles" a las filas de la torre de comparaciones y los numeraremos de arriba a abajo: 0, 1, 2, ...

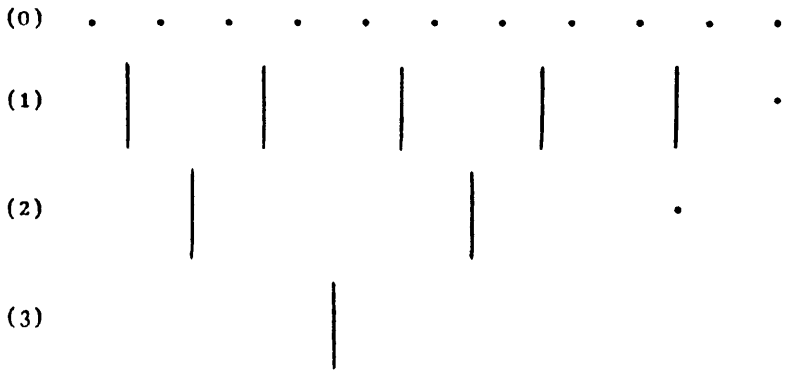


Fig. 4 Torre de comparaciones. La numeración de los niveles, indicada a la izquierda entre paréntesis corresponde a las etapas. Algunos niveles tienen un punto aislado, que corresponde a un elemento del nivel superior que no ha podido ser comparado con otro.

Para ser siempre fiel al algoritmo de fusión-inserción es preciso ahora recorrer en sentido inverso los niveles de la torre de comparaciones y terminar de ordenar totalmente los elementos de cada uno de ellos, por medio del paso 3 del algoritmo. El proceso deberá detenerse en cuanto se determine el primer elemento.

Sea i_0 el primer nivel que contenga un elemento aislado, si dicho nivel existe; en caso contrario, sea $i_0 = L(n) + 1$. En razón de la estructura de orden que hemos formado, el mayor elemento de este nivel lo será también de A. El número de comparaciones necesarias para obtener este elemento es, pues, calculable como sigue:

- comparaciones realizadas en los niveles $i_0 + 1$ y siguientes:

$$\sum_{i=i_0+1}^{L(n)} G\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i-1}} \right\rfloor}{2} \right\rceil\right) \qquad (4.9)$$

- obtención del primer elemento del nivel i_0 :

$$H\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i_0-1}} \right\rfloor}{2} \right\rceil\right) \qquad (4.10)$$

Sumando ahora las expresiones (4.8), (4.9) y (4.10) se obtiene el

Teorema I.4.3.-

$$Y_1(n) = \sum_{i=1}^{L(n)} \left\lfloor \frac{n}{2^i} \right\rfloor + \sum_{i=i_0+1}^{L(n)} G\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i-1}} \right\rfloor}{2} \right\rceil\right) +$$

$$+ H\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i_0-1}} \right\rfloor}{2} \right\rceil\right), \qquad n > 0 \qquad (4.11)$$

Puede observarse que si $n = 2^p$, $p > 0$ entonces las fórmulas - (4.8) y (4.9) son iguales a cero, y se tiene simplemente

$$Y_1(n) = \sum_{i=1}^{p} \left\lfloor 2^{p-i} \right\rfloor$$

$$= \sum_{i=0}^{p-1} 2^i \quad , \qquad n = 2^p, \quad p > 0 \qquad (4.12)$$

Solo queda calcular i_0 en función de n. Esto resulta muy sencillo si se utiliza la representación binaria de n : bin (n). Recorriendo bin (n) de derecha a izquierda, la posición del primer 1

encontrado corresponde precisamente a i_0.

Para simplificar la notación, introduciremos la

Definición I.4.2.-

Sea $n > 0$, $j > 0$, $L(n) + 1 \geq p > 0$

a) $$g(n, j, p) = \sum_{i=j}^{p} G\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i-1}} \right\rfloor}{2} \right\rceil\right) \qquad (4.13)$$

b) $$h(n, j) = H\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{j-1}} \right\rfloor}{2} \right\rceil\right) , \quad j \leq L(n) + 1 \qquad (4.14)$$

con lo que la expresión (4.11) se convierte en

$$Y_1(n) = \sum_{i=1}^{L(n)} \left\lfloor \frac{n}{2^i} \right\rfloor + g(n, i_0 + 1, L(n)) + h(n, i_0), \qquad (4.15)$$

para $n > 0$

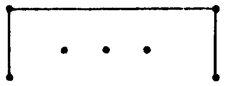
Antes de intentar desarrollar una fórmula para $Y_2(n)$, es conveniente examinar con más detalle el "recorrido" seguido a lo largo de los niveles de la torre de comparaciones, en la determinación de $Y_1(n)$.

Proposición I.4.1

El algoritmo de fusión-inserción obtiene el segundo mayor elemento de A en el nivel 1 de la torre de comparaciones.

Prueba:

La obtención del segundo mayor elemento de A mediante el algoritmo de fusión-inserción precisa la ordenación previa de A en la forma



Si n es par, este resultado se obtendrá después de haber realizado las comparaciones necesarias para ordenar totalmente los niveles - $i = i_0,\ i_0 - 1,\ \ldots,\ 2$. (Si n es potencia exacta de 2, se empezará, evidentemente, en el nivel $i_0 - 1$).

Si n es impar, el caso es aún más sencillo puesto que el mayor elemento de A se obtiene también en el nivel 1.

c.q.d.

En lo que sigue, las siguientes definiciones serán útiles.

Definición I.4.3.-

Llamaremos n_0 al menor entero impar de la sucesión

$$\left\{\left\lfloor \frac{n}{2^i} \right\rfloor\right\}_{i \geq 0} .$$

De la definición de i_0, es evidente entonces que

$$n_0 = \frac{n}{2^{i_0-1}} \qquad (4.16)$$

Definición I.4.4.-

Sea $k(m)$ el menor entero k tal que $m \leq t_k$.

Proposición I.4.2

Para todo k positivo: $t_k < 2^k < t_{k+1}$

Prueba:

Demostraremos que para todo $k \geq 0$: $2^k = \frac{t_k + t_{k+1}}{2}$

En efecto,

$$\frac{t_k + t_{k+1}}{2} = \frac{2^{k+1} + (-1)^k + 2^{k+2} + (-1)^{k+1}}{6}$$

$$= \frac{2^{k+1} \cdot 3}{6}$$

$$= 2^k$$

Pero, para $k > 0$ se tiene $t_k \neq t_{k+1}$

En consecuencia $2^k \in (t_k, t_{k+1})$.

c.q.d.

Proposición I.4.3

a) $k(1) = 0$

b) Para $m > 1$:

$$k(m) = \begin{cases} L(m) + 1 & \text{si} \quad m \leq t_{L(m)+1} \\ L(m) + 2 & \text{en caso contrario} \end{cases} \qquad (4.17)$$

Prueba:

Para todo $m > 1$ existe $k > 1$ tal que $2^{k-1} \leq m < 2^k$.

Se tiene entonces la siguiente dicotomía:

o bien $2^{k-1} \leq m \leq t_k$

o bien $t_k < m < 2^k$

Basta por lo tanto comparar m con $t_{\lfloor \log_2 m \rfloor + 1}$ para conocer $k(m)$.

Por otra parte, $t_0 = t_1 = 1$. De donde resulta $k(1)=0$.

c.q.d.

Definición I.4.5.-

$$sg(m) = \begin{cases} 0 & \text{si} \quad m = 0 \\ 1 & \text{si} \quad m > 0 \end{cases}$$

Definición I.4.6.-

a) $\tilde{H}(m) = G(m) - H(m) , \qquad m > 0 \qquad (4.18)$

b) $\tilde{h}(n, j) = \tilde{H}\left(\left|\frac{\left\lceil \frac{n}{2^{j-1}} \right\rceil}{2}\right|\right), \quad j \leq L(n)+1 , \qquad n > 0 \qquad (4.19)$

Observación :

Aunque desprovisto de sentido real, será útil prolongar la validez de la definición de t_k (ver (4.1)), hasta $k = -1$. Resulta así, $t_{-1} = 0$. Aceptando además por convención que $k(0) = -1$, puede enunciarse el

Teorema I.4.4.-

Sea $n > 0$

$$Y_2(n) = \begin{cases} \tilde{h}(n, i_0) + g(n, 2, i_0-1) + h(n, 1) & \text{para } n \text{ par} \\ k(\lfloor \frac{n}{2} \rfloor) \cdot sg(t_{k(\lfloor \frac{n}{2} \rfloor)} - \lfloor \frac{n}{2} \rfloor) \text{ ó cero} & \text{para } n \text{ impar} \end{cases} \qquad (4.20)$$

Prueba:

a) n par

Tras haber encontrado el mayor elemento de A, el número de comparaciones que faltan para terminar de ordenar completamente el nivel i_0 es

$$G(\lceil \frac{n_0}{2} \rceil) - H(\lceil \frac{n_0}{2} \rceil) = \tilde{H}(\lceil \frac{n_0}{2} \rceil)$$

$$= \tilde{h}(n, i_0) \qquad (4.21)$$

Asímismo, la contribución debida al ordenamiento de los niveles i, $1 < i < i_0$ es

$$g(n, 2, i_0 - 1) \qquad (4.22)$$

Ya no falta más que la obtención del segundo elemento del nivel 1. Obsérvese la figura 5.

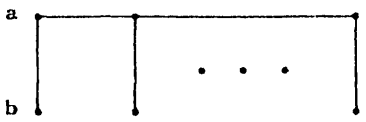


Fig. 5 Nivel 1 de la torre de comparaciones después de haber ordenado completamente los niveles siguientes. (Siendo n par).

Para conocer el segundo elemento es preciso insertar b en la cadena principal, lo que requiere $H\left(\frac{n}{2}\right) = h\,(n\,,\,1)$ comparaciones. Sumando este término a (4.14) y (4.15) se obtiene el resultado buscado.

b) n impar

En este caso, $Y_2(n)$ equivaldrá simplemente al número de comparaciones necesarias para la inserción del elemento $x_{\left\lfloor\frac{n}{2}\right\rfloor}$ en la cadena principal (fig. 6) después de que se haya determinado el mayor elemento

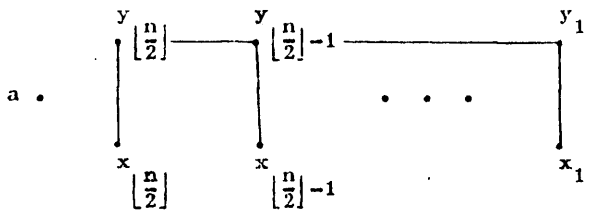


Fig. 6 Nivel 1 de la torre de comparaciones después de haber ordenado completamente los niveles siguientes. (Siendo n impar).

Dos situaciones pueden presentarse ahora:

o bien $\lfloor \frac{n}{2} \rfloor < t_{k(\lfloor \frac{n}{2} \rfloor)}$

o bien $\lfloor \frac{n}{2} \rfloor = t_{k(\lfloor \frac{n}{2} \rfloor)}$

En el primer caso el número de comparaciones es $k(\lfloor \frac{n}{2} \rfloor)$ o cero, -- ocurriendo esto último solo si a hubiera resultado ser el mayor - elemento de A puesto que entonces $y_{\lfloor \frac{n}{2} \rfloor}$ sería necesariamente el - segundo elemento.

En el segundo caso, el elemento $x_{\lfloor \frac{n}{2} \rfloor}$ ya ha sido insertado en la - cadena principal durante la determinación del mayor elemento. Todo lo cual puede expresarse por:

$$Y_2(n) = k(\lfloor \tfrac{n}{2} \rfloor) \cdot sg(t_{k(\lfloor \frac{n}{2} \rfloor)} - \lfloor \tfrac{n}{2} \rfloor) \quad \text{ó cero, para n impar}$$

c.q.d.

A partir de esto punto, no es difícil intuir en líneas generales el camino a seguir para determinar $Y_i(n)$, $i > 2$.

Nótese que si A no queda totalmente ordenada como consecuencia de la determinación de su segundo elemento, todas las comparaciones restantes tendrán lugar en el nivel 1 de la torre de -- comparaciones. En la figura 7 se observa lo que ocurre en dicho nivel en el proceso de obtención del segundo elemento de A, siendo n par. (El lector reconstruirá sin dificultad el proceso para n impar).

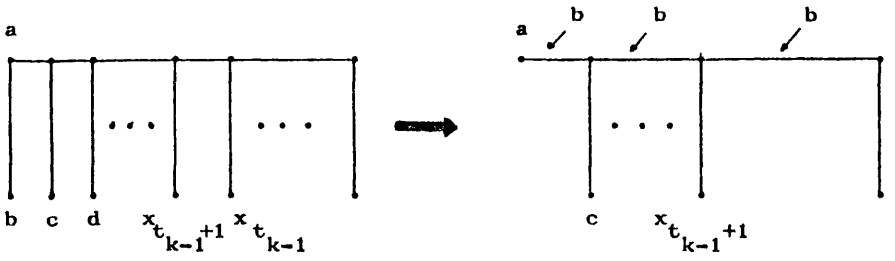


Fig. 7 Obtención del segundo elemento de A en el nivel 1 de la torre de comparaciones (n par). Las fechas indican posibles lugares de inserción de b. El índice k corresponde, evidentemente, a $k(\lfloor \frac{n}{2} \rfloor)$.

Si b se inserta en el lugar indicado por la flecha de la izquierda, el segundo y el tercer elemento se habrán determinado simultáneamente. En caso contrario, el tercer elemento de A se encontrará tras la inserción de c en la cadena principal. En forma análoga puede relacionarse la determinación del cuarto elemento de A con la inserción de d en la cadena principal y continuar el proceso hasta la inserción de $x_{t_{k-1}+1}$, después de lo cual A quedará completamente ordenado.

Las consideraciones realizadas y la definición siguiente nos permiten dar una expresión matemática para $Y_i(n)$, $i \geq 3$.

Definición I.4.7.-

a) $$p \dot{-} q = \begin{cases} p - q & \text{si} \quad p \geq q \\ 0 & \text{si} \quad p < q \end{cases}$$

b) $$\text{par}(n) = \begin{cases} 1 & \text{si} \quad n \text{ es impar} \\ 0 & \text{si} \quad n \text{ es par} \end{cases}$$

Teorema I.4.5.-

$$Y_i(n) = k(\lceil \tfrac{n}{2} \rceil) \cdot sg\left[(par(n) + 2(\lfloor \tfrac{n}{2} \rfloor - t_{k(\lceil \frac{n}{2} \rceil)-1})) \dot{-} i\right] \text{ ó cero} \qquad (4.23)$$

$$\text{para } i \geq 3 \quad \text{y} \quad n > 0$$

Prueba:

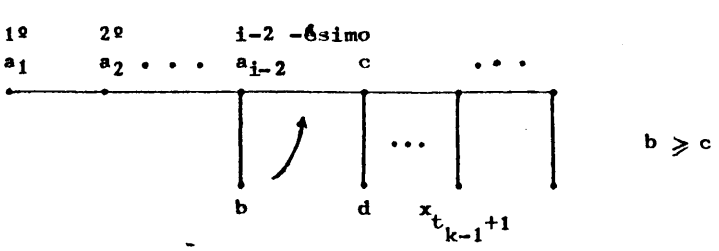
Para todo elemento x_j que, tras la determinación del segundo elemento de A, no haya sido aún insertado en la cadena principal se cumple

$$t_{k(\lceil \frac{n}{2} \rceil)} > \lfloor \tfrac{n}{2} \rfloor - 1 > j > t_{k(\lceil \frac{n}{2} \rceil)-1}$$

y en consecuencia el número de comparaciones necesarias para su inserción es de $k(\lceil \frac{n}{2} \rceil)$.

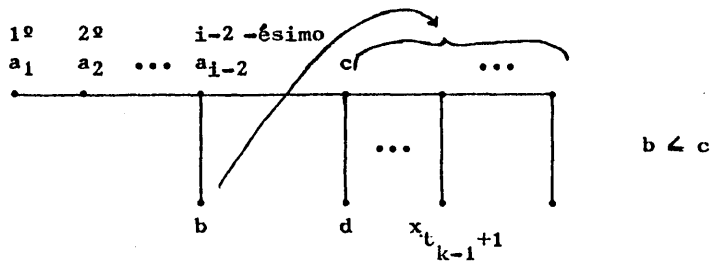
Supongamos ahora que ya se conocen los elementos 1º y 2º - de A. Si A no está aún totalmente ordenado, puede ocurrir que el - conocimiento del tercer elemento resulte inmediatemente de la determinación del segundo, y por tanto $Y_3(n) = 0$ (ver Fig. 7); en caso contrario $Y_3(n) = k(\lceil \frac{n}{2} \rceil)$. En general, el proceso de determinación del elemento i-1 -ésimo puede dar lugar a dos situaciones:

a)



En este caso, resulta $a_{i-1} = b$ y $a_i = c$; es decir, el elemento - i-ésimo ha sido determinado automáticamente con el i-1 -ésimo. Así pues, $Y_i(n) = 0$.

b)



En este caso se obtiene $a_{i-1} = c$, y será necesario insertar d en la cadena principal para conocer al elemento i-ésimo, Así:
$Y_i(n) = k(\lceil \frac{n}{2} \rceil)$.

Finalmente, cuando todos los elementos de la clase $\bar{k}(\lceil \frac{n}{2} \rceil)$ se hayan insertado en la cadena principal, la ordenación de A habrá terminado (Corolario I.4.2). Ahora bien, en el nivel 1 de la torre de comparaciones de A esta clase tiene $\lfloor \frac{n}{2} \rfloor - t_{k(\lceil \frac{n}{2} \rceil)-1}$ elementos si n es par, y en consecuencia la ordenación de A se producirá, a lo sumo - tras la determinación del elemento $2(\lfloor \frac{n}{2} \rfloor - t_{k(\lceil \frac{n}{2} \rceil)-1})$ -ésimo. (Lo - que corresponde a la repetición de la situación a)). Si n es impar, basta evidentemente sumar uno a esta expresión.

c.q.d.

Observación:

Si n es impar, la expresión (4.23) también es válida para

i=2, como puede comprobarse fácilmente.

Proposición I.4.4

$$\text{a)} \quad \sum_{i\geq 2} Y_i(n) = k(\lceil \tfrac{n}{2} \rceil) \cdot (\lfloor \tfrac{n}{2} \rfloor - t_{k(\lceil \frac{n}{2} \rceil)-1}) \qquad \text{si n es impar} \quad (4.24)$$

$$\text{b)} \quad \sum_{i\geq 3} Y_i(n) = k(\lceil \tfrac{n}{2} \rceil) \cdot (\lfloor \tfrac{n}{2} \rfloor - t_{k(\lceil \frac{n}{2} \rceil)-1} - 1) \qquad \text{si n es par} \quad (4.25)$$

Prueba:

Presentaremos únicamente la prueba de a), puesto que la de b) es análoga.

Después de la determinación del mayor elemento de A y siendo n impar, se necesita insertar aún $\lfloor \frac{n}{2} \rfloor - t_{k(\lceil \frac{n}{2} \rceil)-1}$ elementos en la cadena principal del nivel 1 de la torre de comparaciones para ordenar totalmente A. Por otro lado, si $\lfloor \frac{n}{2} \rfloor \in \bar{k}(\lceil \frac{n}{2} \rceil)$ cada uno de estos elementos necesita $k(\lceil \frac{n}{2} \rceil)$ comparaciones para ser insertado en dicha cadena. Si, por el contrario, $\lfloor \frac{n}{2} \rfloor \notin \bar{k}(\lceil \frac{n}{2} \rceil)$ entonces necesariamente $\lfloor \frac{n}{2} \rfloor = k(\lceil \frac{n}{2} \rceil) - 1 = k(\lfloor \frac{n}{2} \rfloor)$ y A ya está totalmente ordenado.

c.q.d.

Teorema I.4.6.-

$$U(n) = \sum_{i\geq 1} Y_i(n) \qquad \text{para todo } n > 0 \qquad (4.26)$$

Prueba :

Para n = 1 se verifica trivialmente que $Y_i(1)=0$ para $i \geq 1$. También se tiene $U(1) = 0$ (4.2).

Para n = 2 puede comprobarse que $Y_1(2) = 1$ (de(4.4)), $Y_i(2)=0$ para $i \geq 2$ (de(4.20) y (4.23)), y $U(2) = 1$ (de (4.2)).

Para $n \geq 3$:

Desarrollando (4.2) se obtiene:

$$U(n) = \left\lfloor \frac{n}{2} \right\rfloor + U\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + G\left(\left\lceil \frac{n}{2} \right\rceil\right)$$

$$= \sum_{i=1}^{L(n)} \left\lfloor \frac{n}{2^i} \right\rfloor + U(1) + \sum_{i=1}^{L(n)} G\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i-1}} \right\rfloor}{2} \right\rceil\right)$$

$$= \sum_{i=1}^{L(n)} \left\lfloor \frac{n}{2^i} \right\rfloor + \sum_{i=1}^{L(n)} G\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i-1}} \right\rfloor}{2} \right\rceil\right) \qquad (4.27)$$

Por otro lado, de (4.11), (4.20), (4.23) y (4.25), y suponiendo n par:

$$\sum_{i\geq 1} Y_i(n) = Y_1(n) + Y_2(n) + \sum_{i\geq 3} Y_i(n) =$$

$$= \sum_{i=1}^{L(n)} \left\lfloor \frac{n}{2^i} \right\rfloor + \sum_{i=i_0+1}^{L(n)} G\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i-1}} \right\rfloor}{2} \right\rceil\right) + H\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i_0-1}} \right\rfloor}{2} \right\rceil\right) +$$

$$+ G\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i_0-1}} \right\rfloor}{2} \right\rceil\right) - H\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i_0-1}} \right\rfloor}{2} \right\rceil\right) +$$

$$+ \sum_{i=2}^{i_0-1} G\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i-1}} \right\rfloor}{2} \right\rceil\right) + H\left(\left\lceil \frac{n}{2} \right\rceil\right) +$$

$$+ k\left(\left\lceil \frac{n}{2} \right\rceil\right) \cdot \left(\left\lfloor \frac{n}{2} \right\rfloor - t_{k\left(\left\lceil \frac{n}{2} \right\rceil\right)-1} - 1\right)$$

Reagrupando las sumas de G y desarrollando $H\left(\left\lceil \frac{n}{2} \right\rceil\right)$ mediante la expresión (4.6):

$$\sum_{i\geq 1} Y_i(n) = \sum_{i=1}^{L(n)} \left\lfloor \frac{n}{2^i} \right\rfloor + \sum_{i=2}^{L(n)} G\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i-1}} \right\rfloor}{2} \right\rceil\right) +$$

$$+ G\left(\left\lceil \frac{n}{2} \right\rceil\right) - k\left(\left\lceil \frac{n}{2} \right\rceil\right) \cdot \left(\left\lceil \frac{n}{2} \right\rceil - t_{k\left(\left\lceil \frac{n}{2} \right\rceil\right)-1} - 1\right) +$$

$$+ k\left(\left\lceil \frac{n}{2} \right\rceil\right) \left(\left\lfloor \frac{n}{2} \right\rfloor - t_{k\left(\left\lceil \frac{n}{2} \right\rceil\right)-1} - 1\right)$$

Teniendo en cuenta que si n es par $\left\lfloor \frac{n}{2} \right\rfloor = \left\lceil \frac{n}{2} \right\rceil$, resulta :

$$\sum_{i\geq 1} Y_i(n) = \sum_{i=1}^{L(n)} \left\lfloor \frac{n}{2^i} \right\rfloor + \sum_{i=1}^{L(n)} G\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i-1}} \right\rfloor}{2} \right\rceil\right) \qquad (4.28)$$

con lo que el teorema está probado para n par.

Si n es impar, de (4.11), (4.20), (4.23) y (4.24), y observando que

$i_0 = 1$, se obtiene:

$$\sum_{i\geq 1} Y_i(n) = Y_1(n) + \sum_{i\geq 2} Y_i(n)$$

$$= \sum_{i=1}^{L(n)} \left\lfloor \frac{n}{2^i} \right\rfloor + \sum_{i=2}^{L(n)} G\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i-1}} \right\rfloor}{2} \right\rceil \right) + H\left(\left\lceil \frac{n}{2} \right\rceil\right) +$$

$$+ k\left(\left\lceil \frac{n}{2} \right\rceil\right) \cdot \left(\left\lceil \frac{n}{2} \right\rceil - t_{k\left(\left\lceil \frac{n}{2} \right\rceil\right)-1} \right)$$

Desarrollando una vez más $H\left(\left\lceil \frac{n}{2} \right\rceil\right)$ según (4.6) :

$$\sum_{i\geq 1} Y_i(n) = \sum_{i=1}^{L(n)} \left\lfloor \frac{n}{2^i} \right\rfloor + \sum_{i=2}^{L(n)} G\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i-1}} \right\rfloor}{2} \right\rceil \right) + G\left(\left\lceil \frac{n}{2} \right\rceil\right) -$$

$$- k\left(\left\lceil \frac{n}{2} \right\rceil\right) \cdot \left(\left\lceil \frac{n}{2} \right\rceil - t_{k\left(\left\lceil \frac{n}{2} \right\rceil\right)-1} - 1 \right) +$$

$$+ k\left(\left\lceil \frac{n}{2} \right\rceil\right) \cdot \left(\left\lfloor \frac{n}{2} \right\rfloor - t_{k\left(\left\lceil \frac{n}{2} \right\rceil\right)-1} \right)$$

Fianlmente, por ser n impar, $\left\lfloor \frac{n}{2} \right\rfloor = \left\lceil \frac{n}{2} \right\rceil - 1$ y :

$$\sum_{i\geq 1} Y_i(n) = \sum_{i=1}^{L(n)} \left\lfloor \frac{2}{2^i} \right\rfloor + \sum_{i=1}^{L(n)} G\left(\left\lceil \frac{\left\lfloor \frac{n}{2^{i-1}} \right\rfloor}{2} \right\rceil \right) \qquad (4.29)$$

quedando el teorema probado también para n impar

c.q.d.

I.5 Dos cuestiones recíprocas

Consideramos ahora las dos cuestiones recíprocas siguientes:

1) Para cada $n \in \mathbb{Z}^+$, determinar el menor j tal que

$$U(n) = \sum_{i=1}^{j} Y_i(n) \qquad (5.1)$$

es decir, tal que la determinación de los j mayores elementos de A, mediante fusión-inserción, implique la ordenación total de A.

2) Para cada $j \in \mathbb{Z}^+$, encontrar los valores de n, $n \in \mathbb{Z}^+$, para los que

$$U(n) = \sum_{i=1}^{j} Y_i(n) > \sum_{i=1}^{j-1} Y_i(n) \qquad (5.2)$$

El interés de estos problemas estriba en que su resolución nos permitiría conocer las cardinalidades "buenas" y "malas" desde el punto de vista del comportamiento del algoritmo 3^*. Sin embargo, de los Teoremas I.4.4 y I.4.5 se desprende que la solución de 1) no está unívocamente determinada, dependiendo del orden de "presentación" o "aparición" de los elementos de A. Dichos problemas deben ser entonces replanteados en la forma siguiente:

1) Para cada $n \in \mathbb{Z}^+$, determinar los valores de j ($j \leq n$), para los que la siguiente relación es posible:

$$U(n) = \sum_{i=1}^{j} Y_i(n) > \sum_{i=1}^{j-1} Y_i(n) \qquad (5.3)$$

(Para que el problema esté correctamente planteado, para $n = 1$ debe exigirse únicamente la igualdad).

2) Para cada $j \in \mathbb{Z}^+$, determinar los valores de n para los que - (5.3) puede verificarse.

Definición I.5.1.-

Llamaremos $\bar{k}'$ a la clase de elementos de la cadena principal que están directamente unidos a los elementos de la clase $\bar{k}$.

Teorema I.5.1.-

Los valores de j que resuelven el problema 1) están dados por:

a) Para $n \in \{1, 2\}$, $j = 1$

b) Para $n > 2$,

$$\left\lceil \frac{n}{2} \right\rceil - t_{k(\left\lceil \frac{n}{2} \right\rceil)-1} + 1 - par(n) \leq j \leq \qquad (5.4)$$

$$\leq 2\left(\left\lceil \frac{n}{2} \right\rceil - t_{k(\left\lceil \frac{n}{2} \right\rceil)-1}\right) - par(n)$$

Prueba :

a) evidente

b) consideremos el nivel 1 de la torre de comparaciones, -

después de haber ordenado completamente los niveles siguientes, y - de haber insertado en la cadena principal, en forma sucesiva, las - clases $\bar{1}$, $\bar{2}$, ..., $\bar{k}(\lceil \frac{n}{2} \rceil) - 1$. Quedan por insertar solamente los - elementos de la clase $\bar{k}(\lceil \frac{n}{2} \rceil)$. Si todos los elementos de esta clase son menores que $y_{t_{k(\lceil \frac{n}{2} \rceil)-1}+1}$ (ver figura I.5.1), entonces se insertarán a la derecha de dicho elemento que será el $\left|\bar{k}^{-1}(\lceil \frac{n}{2} \rceil)\right|$ -ésimo mayor elemento de A. En consecuencia, recordando que los - elementos de $\bar{k}(\lceil \frac{n}{2} \rceil)$ se insertan en el orden $x_{\lceil \frac{n}{2} \rceil}$, $x_{\lceil \frac{n}{2} \rceil - 1}$, ..., la inserción de $x_{t_{k(\lceil \frac{n}{2} \rceil)-1}+1}$ determina el $\left|\bar{k}^{-1}(\lceil \frac{n}{2} \rceil)\right| + 1$ -ésimo elemento de A y, simultáneamente, la ordenación total de A.

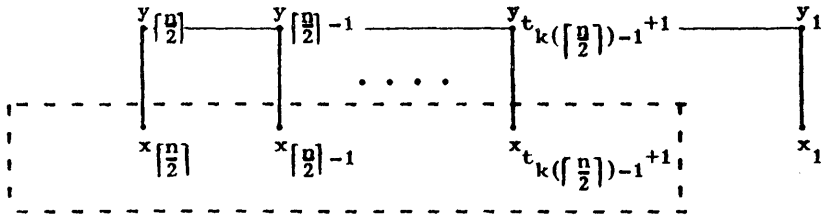


Fig. I.5.1 (a)

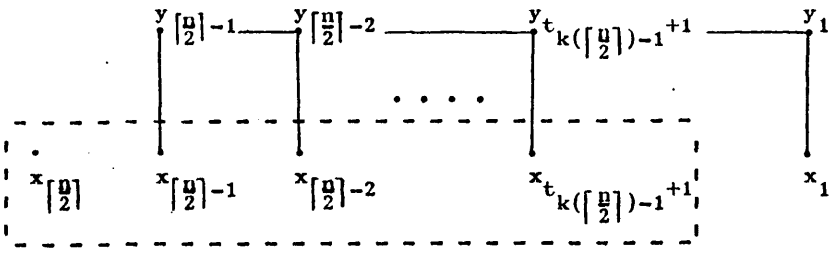


Fig. I.5.1 (b)

Fig. I.5.1 Los elementos en el rectángulo de linea interrumpida - son los de la clase $\bar{k}(\lceil \frac{n}{2} \rceil)$. En (a) se observa el caso n par y - en (b) el caso n impar.

Si todos los elementos de $\bar{k}(\lceil \frac{n}{2} \rceil)$, con la excepción evidente de -- $x_{t_{k(\lceil \frac{n}{2} \rceil)-1}+1}$, son mayores que $y_{t_{k(\lceil \frac{n}{2} \rceil)-1}+1}$ entonces se insertarán a su izquierda y este elemento será el $\left|\bar{k}(\lceil \frac{n}{2} \rceil)\right| - 1 + \left|\bar{k}'(\lceil \frac{n}{2} \rceil)\right|$ -ésimo mayor elemento de A. Por lo tanto la inserción de $x_{t_{k(\lceil \frac{n}{2} \rceil)-1}+1}$ determinará el $\left|\bar{k}(\lceil \frac{n}{2} \rceil)\right| + \left|\bar{k}'(\lceil \frac{n}{2} \rceil)\right|$ -ésimo elemento de A, así como su ordenación total.

Finalmente, si se consideran los casos intermedios, en que algunos elementos de $\bar{k}(\lceil \frac{n}{2} \rceil)$ son mayores que $y_{t_{k(\lceil \frac{n}{2} \rceil)-1}+1}$ y otros son menores o iguales, resulta que

$$\left|\bar{k}'(\lceil \frac{n}{2} \rceil)\right| + 1 \leq j \leq \left|\bar{k}(\lceil \frac{n}{2} \rceil)\right| + \left|\bar{k}'(\lceil \frac{n}{2} \rceil)\right| \qquad (5.5)$$

(Obsérvese que los elementos iguales a $y_{t_{k(\lceil \frac{n}{2} \rceil)-1}+1}$ pueden insertarse arbitrariamente a un lado u otro de este elemento).

Ahora, como

$$\left|\bar{k}(\lceil \tfrac{n}{2} \rceil)\right| = \lceil \tfrac{n}{2} \rceil - t_{k(\lceil \frac{n}{2} \rceil)-1} \qquad (5.6)$$

y

$$\left|\bar{k}'(\lceil \tfrac{n}{2} \rceil)\right| = \left|\bar{k}(\lceil \tfrac{n}{2} \rceil)\right| - \text{par}(n) \qquad (5.7)$$

se obtiene

$$\lceil \tfrac{n}{2} \rceil - t_{k(\lceil \frac{n}{2} \rceil)-1} - \text{par}(n) + 1 \leq j \leq 2(\lceil \tfrac{n}{2} \rceil - t_{k(\lceil \frac{n}{2} \rceil)-1}) - \text{par}(n) \qquad (5.8)$$

c.q.d.

El resultado que se ha obtenido, y cuyos valores numéricos se encuentran parcialmente tabulados en la tabla I.5.1 implica que para ciertos valores de n el algoritmo AS_3 "puede" ser más eficiente que el algoritmo AS_2. Así, por ejemplo, para $n = 22$, si una solución del problema propuesto en I.1 está dada por los k mayores elementos de A, y si $k < 7$, entonces AS_3 encontrará esta solución sin necesidad de ordenar totalmente A; si $7 \leq k < 12$, el orden de presentación de los elementos de A influirá en que AS_3 sea preferible o equivalente a AS_2; finalmente, si $k \geq 12$ ambos algoritmos ordenan totalmente A antes de encontrar la solución y tienen la misma

complejidad. Por otro lado, para $n = 23$ AS_2 y AS_3 son el mismo algoritmo.

n	j_{min}	j_{max}
1	1	1
2	1	1
3	1	1
4	2	2
5	2	3
6	3	4
7	1	1
8	2	2
9	2	3
10	3	4
11	1	1
12	2	2
13	2	3
14	3	4
15	3	5
16	4	6
17	4	7
18	5	8
19	5	9
20	6	10
21	6	11
22	7	12
23	1	1
24	2	2
25	2	3
26	3	4
27	3	5
28	4	6
29	4	7
30	5	8
31	5	9
32	6	10
33	6	11
34	7	12
35	7	13

Tabla I.5.1

Resultados del teorema I.5.1 para $1 \leq n \leq 35$

$$j_{min} = \left\lceil \frac{n}{2} \right\rceil - t_{k(\lceil \frac{n}{2} \rceil)-1} + 1 - par\ (n)$$

$$j_{max} = 2\left(\left\lceil \frac{n}{2} \right\rceil - t_{k(\lceil \frac{n}{2} \rceil)-1}\right) - par\ (n)$$

Complementaremos este resultado con la respuesta a la cuestión 2) formulada al principio de esta sección.

Lema I.5.1

$$n = 2t_{k(\lceil \frac{n}{2} \rceil)-1} + \left|\bar{k}(\lceil \tfrac{n}{2} \rceil)\right| + \left|\bar{k}'(\lceil \tfrac{n}{2} \rceil)\right| \qquad (5.9)$$

Prueba :

$$n = \sum_{i=1}^{k(\lceil \frac{n}{2} \rceil)-1} (|\bar{i}| + |\bar{i}'|) + \left|\bar{k}(\lceil \tfrac{n}{2} \rceil)\right| + \left|\bar{k}'(\lceil \tfrac{n}{2} \rceil)\right| \qquad (5.10)$$

Basta ahora observar que $|\bar{i}| = |\bar{i}'|$ para $i = 1, \ldots, k(\lceil \frac{n}{2} \rceil)-1$ y que

$$\sum_{i=1}^{k(\lceil \frac{n}{2} \rceil)-1} (|\bar{i}|) = t_{k(\lceil \frac{n}{2} \rceil)-1} \qquad (5.11)$$

para completar la prueba.

c.q.d.

Definición I.5.2.-

a) Llamaremos $\left[n_{min}\right]_j$ al conjunto de cardinalidades para las cuales A quedará totalmente ordenado, a lo sumo, con la determinación de su j-ésimo mayor elemento.

b) Análogamente, llamaremos $\left[n_{max}\right]_j$ al conjunto de cardinalidades tales que A pueda resultar totalmente ordenado con la determinación del j-ésimo mayor elemento, pero no antes.

Los dos teoremas siguientes caracterizan los conjuntos $[n_{min}]_j$ y $[n_{max}]_j$ arriba definidos.

Teorema I.5.2

a) $[n_{min}]_1 = \{1, 2, 2t_i+1 : i \geq 1\}$ (5.12)

b) Para todo $j > 1$

$$[n_{min}]_j = \{2t_i+j : t_{i+1} - t_i \geq \lceil \tfrac{j}{2} \rceil \}$$ (5.13)

Prueba :

Para $n = 1$ y $n = 2$ la demostración es trivial. Para $n > 2$, las tesis a) y b) adoptan la misma forma. Supongamos que A queda totalmente ordenado a lo sumo con la determinación de su -- j-ésimo mayor elemento. Esto implica que

$$j = \left| \bar{k}(\lceil \tfrac{n}{2} \rceil) \right| + \left| \bar{k}'(\lceil \tfrac{n}{2} \rceil) \right|$$ (5.14)

Por el lema anterior,

$$n = 2t_i + j$$ (5.15)

para i tal que

$$\left| \bar{k}(\lceil \tfrac{n}{2} \rceil) \right| \leq t_{i+1} - t_i$$ (5.16)

Por otro lado,

$$\left| \bar{k}'(\lceil \tfrac{n}{2} \rceil) \right| = \left| \bar{k}(\lceil \tfrac{n}{2} \rceil) \right| - par(n)$$ (5.17)

De (5.14) y (5.17) :

$$\left|\bar{k}\left(\left\lceil\frac{n}{2}\right\rceil\right)\right| = \left\lceil\frac{j}{2}\right\rceil \qquad (5.18)$$

Con lo que la condición (5.16) se convierte en

$$\left\lceil\frac{j}{2}\right\rceil \leq t_{i+1} - t_i \qquad (5.19)$$

c.q.d.

Teorema I.5.3.-

$$\text{a)}\ \left[n_{max}\right]_1 = \left\{1,\ 2,\ 2t_i + 1\ :\ i \geq 1\right\} \qquad (5.20)$$

b) Para todo $j > 1$

$$\left[n_{max}\right]_j = \left\{ 2(t_i + j - 1) + par(n)\ : \right.$$
$$\left.: t_{i+1} - t_i \geq j - 1 + par(n)\right\} \qquad (5.21)$$

Prueba :

Trivial para $n = 1$ y $n = 2$. Para $n > 2$, a) y b) coinciden. Supongamos que A puede resultar totalmente ordenado con la determinación de su j-ésimo elemento, pero no con la determinación de un elemento anterior. Entonces,

$$j = \left|\bar{k}'\left(\left\lceil\frac{n}{2}\right\rceil\right)\right| + 1 \qquad (5.22)$$

Por el lema I.5.1,

$$n = 2t_i + j - 1 + \left|\bar{k}\left(\left\lceil\frac{n}{2}\right\rceil\right)\right| \qquad (5.23)$$

donde i verifica

$$\left|\bar{k}\left(\left\lceil\frac{n}{2}\right\rceil\right)\right| \leq t_{i+1} - t_i \qquad (5.24)$$

Ahora, como

$$\left|\bar{k}\left(\left\lceil\frac{n}{2}\right\rceil\right)\right| = \left|\bar{k}'\left(\left\lceil\frac{n}{2}\right\rceil\right)\right| + \text{par}(n) \qquad (5.25)$$

sustituyendo dicha igualdad en (5.23) y utilizando (5.22),

$$n = 2(t_i + j - 1) + \text{par}(n) \qquad (5.26)$$

Finalmente, de (5.25), (5.22) y (5.24), i debe ser tal que

$$j - 1 + \text{par}(n) \leq t_{i+1} - t_i \qquad (5.27)$$

c.q.d.

Corolario I.5.1.-

Los valores de n que resuelven el problema 2) están dados por:

a) Para $j = 1$,

$$n \in \left\{1,\ 2,\ 2t_i + 1 \ :\ i \geq 1\right\} \qquad (5.28)$$

b) Para $j > 1$,

$$n = 2t_i + j \qquad (5.29)$$

si $j - 1 + \text{par}(n) > t_{i+1} - t_i \geq \left\lceil\frac{j}{2}\right\rceil$;

$$2t_i + j \leq n \leq 2(t_i + j - 1) + \text{par}(n) \qquad (5.30)$$

si $t_{i+1} - t_i \geq j - 1 + \text{par}(n)$

Prueba : inmediata

Como $t_{i+1} - t_i$ es función monótona creciente de i, para - tabular los resultados obtenidos en el corolario anterior resulta - conveniente conocer expresiones que permitan calcular fácilmente los valores mínimos del índice i que satisfacen las inecuaciones introducidas en (5.13) y (5.21). Las dos proposiciones siguientes dan respuesta a esta cuestión.

Proposición I.5.1

Sea i_1 (j, par(n)) el menor entero positivo para el que $t_{i+1} - t_i \geq j - 1 + par(n)$.

Entonces, para $j > 1$:

$$i_1(j, par(n)) = \begin{cases} \lfloor \log_2 3(j-1) \rfloor - 1 & \text{si n es par y } \lfloor \log_2 3(j-1) \rfloor = \log_2(3(j-1) - 2) \quad (5.31) \\ \lfloor \log_2 3(j-1) \rfloor + 1 & \text{si n es impar y } \lfloor \log_2 3(j-1) \rfloor = \log_2(3(j-1)+2) - 1 \quad (5.32) \\ \lfloor \log_2 3(j-1) \rfloor & \text{en los casos restantes} \quad (5.33) \end{cases}$$

Prueba:

Sea $j > 1$. De la definición de t_i (4.1), i_1 (j , par(n)) es el menor entero positivo para el que

$$2^{i+1} + 2(-1)^{i+1} \geq 3(j-1) + 3\ par(n) \qquad (5.34)$$

Sea $r = \lfloor \log_2 3(j-1) \rfloor$.

Se tiene entonces $2^r < 3(j-1) < 2^{r+1}$ (5.35)

a) Si se cumplen las condiciones (5.31), entonces $2^r + 2 = 3(j-1)$, y de (5.34) : $i_1(j, par(n)) = r - 1$

b) Si se cumplen las condiciones (5.32), entonces $2^{r+1} - 2 = 3(j-1)$, y de (5.34) : $i_1(j, par(n)) = r + 1$

c) Supongamos ahora que no se cumplen las condiciones (5.31) y (5.32).

Entonces, $2^r + 2 \neq 3(j-1)$ y $2^{r+1} \neq 3(j-1)$ (5.36)

Si r es un número par, entonces

$$3 \mid (2^r + 2) \quad y \quad 3 \mid (2^{r+1} - 2) \qquad (2) \qquad (5.37)$$

Además,

$$2^r + 2 < 3(j-1) < 2^{r+1} - 2 \qquad (5.38)$$

pues si no fuese así :

1º $2^r + 2 > 3(j-1)$ y la primera de las relaciones (5.37) implican $2^r > 3(j-1)$

2º $2^{r+1} - 2 < 3(j-1)$ y la segunda de las relaciones (5.37) implican $2^{r+1} < 3(j-1)$

Ambas conclusiones contradicen (5.35).

3º Finalmente la igualdad en (5.38) es imposible por (5.36)

En consecuencia, (5.38) y (5.34) implican $i_1(j, par(n)) = r$.

Por el contrario, si r es un número impar,
entonces

$$3 \mid (2^r - 2) \quad y \quad 3 \mid (2^{r+1} + 2), \qquad (5.39)$$

(2)

relaciones que son compatibles con la consecuencia (5.36) de la hipótesis c).

Finalmente, de (5.35),

$$2^r - 2 < 3(j - 1) + 1 < 2^{r+1} + 2 \qquad (5.40)$$

y utilizando (5.39) y (5.34) se concluye que $i_1(j, par(n)) = r$

c.q.d.

Proposición I.5.2

Sea $i_0(j)$ el menor entero positivo para el que $t_{i+1} - t_i \geq \lceil \frac{j}{2} \rceil$.

Entonces, para $j > 1$:

$$i_0(j) = \begin{cases} \left\lfloor \log_2 3\lceil \frac{j}{2} \rceil \right\rfloor - 1 & \text{si } \left\lfloor \log_2 3\lceil \frac{j}{2} \rceil \right\rfloor = \log_2(3\lceil \frac{j}{2} \rceil - 2) \\ \left\lfloor \log_2 3\lceil \frac{j}{2} \rceil \right\rfloor & \text{en caso contrario} \end{cases} \qquad (5.41)$$

Prueba :

Análoga a la de la proposición I.5.1

Con los resultados de las proposiciones anteriores y del Corolario I.5.1 se puede tabular la solución del problema 2) (ver tabla I.5.2).

En esta tabla, a la derecha de cada valor del índice $j \neq 1$ se representan, en la fila superior los elementos del conjunto $\left[n_{min}\right]_j$ definidos para $i_0(j) \leq i \leq 8$, y en la segunda fila los elementos de $\left[n_{max}\right]_j$ correspondientes a $i_1(j, par(n)) \leq i \leq 8$. Para $j = 1$, se ha utilizado una sola fila pues $\left[n_{min}\right]_1 = \left[n_{max}\right]_1$; sus dos primeros elementos (1 y 2) aparecen solos en las primeras columnas de la tabla pues están definidos independientemente del índice i.

j \ i	1	2	3	4	5	6	7	8
1	1 2	3	7	11	23	43	87	171	343
2	4	8	12	24	44	88	172	344
	4 5	8 9	12 13	24 25	44 45	88 89	172 173	344 345
3	5	9	13	25	45	89	173	345
	6	10	14 15	26 27	46 47	90 91	174 175	346 347
4	6	10	14	26	46	90	174	346
			16 17	28 29	48 49	92 93	176 177	348 349
5			15	27	47	91	175	347
			18 19	30 31	50 51	94 95	178 179	350 351
6			16	28	48	92	176	348
			20 21	32 33	52 53	96 97	180 181	352 353
7			17	29	49	93	177	349
			22	34 35	54 55	98 99	182 183	354 355
8			18	30	50	94	178	350
				36 37	56 57	100 101	184 185	356 357
9			19	31	51	95	179	351
				38 39	58 59	102 103	186 187	358 359
10			20	32	52	96	180	352
				40 41	60 61	104 105	188 189	360 361
11			21	33	53	97	181	353
				42	62 63	106 107	190 191	362 363
12			22	34	54	98	182	354
					64 65	108 109	192 193	364 365
13				35	55	99	183	355
					66 67	110 111	194 195	366 367
14				36	56	100	184	356
					68 69	112 113	196 197	368 369
15				37	57	101	185	357
					70 71	114 115	198 199	370 371

Tabla I.5.2

Resultados del Corolario I.5.1 para $1 \leq j \leq 15$ y $1 \leq i \leq 8$.

NOTAS A PIE DE PAGINA DEL CAPITULO I

(1) Desde el punto de vista del caso más desfavorable.

(2) Estas relaciones pueden probarse fácilmente por inducción sobre r.

56 [illegible]

C A P I T U L O II

DETERMINACION DE LA COMPLEJIDAD DE UNA GENERALIZACION DEL PROBLEMA, Y ANALISIS DE UN NUEVO ALGORITMO PARA RESOLVERLO.

II.0 Introducción

En este capítulo se estudiará una generalización del problema considerado en el capítulo precedente.

Sea A una sucesión finita en $\mathbb{Z}^+$, $s \in \mathbb{N}$, y $S = \left\{B : B \subseteq A \text{ y } \sum_{x \in B} x \geq s\right\}$; evidentemente, S está ordenado parcialmente por la relación de inclusión. Nos interesará ahora encontrar la familia $S' \subseteq S$ de elementos minimales por inclusión en S. Así, en el ejemplo de la sección I.1, $A = \{2, 3, 5, 1, 8, 3\}$; $s = 11$, se tiene $S' = \{\{3, 8\}, \{5, 8\}, \{1, 2, 8\}, \{3, 3, 5\}, \{1, 2, 3, 5\}\}$. Nótese que 3 aparece dos veces en A, pero $\{3, 8\}$ figura una sola vez en S'. Es decir consideraremos S' como conjunto, y no como multiconjunto.

En la sección II.1 se formaliza el problema y se introduce la notación básica necesaria. A continuación, en la sección II.2 se estudia la complejidad intrínseca del problema. Finalmente, en la sección II.3 se adapta convenientemente el algoritmo de Hammer y Rudeanu [1970] de resolución de ecuaciones pseudo-booleanas, y se modifica para mejorar su eficiencia en función de la multiplicidad de los elementos de A.

II.1 Formalización y notación.

Sea $\mathbf{c} = \{c_1, c_2, \ldots, c_n\} \in \mathbb{Z}^{+n}$, $n > 0$

$s \in N$, dados.

Problema: determinar la familia

$$T = \left\{ I : I \subseteq \{1, 2, \ldots, n\} , \sum_{i \in I} c_i \geq s , \right.$$

$$\left. (\forall J) \left[J \subset I \Rightarrow \sum_{i \in J} c_i < s \right] \right\}$$

En una sección posterior se verá que el problema así propuesto es muy general, puesto que el caso $\mathbf{c} \in \mathbb{Z}^n$ puede ser reducido en forma sencilla al caso arriba definido.

Análogamente puede definirse un problema simétrico del anterior:

encontrar

$$T' = \left\{ I : I \subseteq \{1, 2, \ldots, n\} , \sum_{i \in I} c_i \leq s , \right.$$

$$\left. (\forall J) \left[I \subset J \subseteq \{1, 2, \ldots, n\} \Rightarrow \sum_{i \in J} c_i > s \right] \right\}.$$

En lo que sigue, se preferirá la notación $T_{\geq}(\mathbf{c}, s)$ y $T_{\leq}(\mathbf{c}, s)$ para designar a las familias buscadas en los dos problemas simétricos propuestos, salvo si los argumentos $\mathbf{c}$, s y (o) los símbolos " $\geq$, $\leq$ " están implícitos en el contexto. Excepcionalmente puede resultar también cómodo trabajar con la notación

$C_{\geq}(\mathbf{c}, s) = \left\{ C : C = \{c_i\}_{i \in I} , I \in T_{\geq}(\mathbf{c}, s) \right.$, notación que -

en general representa a un multiconjunto. El significado de $T_{>}(c, s)$, $T_{<}(c, s)$ y $T_{=}(c, s)$ es claro en sí mismo. Finalmente, se designará por $\bar{I}$ al complemento de I con respecto a $\{1, 2, \ldots, n\}$.

El teorema siguiente relacionará entre sí a $T_{\geqslant}$ y $T_{\leqslant}$, dando simultáneamente un medio de determinar una de estas familias cuando se conoce la otra.

Teorema II.1.1.-

$$I \in T_{\leqslant}(c, s) \quad \text{sii} \quad \bar{I} \in T_{\geqslant}(c, \sum_{i=1}^{n} c_i - s)$$

Prueba :

Para todo $I \subseteq \{1, 2, \ldots, n\}$ se tiene:

$$\sum_{i \in I} c_i \leqslant s \iff -\sum_{i \in I} c_i \geqslant -s$$

$$\iff -(\sum_{i=1}^{n} c_i - \sum_{i \in \bar{I}} c_i) \geqslant -s$$

$$\iff \sum_{i \in \bar{I}} c_i \geqslant \sum_{i=1}^{n} c_i - s \qquad (2.1)$$

Sean I, J tales que $I \in T_{\leqslant}(c, s)$ y $J \subset \bar{I}$.

Entonces, $I \subset \bar{J}$ y

a) $\sum_{i \in I} c_i \leqslant s$

b) $\sum_{i \in \bar{J}} c_i > s$

utilizando ahora (2.1), se tiene

c) $\sum_{i \in \bar{I}} c_i \geq \sum_{i=1}^{n} c_i - s$

d) $\sum_{i \in J} c_i < \sum_{i=1}^{n} c_i - s$

En consecuencia, $\bar{I} \in T_{\geq}(c, \sum_{i=1}^{n} c_i - s)$

La prueba del recíproco es análoga.

c.q.d.

El ejemplo siguiente ilustrará el teorema anterior.

<u>Ejemplo II.1.1</u>.-

Sea $c = \{26, 9, 9, 7, 5, 5, 5, 4, 3\}$; $s = 47$

Con lo cual $\sum_{i} c_i - s = 73 - 47 = 26$.

Si se conoce una (cualquiera) de las familias $T_{\geq}(c, \sum_{i=1}^{n} c_i - s)$, $T_{\leq}(c, s)$, la otra puede entonces determinarse encontrando el complemento de cada I con respecto a $\{1, 2, \ldots, n\}$ (Ver tabla II.1.1).

$T_{\gg}(\mathbf{c}, \sum_i c_i - s)$		$T_{\ll}(\mathbf{c}, s)$	
$\{1\}$	$[26]_{26}$	$\{2,3,4,5,6,7,8,9\}$	$[9,9,7,5,5,5,4,3]_{47}$
$\{2,3,4,5\}$	$[9,9,7,5]_{30}$	$\{1,6,7,8,9\}$	$[26,5,5,4,3]_{43}$
$\{2,3,4,6\}$		$\{1,5,7,8,9\}$	
$\{2,3,4,7\}$		$\{1,5,6,8,9\}$	
$\{2,3,4,8\}$	$[9,9,7,4]_{29}$	$\{1,5,6,7,9\}$	$[26,5,5,5,3]_{44}$
$\{2,3,4,9\}$	$[9,9,7,3]_{28}$	$\{1,5,6,7,8\}$	$[26,5,5,5,4]_{45}$
$\{2,4,5,6\}$	$[9,7,5,5]_{26}$	$\{1,3,7,8,9\}$	$[26,9,5,4,3]_{47}$
$\{2,4,5,7\}$		$\{1,3,6,8,9\}$	
$\{2,4,6,7\}$		$\{1,3,5,8,9\}$	
$\{3,4,5,6\}$		$\{1,2,7,8,9\}$	
$\{3,4,5,7\}$		$\{1,2,6,8,9\}$	
$\{3,4,6,7\}$		$\{1,2,5,8,9\}$	
$\{2,4,5,8,9\}$	$[9,7,5,4,3]_{28}$	$\{1,3,6,7\}$	$[26,9,5,5]_{45}$
$\{2,4,6,8,9\}$		$\{1,3,5,7\}$	
$\{2,4,7,8,9\}$		$\{1,3,5,6\}$	
$\{3,4,5,8,9\}$		$\{1,2,6,7\}$	
$\{3,4,6,8,9\}$		$\{1,2,5,7\}$	
$\{3,4,7,8,9\}$		$\{1,2,5,6\}$	
$\{2,5,6,7,8\}$	$[9,5,5,5,4]_{28}$	$\{1,3,4,9\}$	$[26,9,7,3]_{45}$
$\{3,5,6,7,8\}$		$\{1,2,4,9\}$	
$\{2,5,6,7,9\}$	$[9,5,5,5,3]_{27}$	$\{1,3,4,8\}$	$[26,9,7,4]_{46}$
$\{3,5,6,7,9\}$		$\{1,2,4,8\}$	
$\{4,5,6,7,8\}$	$[7,5,5,5,5]_{26}$	$\{1,2,3,9\}$	$[26,9,9,3]_{47}$

Tabla II.1.1

A la derecha de cada conjunto I de índices se encuentra, entre paréntesis la subsucesión correspondiente de $\mathbf{c}$, con un subíndice igual $\sum_{i \in I} c_i$.

Proposición II.1.1

El tiempo necesario para obtener $T_{\leqslant}(c, s)$ a partir de $T_{\geqslant}(c, \sum_i c_i - s)$ (o vice versa), está acotado superiormente por $O(n \mid T_{\leqslant}(c, s) \mid)$.

Prueba :

Evidente.

II.2 . Complejidad intrínseca del problema

En esta sección se dan argumentos que hacen pensar en la inverosimilitud de que existan algoritmos convenientes (en el sentido de Edmonds - ver Cap. 0 -) para determinar $T_{\geqslant}(c, s)$.

Consideremos el conjunto $T_{=}(c, s)$:

Teorema II.2.1.-

El problema de decidir si $T_{=}(c, s) \neq \emptyset$ (1) es NP-completo

Prueba :

Definimos los problemas siguientes

p_1 : problema de la satisfactibilidad.

p_2 : problema de la satisfactibilidad, cuyas claúsulas c_i tienen al menos tres literales.

p_3 : problema del número cromático (dados un grafo $G = (X, U)$ y

$n \in \mathbb{Z}^+$, determinar si es posible asignar n colores a los vértices de G sin que dos vértices adyacentes reciban el mismo color).

p_4 : problemas de la partición (dados un conjunto $A=\{a_1, a_2, .., a_p\}$ y una familia $\{B_i\} \subseteq \mathcal{P}(A)$, determinar si existe una sub-familia $\{C_j\} \subseteq \{B_i\}$ que sea una partición de A

p_5 : determinar si $T_{=}(c, s) \neq \emptyset$.

El teorema resulta de probar que

$$p_1 \propto p_2 \propto p_3 \propto p_4 \propto p_5$$

La prueba es técnica y se incluye en la última sección de este capítulo.

c.q.d.

Teorema II.2.2.-

Sean A y B dos conjuntos cualesquiera. Entonces, bastan $|A| \cdot |B|$ comparaciones para determinar si $A \cap B = \emptyset$.

Prueba :

Inmediata, considerando comparaciones elemento por elemento.

c.q.d.

Observación:

E. Reingold [1972] probó que si se autorizan únicamente comparaciones de elementos de A con elementos de B, del tipo " $=, \neq$ " , entonces este algoritmo es optimal. Si, por el contraio, se permite sumar elementos así como comparar entre sí elementos de un mismo con

junto, entonces bastan $O(n \log n)$ comparaciones, con $n = \max(|A|, |B|)$.

Teorema II.2.3.-

Si existen algoritmos para determinar $T_{\geq}(c, s)$ y $T_{\leq}(c, s)$ en tiempo polinomial, entonces P = NP.

Prueba :

Si se pueden encontrar $T_{\geq}(c, s)$, y $T_{\leq}(c, s)$ en tiempo polinomial, lo mismo ocurrirá con $T_{=}(c, s)$, puesto que $T_{=}(c, s) = T_{\geq}(c, s) \cap T_{\leq}(c, s)$. En particular, se habrá decidido en tiempo polinomial si $T_{=}(c, s) \neq \emptyset$. De lo que, por el Teorema II.2.1 y la Def. 0.4.1 resulta P = NP.

c.q.d.

Este resultado nos permite, como ya es clásico, realizar la reflexión siguiente; dado que:

a) diversos problemas NP-completos de áreas tan distintas como teoría de grafos, lógica matemática, teoría de lenguajes formales, etc., han sido objeto de amplios estudios por especialistas de dichas áreas, sin que por ello se haya encontrado un algoritmo polinomial para alguno de esos problemas;

b) si P = NP entonces existirían algoritmos polinomiales para todos los problemas NP-completos.
Podemos entonces concluir del teorema anterior la inverosimilitud de que exista un algoritmo polinomial para encontrar $T_{\geq}(c, s)$.

Finalmente, el corolario siguiente relaciona las complejidades intrínsecas de los problemas correspondientes a la determinación de $T_{\geq}$, $T_{>}$, $T_{\leq}$, $T_{<}$ y $T_{=}$

Corolario II.2.1.-

Dados los problemas de la determinación de $T_{\geqslant}$, $T_{>}$, $T_{\leqslant}$, $T_{<}$ y $T_{=}$, entonces, o bien existen algoritmos polinomiales para cada uno de estos problemas, o bien no los hay para ninguno de ellos.

Prueba :

El corolario es consecuencia de la proposición II.1.1, del Teorema II.2.3 y de las igualdades siguientes:

1) $T_{=}(\mathfrak{c}, s) = T_{\geqslant}(\mathfrak{c}, s) \bigcap T_{\leqslant}(\mathfrak{c}, s)$

2) $T_{>}(\mathfrak{c}, s) = T_{\geqslant}(\mathfrak{c}, s+1)$

3) $T_{<}(\mathfrak{c}, s) = T_{\leqslant}(\mathfrak{c}, s-1)$

4) $T_{\geqslant}(\mathfrak{c}, s) = \bigcup_{i=0}^{\sum c_i - s} T_{=}(\mathfrak{c}, s+i)$

5) $T_{\leqslant}(\mathfrak{c}, s) = \bigcup_{i=0}^{s-1} T_{=}(\mathfrak{c}, s+i)$

1), 2) y 3) son evidentes. La igualdad 4) resulta de

$$T_{\geqslant}(\mathfrak{c}, s) = T_{>}(\mathfrak{c}, s) \bigcup T_{=}(\mathfrak{c}, s)$$

$$= T_{\geqslant}(\mathfrak{c}, s+1) \bigcup T_{=}(\mathfrak{c}, s) \text{ , por 2)}$$

.

.

.

$$= \bigcup_{i=0}^{\sum c_i - s} T_{=}(\mathbf{c}, s + i)$$

La obtención de 5) es similar.

c.q.d.

A la vista de estos resultados, interesará especialmente el desarrollo de algoritmos cuyos comportamientos asintóticos, dentro de la exponencialidad, tengan cotas reducidas (dentro del comportamiento exponencial, es evidentemente preferible un algoritmo de complejidad $O(2^n)$ frente a otro de complejidad $O(2^{2^n})$). También es interesante el estudio de algoritmos heurísticos que, para algún concepto "aceptable" de solución aproximada, resuelvan estos problemas en tiempo polinomial. Finalmente, una tercera vía de investigación es posible (¡y deseable!) : estudiar la dificultad intrínseca de los problemas de determinar $T_{\geq}$, $T_{\leq}$ etc., desde el punto de vista de la complejidad media de los algoritmos que los resuelven, y no, como hemos hecho hasta aquí, desde el punto de vista del caso más desfavorable. La primera de estas líneas será la que seguiremos en el resto del capítulo, a partir de la idea básica de aprovechar la posible multiplicidad de los elementos de A. La segunda vía propuesta no será contemplada en este trabajo, pero pensamos que debe ser fructífera, ya que el uso de heurísticas se ha revelado particularmente útil para el problema de la mochila (ver Garey y Johnson [1979]). La tercera de estas vías presenta la dificultad del desconocimiento de la distribución de probabilidad de los distintos casos de un problema de tamaño n dado, y abre el

camino al estudio de algoritmos probabilísticos. La escasez de resultados referente a la complejidad de este tipo de algoritmos es la que ha motivado las investigaciones expuestas a partir del capítulo siguiente.

II.3 Un problema equivalente

Estudiaremos aquí un problema equivalente al nuestro, y una modificación de un algoritmo de Hammer y Rudeanu [1970] para mejorar su eficiencia.

Definición II.3.1.-

Sean $c \in \mathbb{Z}^n$, $s \in \mathbb{Z}$.

Llamaremos ecuación pseudo-booleana a

$$c \cdot x = s \qquad (3.1)$$

e inecuación pseudo-booleana a

$$c \cdot x \geq s \qquad (3.2)$$

$$c \cdot x > s \qquad (3.3)$$

donde x es un vector de variables booleanas sobre $\{0, 1\}^n$ y " $\geq$ ", " $>$ " pueden intercambiarse por " $\leq$ " y " $<$ " respectivamente.

Se observará que toda inecuación del tipo (3.3) puede convertirse en una del tipo (3.2) sumando 1 a s. Además, podemos limitarnos al caso en que $c \in \mathbb{Z}^{+n}$. En efecto, si llamamos $\bar{x}_i$ a la variable complemento de x_i, basta utilizar la identidad

$x_i \equiv 1 - \bar{x}_i$ siempre que se tenga $c_i < 0$. Así, por ejemplo, la inecuación

$$c \cdot x \geq s \quad , \qquad c \in \mathbb{Z}^k$$

se transforma en

$$c' \cdot x' \geq s'$$

con $$c'_i = \begin{cases} c_i & \text{si} \quad c_i > 0 \\ -c_i & \text{si} \quad c_i < 0 \end{cases}$$

$$x'_i = \begin{cases} x_i & \text{si} \quad c_i > 0 \\ \bar{x}_i & \text{si} \quad c_i < 0 \end{cases}$$

$$s' = s - \sum_{c_i<0} c_i$$

Análogamente, el caso en que el vector de incógnitas pueda contener variables complementadas puede reducirse a ecuaciones o inecuaciones con coeficientes en $\mathbb{Z}^{+n}$. (2)

Definición II.3.2.-

Sea $x^* \in \{0, 1\}^n$ una solución de (3.2)

$I \subseteq \{1, 2, \ldots, n\}$

$$\sum (x^*, I) = \{x \in \{0, 1\}^n \ : \ x_i = x_i^* \ \ \forall \ i \in I\} \quad (3.4)$$

Diremos que $\sum (x^*, I)$ es la familia de soluciones de (3.2) generada por (x^*, I), si $\forall x \in \sum (x^*, I)$, x es solución de (3.2).

Observación: es evidente que no todo (x^*, I) genera una familia de soluciones.

Las propiedades siguientes son inmediatas:

1) $\sum (x^*, I) = 2^{n - |I|}$

2) $x^* \in \sum (x^*, I)$

3) $\sum (x^*, I) = \{x^*\}$ sii $I = \{1, 2, \ldots, n\}$; en este caso, diremos que la familia es degenerada.

Definición II.3.3.-

x^* es una solución fundamental de (3.2) si

a) $c \cdot x^* \geq s$

b) Para todo $i = 1, \ldots, n$:

$$x_i^* = 1 \implies c \cdot x'^* < s$$

$$\text{donde } x'_j = \begin{cases} x_j & \text{si } j \neq i \\ 0 & \text{si } j = i \end{cases}$$

Es claro que las soluciones de la ecuación $c \cdot x = s$ también son soluciones fundamentales de (3.2).

Teorema II.3.1.-

Sea $I \subseteq \{1, 2, \ldots, n\}$. Una condición necesaria y suficiente para que $I \in T_{\geqslant}(c, s)$ es que x^*, definida por $x_i^* = 1$ sii $i \in I$, sea una solución fundamental de (3.2)

Prueba :

$$\left.\begin{array}{l} \sum_{i \in I} c_i \geqslant s \\ \quad y \\ \sum_{i \in J} c_i < s \quad ; \quad \forall J \subset I \end{array}\right\} \Longleftrightarrow \begin{cases} c \cdot x^* \geqslant s \ , & x_i^* = \begin{cases} 1 & \text{si } i \in I \\ 0 & \text{en caso contrario} \end{cases} \\ \\ c \cdot x'^* < s \ , & x_i'^* = \begin{cases} 1 & \text{si } i \in J \subset I \\ 0 & \text{en caso contrario} \end{cases} \\ \qquad\qquad \forall J \subset I \end{cases}$$

c.q.d.

En consecuencia, el problema de encontrar la familia $T_{\geqslant}(c, s)$ es equivalente al problema de determinar todas las soluciones fundamentales de la inecuación pseudo-booleana (3.2).Además, una vez conocida $T_{\geqslant}(c, s)$ se obtiene directamente :

Corolario II.3.1.-

Sea $T_{\geqslant}(c, s) = \{I_1, I_2, \ldots, I_m\}$. Entonces, toda solución de (3.2) pertenece a una y solo una de las familias

$\sum (x^*_j , I_j)$, $\quad 1 \leq j \leq m$.

Complementos a la notación :

Si $\mathfrak{x} = \{x_1, x_2, \ldots, x_n\}$, designaremos por $\mathfrak{x}^{(i,j)}$, $1 \leq i \leq j \leq n$, a $\{x_i, x_{i+1}, \ldots, x_j\}$. Por lo tanto $\mathfrak{x}^{(1,n)} = \mathfrak{x}$ y $\mathfrak{x}^{(i,i)} = x_i$. Una notación análoga se usará para $\mathfrak{c}$. También utilizaremos cuando sea conveniente $\mathbb{0}^{(1,j)}$ para el caso en que las j componentes sean nulas.

II.4 Algoritmos

El algoritmo más intuitivo posible para el problema en estudio es, tal vez, el siguiente:

Algoritmo A [Generar $\mathcal{P}(\mathfrak{c})$ y determinar $T_{\geqslant}(\mathfrak{c}, s)$]

1.- [Preparación]

$\mathfrak{x} \longleftarrow \mathbb{0}^{(1,n)}$; $T_{\geqslant}(\mathfrak{c}, s) \longleftarrow \emptyset$

2.- [Iteraciones y finalización]

Mientras $\mathfrak{c} \cdot \mathfrak{x} < s$: $\mathfrak{x} \longleftarrow \mathfrak{x} + 1$ (suma en base binaria)

si $x_i = 1$ $\forall_i$: FIN

$i \longleftarrow 1$

mientras $i \leq n$:

si $x_i = 1$: $\tilde{\mathfrak{x}} \longleftarrow \mathfrak{x} - (\mathbb{0}, x_i, \mathbb{0})$

si $\mathfrak{c} \cdot \tilde{\mathfrak{x}} \geqslant s$: $\mathfrak{x} \longleftarrow \mathfrak{x} + 1$ (suma en base binaria) volver al principio del paso 2.

$i \longleftarrow i + 1$

$I \longleftarrow \{j : x_j = 1\}$

$T_{\geqslant}(c, s) \longleftarrow T_{\geqslant}(c, s) \cup \{I\}$

si $x_i = 1$ V_i : FIN

$\alpha \longleftarrow \alpha + 1$ (suma en base binaria)

volver al principio del paso 2.

El algoritmo A no es más que un procedimiento "ciego" de enumeración que realiza el segundo paso $2^n - 1$ veces y que no aprovecha la información ganada cada vez que se obtiene una solución; en efecto, si α es una solución fundamental, ya no es necesario examinar las α' tales que $x_i = 1 \Longrightarrow x'_i = 1$. Esta es la clave del algoritmo siguiente, debido a Hammer y Rudeanu [1970]. Dicho algoritmo precisa que c esté ordenado en sentido decreciente; trabajaremos, pues, con inecuaciones

$$c \cdot \alpha \geqslant s \quad ; \quad c_i \geqslant c_j \quad \text{para} \quad 1 \leqslant i \leqslant j \leqslant n \qquad (3.5)$$

La idea fundamental consiste en considerar, para cada $i = 1, 2, .., n$, la existencia de soluciones que tengan a c_i como mayor elemento y, en caso afirmativo resolver los sub-problemas determinados por las inecuaciones

$$c^{(j,n)} \geqslant s - c_i \quad ; \quad i + 1 \leqslant j \leqslant n \qquad (3.6)$$

lo que corresponde al recorrido de un árbol T de raiz r definido recursivamente por

$$\begin{aligned} T &= (r, \{T_i\}_{i=1}^{n}) \quad ; \\ T_i &= (c_i, \{T_j\}_{j=i+1}^{n}) \quad ; \quad 1 \leqslant i \leqslant n \end{aligned} \qquad (3.7)$$

donde c_i es la raiz de T_i, $\left\{T_j\right\}_{j=i}^{n}$ son los sub-árboles relacionados directamente con c_i, para $i = 1, \ldots, n$ y $c_0 = r$ (ver figura 1)

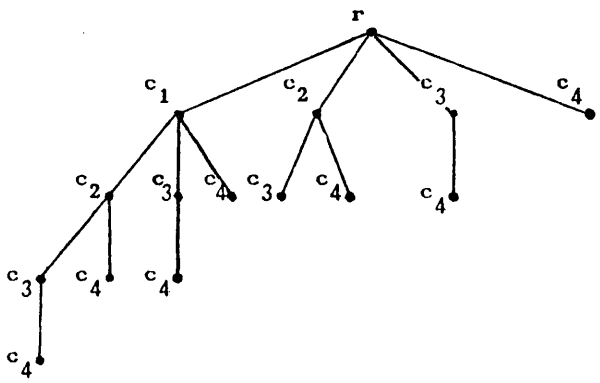


Fig. 1 $T = (r, \left\{T_i\right\}_{i=1}^{4})$

Algoritmo HR

Determinar $T_{\geq}(c, s)$ recursivamente. Utiliza una variable auxiliar AUX para transmitir a cada sub-problema obtenido los elementos de $\{1, 2, \ldots, n\}$ que hay que añadir a las soluciones de dicho sub-problema para obtener soluciones del problema inicial. Inicialmente, AUX = $\emptyset$.

1.- [Preparación]

p ⟵ 1 (índice que varía de 1 a n)

SP ⟵ 0 (variable que contiene sumas parciales de elementos de c).

2.- Obtención de soluciones inmediatas

Mientras $c_p > s$:

escribir la solución AUX $\cup \{p\}$

si $p = n$: FIN

$p \longleftarrow p + 1$

3.- [Suma de elemntos menores que s]

Para $i = p$ **hasta** n :

$SP \longleftarrow SP + c_i$

4.- [Comparar SP con s y obtener otras soluciones inmediatas]

Si $SP < s$: FIN

si no : **si** $SP = s$:

escribir la solución AUX $\cup \{p, p + 1, \ldots, n\}$

FIN

si no : **si** $p = n - 1$:

escribir la solución AUX $\cup \{n - 1, n\}$

FIN

si no : $SP \longleftarrow SP - c_p$

5.- [Resolución de sub-problemas]

5.1.- [Obtención de soluciones que no contienen a c_p]

Si $SP > s$:

resolver $T_{\geq}\ (e^{(p+1, n)}, s)$

si no : si SP = s

escribir la solución AUX $\cup \{p+1, \ldots, n\}$

5.2.- [Obtención de soluciones que contienen a c_p]

AUX $\longleftarrow$ AUX $\cup \{p\}$

resolver $T_{\geq}(\mathbf{c}^{(p+1,n)}, s - c_p)$

FIN

Enunciaremos algunos lemas en que se basa la demostración de que el algoritmo HR es correcto. Sus pruebas respectivas son muy simples y no se presentarán aquí (ver por ejemplo Hammer y Rudeanu [1970]).

Lema II.4.1.-

Si $I \in T_{\geq}(\mathbf{c}, s)$ y $A_i = I \cap \{1, 2, \ldots, i\}$

entonces, para todo $i = 1, 2, \ldots, n-1$ se tiene

$I - A_i \in T_{\geq}(\mathbf{c}^{(i+1,n)}, s - \sum_{j \in A_i} c_j)$.

Lema II.4.2.-

Si $I \in T_{\geq}(\mathbf{c}^{(i+1,n)}, s)$ entonces $I \in T_{\geq}(\mathbf{c}, s)$.

Lema II.4.3.-

Si $s > 0$ y $I \in T_{\geq}(\mathbf{c}^{(2,n)}, s - c_1)$ entonces,

$I \cup \{1\} \in T_{\geq}(\mathbf{c}, s)$

Teorema II.4.1.-

Sea $(c, s) \in \mathbb{Z}^{+n} \times \mathbb{Z}$, con c ordenado en forma decreciente.

a) Si $s \leq 0$ entonces $T_{\geq}(c, s) = \{\emptyset\}$

Si $s > 0$:

b) Si $\exists\, p \in \{1, 2, \ldots, n-1\}$ tal que $c_p \geq s > c_{p+1}$ entonces

1) $\forall j \in \{1, 2, \ldots, p\}$ se tiene $\{j\} \in T_{\geq}(c, s)$

2) El conjunto de soluciones restantes de $T_{\geq}(c, s)$ es $T_{\geq}(c^{(p+1, n)}, s)$

c) Si $c_n \geq s$ entonces $T_{\geq}(c, s) = \{\{1\}, \{2\}, \ldots, \{n\}\}$

d) Si $s > c_1$, se distinguen cuatro casos :

1) Si $\sum_{i=1}^{n} c_i < s$ entonces $T_{\geq}(c, s) = \emptyset$

2) Si $\sum_{i=1}^{n} c_i = s$ entonces $T_{\geq}(c, s) = \{\{1, 2, \ldots, n\}\}$

3) Si $\sum_{i=1}^{n} c_i > s$ y $\sum_{i=2}^{n} c_i < s$ entonces

$$T_{\geq}(c, s) = \{I : I = \{1\} \cup I',\ \forall I' \in T_{\geq}(c^{(2,n)}, s - c_1)\}$$

4) Si $\sum_{i=1}^{n} c_i > s$ y $\sum_{i=2}^{n} c_i \geq s$ entonces

$$T_{\geq}(c, s) = T_{\geq}(c^{(2,n)}, s) \cup$$

$$\cup \{I : I = \{1\} \cup I',\ \forall I' \in T_{\geq}(c^{(2,n)}, s - c_1)\}$$

En lo que respecta al problema simétrico (determinar $T_{\leqslant}(\mathbf{c}, s)$), puede enunciarse el siguiente teorema:

Teorema II.4.2.-

Sea $(\mathbf{c}, s) \in \mathbb{Z}^{+n} \times \mathbb{Z}$, con $\mathbf{c}$ ordenado en forma decreciente.

a) Si $s < 0$, entonces $T_{\leqslant}(\mathbf{c}, s) = \emptyset$

Si $s \geqslant 0$:

b) Si $c_n > s$ entonces $T_{\leqslant}(\mathbf{c}, s) = \{\emptyset\}$

c) Si $\exists\, p \in \{1, 2, \ldots, n-1\}$ tal que $c_p > s \geqslant c_{p+1}$ entonces

$$T_{\leqslant}(\mathbf{c}, s) = T_{\leqslant}(\mathbf{c}^{(p+1, n)}, s)$$

d) Si $\sum_{i=1}^{n} c_i \leqslant s$ entonces $T_{\leqslant}(\mathbf{c}, s) = \{\{1, 2, \ldots, n\}\}$

e) Si $\sum_{i=1}^{n} c_i > s$ y $\sum_{i=2}^{n} c_i \leqslant s$ entonces

$$T_{\leqslant}(\mathbf{c}, s) = \{\{2, 3, \ldots, n\}\} \cup \{I : I = \{1\} \cup I',\ \forall I' \in T_{\leqslant}(\mathbf{c}^{(2,n)}, s - c_1)\}$$

f) Si $\sum_{i=1}^{n} c_i > s$ y $\sum_{i=2}^{n} c_i > s$ entonces

$$T_{\leqslant}(\mathbf{c}, s) = T_{\leqslant}(\mathbf{c}^{(2,n)}, s) \cup \{I : I = \{1\} \cup I',\ \forall I' \in T_{\leqslant}(\mathbf{c}^{(2,n)}, s - c_1)\}$$

La prueba de este teorema es análoga a la del Teorema II.4.1 e implica la posibilidad de utilizar para este problema un algoritmo similar al algoritmo HR.

El algoritmo HR es en general más eficiente que el algoritmo A, que siempre realiza $O(2^n)$ operaciones elementales (sumas y comparaciones entre dos números) :

Teorema II.4.3.-

El número mínimo de operaciones elementales necesarias para determinar $T_{\geq}(c, s)$ mediante el algoritmo HR es de $O(n)$.

Prueba :

Esta cota inferior de la complejidad del algoritmo HR se alcanza en los siguientes casos:

a) si $c_{n-2} \geq s$

b) si $\sum_{i=1}^{n} c_i \leq s$

c) si $c_p \geq s > c_{p+1}$ y $\sum_{i=p+1}^{n} c_i \leq s$ para algún $p < n - 2$

En efecto, en los tres casos la solución del problema se determina en la primera iteración del algoritmo, en los pasos 2 ó 4. Los pasos 2 y 3 consumen, juntos, $O(n)$ operaciones elementales, y el paso 4 realiza, a lo sumo, 3 comparaciones. Lo cual prueba la complejidad lineal de HR en los casos a), b) y c) . Todos los casos restantes necesitan, al menos, resolver además un sub-problema. Por -

lo tanto, O(n) es efectivamente cota inferior estrictade la complejidad de HR.

c.q.d.

Es fácil, por otro lado, encontrar casos en os que el algoritmo HR tiene complejidad exponencial (como podíapreverse de los resultados de la sección 2 de este capítulo). Estoes consecuencia directa del siguiente resultado:

Proposición II.4.1.-

$$\max_{(c,s)\in \mathbb{Z}^{+n}\times\mathbb{Z}} \left| T_{\geq}(c,s)\right| = \binom{n}{\lfloor \frac{n}{2}\rfloor}$$

Prueba :

La proposición resulta de constatar que

$\max_t \binom{n}{t} = \binom{n}{\lfloor \frac{n}{2}\rfloor} = \binom{n}{\lceil \frac{n}{2}\rceil}$ (pues si n es par, $\lfloor \frac{n}{2}\rfloor = \lceil \frac{n}{2}\rceil$ y si no,

$\lfloor \frac{n}{2}\rfloor < \frac{n}{2} < \lceil \frac{n}{2}\rceil$; además para todo n $\lceil \frac{n}{2}\rceil = n - \lfloor \frac{n}{2}\rfloor$) , y tel lema siguiente:

Lema II.4.4.-

Dado n > 0, la cardinalidad de $T_{\geq}(c,s)$ esmáxima sii

para todo $I, J \in T_{\geq}(c,s)$, $|I| = |J| = \begin{cases} \lfloor \frac{n}{2}\rfloor & \delta \\ \lceil \frac{n}{2}\rceil \end{cases}$

La prueba de este lema se presenta en la última sección del capítulo.

Teorema II.4.4.-

Si $| T_{\geq}(c, s)| = \binom{n}{\lfloor \frac{n}{2} \rfloor}$ entonces el número de iteraciones realizadas por el algoritmo HR es de $O(2^n)$.

Prueba :

Sea $I \in T_{\geq}(c, s)$. Si $1 < |I| < n$, el algoritmo HR determina I en $O(|I|)$ iteraciones (consecuencia del Teorema II.4.1, d-3) y d-4)). Si, además $| T_{\geq}(c, s)| = \binom{n}{\lfloor \frac{n}{2} \rfloor}$ entonces para todo $I \in T_{\geq}(c, s)$ $|I| = \lfloor \frac{n}{2} \rfloor$ ($\delta|I| = \lceil \frac{n}{2} \rceil$) y se necesitarán $O(\frac{n}{2}\binom{n}{\lfloor \frac{n}{2} \rfloor}) = O(2^n)$ iteraciones para determinar $T_{\geq}(c, s)$

c.q.d.

Observación:

Si c no estuviese totalmente ordenado, la complejidad total del proceso aumentaría en $O(n \log_2 n)$ (ver Capítulo 1, sección 3), valor que solamente influiría en la cota inferior establecida en el Teorema II.4.3.

II.5 Optimización del algoritmo HR

Aunque el algoritmo HR acelera la búsqueda de soluciones, recurriendo esencialmente a la evaluación de sumas de ciertos elementos de $\mathbf{c}$, no tiene cuenta de la posible repetición de elementos. Si por ejemplo, el problema está dado por $\mathbf{c}^{(1,n)} = (2, 2, \ldots, 2)$ y un umbral s tal que $n \geq 2t \geq s > 2(t-1) \geq 0$, para algún $t > 0$, hay $\binom{n}{t}$ soluciones idénticas, cada una de las cuales se obtiene separadamente. Es, pues, importante que el algoritmo utilizado, pueda agrupar soluciones idénticas, calculando solo una.

Notación: Supondremos, para fijar ideas, que

$$\mathbf{c}^{(1,n)} = (c_1, \ldots, c_{m_1}, c_{m_1+1}, \ldots, c_{m_1+m_2}, \ldots, c_{\sigma(k-1)+1}, \ldots, c_{\sigma(k)})$$

con $c_1 = \ldots = c_{m_1} > c_{m_1+1} = \ldots = c_{m_1+m_2} > \ldots > c_{\sigma(k-1)+1} = \ldots = c_{\sigma(k)}$,

donde, para simplificar la escritura, $\sigma(j) = \sum_{i=1}^{j} m_i$ (3), $1 \leq j \leq k$. Es claro que $\sigma(k) = \sum_{i=1}^{k} m_i = n$. Representaremos el multiconjunto $\{\underbrace{a, a, \ldots, a}_{p\text{-veces}}\}$ por $[p\,;\,a]$, y $[p\,;\,a] \cup [q\,;\,b]$ corresponderá al multiconjunto $\{\underbrace{a, a, \ldots, a}_{p\text{-veces}}, \underbrace{b, b, \ldots, b}_{q\text{-veces}}\}$

Definición II.5.1.-

Sea $\mathbf{c}^{(1,n)} \in \mathbb{Z}^{+n}$. Dados $i, j \in \{1, 2, \ldots, n\}$, diremos que $i \equiv j$ si $c_i = c_j$. (Es evidente que la relación " $\equiv$ "

así definida es una relación de equivalencia). Llamaremos $\mathcal{E}(i)$ a la clase de equivalencia de i .

Definición II.5.2.-

Sea $\{\alpha_i\}_{i=1}^{t} \in \mathcal{P}(\{1, 2, \ldots, n\})$. Diremos que $\bigcup_{i=1}^{t}[p_i ; \alpha_i]$ es irreductible si $i \neq j \Longrightarrow \alpha_i \neq \alpha_j$ para $1 \leq i,j \leq t$.

Es inmediato que siempre es posible escribir $\bigcup_{i=1}^{t}[p_i ; \alpha_i]$ en forma irreductible, puesto que si existe $\{\alpha_{h_i}\}_{i=1}^{r} \subseteq \{\alpha_i\}_{i=1}^{t}$ tal que $\alpha_{h_i} \equiv \alpha_{h_j}$ para $1 \leq i,j \leq r$, basta reemplazar $\bigcup_{i=1}^{r}[p_{h_i} ; \alpha_{h_i}]$ por $\left[\sum_{i=1}^{r} p_{h_i} ; \alpha_{h_1}\right]$.

Consideremos ahora la extensión siguiente de la relación de equivalencia " $\equiv$ " :

Definición II.5.3.-

Sean $\{\alpha_i\}_{i=1}^{t}$, $\{\beta_i\}_{i=1}^{t} \in \mathcal{P}(\{1, 2, \ldots, n\})$ y sean $\bigcup_{i=1}^{t}[p_i ; \alpha_i]$ y $\bigcup_{i=1}^{t}[p_i ; \beta_i]$ irreductibles. Entonces $\bigcup_{i=1}^{t}[p_i ; \alpha_i] \equiv \bigcup_{i=1}^{t}[p_i ; \beta_i]$ si $\alpha_i = \beta_i$ para $1 \leq i \leq t$.

Definición II.5.4.-

Llamaremos elemento canónico de $\mathcal{E}(\bigcup_{i=1}^{t}[p_i ; \alpha_i])$ a

$\bigcup_{i=1}^{t} [p_i ; \gamma_i]$ si para $1 \leq i \leq t$ y para todo $\beta_i \in \mathcal{E}(\{\alpha_i\})$ se tiene $\gamma_i \geq \beta_i$

De las definiciones anteriores y de la notación introducida al principio de esta sección se observa que los elementos canónicos son de la forma $\bigcup_{i=1}^{t} [p_i ; \sum_{h=1}^{r_i} m_h]$.

En lo que sigue, consideraremos que todo $I \in T_{\geqslant}(c, s)$ está en forma irreductible. Se tiene entonces:

Proposición II.5.1.-

$$I \in T_{\geqslant}(c, s) \iff \mathcal{E}(I) \subseteq T_{\geqslant}(c, s)$$

Prueba :

Trivial.

Puesto que $\{c_i\}_{i \in I_1} = \{c_i\}_{i \in I_2}$ si $I_1 \equiv I_2$, en vez de buscar $T_{\geqslant}(c, s)$ nos limitaremos a determinar el multiconjunto $T'_{\geqslant}(c, s)$ que resulta de reemplazar cada I de $T_{\geqslant}(c, s)$ por el elemento canónico de $\mathcal{E}(I)$.

Definición II.5.5.-

Sea $I \in T'_{\geqslant}(c, s)$. Llamaremos multiplicidad de la solución I a $\mu(I) = |\mathcal{E}(I)|$.

Proposición II.5.2.-

$$\text{Si} \quad I = \bigcup_{i=1}^{t} \left[p_i \; ; \; \sum_{h=1}^{r_i} m_h \right] , \quad \text{entonces} \quad \mu(I) = \prod_{i=1}^{t} \binom{m_{r_i}}{p_i}$$

Prueba :

$$|\mathcal{E}(I)| = \prod_{i=1}^{t} \left| \mathcal{E}\left(\sum_{h=1}^{r_i} m_h \right) \right|$$

$$= \prod_{i=1}^{t} \binom{m_{r_i}}{p_i}$$

c.q.d.

Ejemplo II.5.1.-

Sea $\mathbf{c} = (11, 9, 9, 7, 7, 7, 7, 5, 5, 5, 3, 3)$; $s = 42$

Entonces : $I_1 = [1 ; 3] \cup [3 ; 7] \cup [2 ; 10] \cup [1 ; 12]$ es una solución, y su multiplicidad es $\mu(I_1) = \binom{2}{1} \times \binom{4}{3} \times \binom{3}{2} \times \binom{2}{1} = 48$;

$I_2 = [1 ; 1] \cup [2 ; 3] \cup [2 ; 7]$ es otra solución, que tiene multiplicidad $\mu(I_2) = \binom{1}{1} \times \binom{2}{2} \times \binom{4}{2} = 6$.

Para completar el desarrollo de la idea principal de esta sección asociaremos a $\Gamma_{\geq}(\mathbf{c}, s)$ un conjunto $\Gamma_{\geq}(\tilde{\mathbf{c}}, s)$, definiendo $\tilde{\mathbf{c}}$ por medio de la transformación:

$$\tilde{c}_{\sigma(i-1)+j} = w_{ij} \, c_{\sigma(i)} \quad ; \quad 1 \leq j \leq m_i \; , \quad 1 \leq i \leq k \qquad (5.1)$$

donde $w_{ij} = m_i - j + 1 \quad ; \quad 1 \leq j \leq m_i \; , \quad 1 \leq i \leq k \qquad (5.2)$

Llamaremos a $\tilde{\mathbf{c}}$ vector ponderado asociado a $\mathbf{c}$.

De este modo, cada clase de elementos iguales de $\mathbf{c}$ se transforma en otra que contiene, ordenados en forma decreciente, todas las sumas diferentes posibles de elementos de la clase primitiva. En cuanto al umbral s, se mantiene igual.

Nos proponemos resolver el problema original con la ayuda del problema asociado que acabamos de definir. Con este fin empezaremos por caracterizar la relación entre $T'_{\geqslant}(\mathbf{c}, s)$ y $T_{\geqslant}(\tilde{\mathbf{c}}, s)$ para finalmente enunciar y probar un teorema (basado en esta relación) que está en la línea de los Teoremas II.4.1 y II.4.2.

Proposición II.5.3.-

$$\text{Sea} \quad A = \bigcup_{i=1}^{t} \left[w_{u_i v_i} \; ; \; \sum_{h=1}^{u_i} m_h \right]$$

$$B = \left\{ \sum_{h=1}^{u_1 - 1} m_h + v_1 \; , \; \sum_{h=1}^{u_2 - 1} m_h + v_2 , \; \ldots , \; \sum_{h=1}^{u_t - 1} m_h + v_t \right\}$$

$$\text{con} \quad 1 \leq t \leq n \quad , \quad 1 \leq v_i \leq m_{u_i} \quad , \quad 1 \leq u_i \leq n$$

Entonces $\sum_{i \in A} c_i = \sum_{i \in B} \tilde{c}_i$

Prueba :

$\sum_{i \in A} c_i = \sum_{i=1}^{t} w_{u_i v_i} c_{\sigma(u_i)}$. De esta igualdad y usando (5.1) y (5.2) se obtiene :

$$\sum_{i\in A} c_i = \sum_{i=1}^{t} \tilde{c}_{\sigma(u_i - 1) + v_i} = \sum_{i\in B} \tilde{c}_i$$

c.q.d.

Esta propiedad permite identificar cada solución de $T'_{\geq}(c, s)$ con una y solo una solución de $T_{\geq}(\tilde{c}, s)$. Sin embargo, la recíproca no es cierta en general. La proposición siguiente caracteriza las soluciones de $T_{\geq}(\tilde{c}, s)$ que nos interesan. Supondremos, sin pérdida de generalidad, que para todo i, $u_i < u_{i+1}$.

Proposición II.5.4.-

Sea $1 \leq t \leq n$, $1 \leq v_i \leq m_{u_i}$, $1 \leq u_i \leq n$. Entonces,

$$\bigcup_{i=1}^{t} \left[w_{u_i v_i} ; \sum_{h=1}^{u_i} m_h \right] \in T'_{\geq}(c, s) \quad \text{sii}$$

a) $$\left\{ \sum_{h=1}^{u_i - 1} m_h + v_i \right\}_{i=1}^{t} \in T_{\geq}(\tilde{c}, s)$$

b) Para cada $r \in \{1, 2, \ldots, t\}$:

1) Si $v_r < m_{u_r}$ entonces

$$\left\{ \sum_{h=1}^{u_i - 1} m_h + v_i \ , \ \sum_{h=1}^{u_2 - 1} m_h + v_2 \ , \ \ldots, \ \sum_{h=1}^{u_r - 1} m_h + v_r + 1, \ \ldots \right.$$

$$\left. \ldots, \ \sum_{h=1}^{u_t - 1} m_h + v_t \right\} \notin T_{\geq}(\tilde{c}, s)$$

2) Si $v_r = m_{u_r}$ entonces

$$\left\{ \sum_{h=1}^{u_1-1} m_h + v_1 \, , \ldots, \sum_{h=1}^{u_{r-1}-1} m_h + v_{r-1}, \sum_{h=1}^{u_{r+1}-1} m_h + v_{r+1}, \ldots \right.$$

$$\left. \ldots; \sum_{h=1}^{u_t-1} m_h + v_t \right\} \notin T_{\geqslant}(\tilde{c}, s)$$

Prueba :

Supongamos que $v_i < m_{u_i}$ para $i = 1, 2, \ldots, t$

$$\bigcup_{i=1}^{t} \left[w_{u_i v_i} \; ; \sum_{h=1}^{u_i} m_h \right] \in T'_{\geqslant}(c, s) \iff$$

$$\iff \sum^{t} w_{u_i v_i} c_{\sigma(u_i)} \geqslant s > \sum_{\substack{i=1 \\ i \neq r}}^{t} w_{u_i v_i} c_{\sigma(u_i)} + w_{u_r v_r + 1} \, c_{\sigma(u_r)}$$

para $1 \leq r \leq t$

$$\iff \sum_{i=1}^{t} \tilde{c}_{\sigma(u_i - 1) + v_i} \geqslant s > \sum_{\substack{i=1 \\ i \neq r}}^{t} \tilde{c}_{\sigma(u_i - 1) + v_i} + \tilde{c}_{\sigma(u_r - 1) + v_r + 1}$$

para $1 \leq r \leq t$

$$\Longleftrightarrow \begin{cases} \left\{ \sum_{h=1}^{u_i-1} m_h + v_i \right\}_{i=1}^{t} \quad e \quad T_{\geq}\,(\tilde{c}\,,\,s) \\ \\ \qquad\qquad y \\ \\ \left\{ \sum_{h=1}^{u_1-1} m_h + v_1,\ \sum_{h=1}^{u_2-1} m_h + v_2,\ \ldots,\ \sum_{h=1}^{u_r-1} m_h + v_r + 1 \ \ldots \right. \\ \left. \ldots,\ \sum_{h=1}^{u_t-1} m_h + v_t \right\} \quad \notin \quad T_{\geq}\,(\tilde{c}\,,\,s) \end{cases}$$

Por otro lado, si $v_r = m_{u_r}$ para algún $r \in \{1, 2, .., t\}$:

$$\bigcup_{i=1}^{t} \left[w_{u_i v_i} \,;\, \sum_{h=1}^{u_i} m_h \right] \in T'_{\geq}\,(c\,,\,s) \Longleftrightarrow$$

$$\Longleftrightarrow \sum_{i=1}^{t} w_{u_i v_i}\, c_{\sigma(u_i)} \geq s > \sum_{\substack{i=1 \\ i \neq r}}^{t} w_{u_i v_i}\, c_{\sigma(u_i)}$$

$$\Longleftrightarrow \sum_{i=1}^{t} \tilde{c}_{\sigma(u_i-1)+v_i} \geq s > \sum_{\substack{i=1 \\ i \neq r}}^{t} \tilde{c}_{\sigma(u_i-1)+v_i}$$

$$\Longleftrightarrow \begin{cases} \left\{ \sum_{h=1}^{u_i - 1} m_h + v_i \right\}_{i=1}^{t} \in T_{\geqslant} (\tilde{c}, s) \\ \\ \text{y} \\ \\ \left\{ \sum_{h=1}^{u_1 - 1} m_h + v_1, \ \ldots, \ \sum_{h=1}^{u_{r-1} - 1} m_h + v_{r-1}, \ \sum_{h=1}^{u_{r+1} - 1} m_h + v_{r+1}, \ \ldots \right. \\ \\ \left. \ldots, \ \sum_{h=1}^{u_t - 1} m_h + v_t \right\} \notin T_{\geqslant} (\tilde{c}, s) \end{cases}$$

c.q.d.

Basándonos en estos resultados y en las siguientes generalizaciones de los Lemas II.4.1 y II.4.3 [4] es posible ya determinar (c, s), a partir del conocimiento de $\tilde{c}$ y calculando solo el elemento canónico de cada clase de soluciones $\mathcal{E}(I)$ así como la multiplicidad asociada, como se verá en el Teorema II.5.1.

Notación :

Sea $\Pi \subset \{1, 2, \ldots, n\}$; designaremos por $c^{(\Pi)}$ al vector obtenido eliminando de c todo c_i tal que $i \in \Pi$; $c^{(\bar{\Pi})}$ se obtiene representando por $\bar{\Pi}$ al complemento de Π con respecto a $\{1, 2, \ldots, n\}$.

Lema II.5.1.-

Si $I \in T_{\geqslant} (c, s)$ entonces, para todo $\Pi \subset \{1, 2, \ldots, n\}$: $I - \Pi^{*} \in T_{\geqslant} (c^{(\bar{\Pi})}, s - \sum_{j \in I \cap \Pi} c_j)$.

Prueba :

Por definición

$$I \in T_{\geqslant}(c, s) \Longrightarrow \begin{cases} a) & \sum_{i \in I} c_i \geqslant s \\ b) & J \subset I \Longrightarrow \sum_{i \in J} c_i < s \end{cases}$$

Sea $\Pi \subset \{1, 2, \ldots, n\}$. De a), se tiene

$$\sum_{i \in I-\Pi} c_i = \sum_{i \in I} c_i - \sum_{i \in I \cap \Pi} c_i \geqslant s - \sum_{i \in I \cap \Pi} c_i$$

Por otro lado, para cualquier $H \subset I - \Pi$ y de b), se tiene

$$\sum_{i \in H} c_i = \sum_{i \in H \cup (I \cap \Pi)} c_i - \sum_{i \in I \cap \Pi} c_i < s - \sum_{i \in I \cap \Pi} c_i$$

Y utilizando una vez más la definición de $T_{\geqslant}$ el lema queda probado.

c.q.d.

Lema II.5.2.-

Sea $s > 0$, $j \in \{1, 2, \ldots, n\}$ y $\Pi = \{1, 2, \ldots, n\} - \{j\}$. Entonces,

$$\left.\begin{array}{l} I \in T_{\geqslant}(e^{(\Pi)}, s - c_j) \\ \quad y \\ \sum_{i \in I} c_i < s \end{array}\right\} \Longrightarrow I \cup \{j\} \in T_{\geqslant}(c, s)$$

Prueba :

Por definición,

$$I \in T_{\geqslant}(c^{(\pi)},\ s - c_j) \Longrightarrow \begin{cases} a) & \sum_{i\in I} c_i \geqslant s - c_j \\ b) & J \subset I \Longrightarrow \sum_{i\in J} c_i < s - c_j \end{cases}$$

De a) se obtiene directamente

$$\sum_{i\in I} c_i + c_j \geqslant s \qquad (5.3)$$

De b), de la hipótesis $\sum_{i\in I} c_i < s$ y suponiendo $j \notin I$, resulta

$$\sum_{i\in I} c_i - \min_{i\in I\cup\{j\}} \{c_i\} + c_j < s \qquad (5.4)$$

desigualdad que implica

$$\sum_{i\in H} c_i < s \qquad \text{para todo } H \subset I\cup\{j\} \qquad (5.5)$$

Finalmente, de (5.3), (5.5) y de la definición de $T_{\geqslant}$ se concluye que $I\cup\{j\} \in T_{\geqslant}(c, s)$

c.q.d.

Observación :

Mediante modificaciones muy simples, los lemas previos siguen siendo válidos si se sustituye $T_{\geqslant}$ por $T'_{\geqslant}$.

Estamos ahora en condiciones de probar el teorema siguiente.

Teorema II.5.1.-

Sea $(\mathbf{c}, s) \in \mathbb{Z}^{+n} \times \mathbb{Z}$, con $\mathbf{c}$ ordenado en forma decreciente, y sea $\tilde{\mathbf{c}}$ el vector ponderado asociado a $\mathbf{c}$.

a) Si $s \leq 0$ entonces $T'_{\geqslant}(\mathbf{c}, s) = \{\emptyset\}$

Si $s > 0$:

b) Si existe $B \subseteq \{1, 2, \ldots, k\}$ no vacío tal que para todo $i \in B$,

$\tilde{c}_{\sigma(i-1)+1} \geqslant s$ entonces:

1) $\forall i \in B$ se tiene $\left[w_{ir_i} \; ; \; \sum_{h=1}^{i} m_h \right] \in T'_{\geqslant}(\mathbf{c}, s)$, y su multiplicidad es $\binom{m_i}{w_{ir_i}}$, donde r_i es el mayor entero de $\{1, 2, \ldots, m_i\}$ para el que $\tilde{c}_{\sigma(i-1)+r_i} \geqslant s$.

2) El conjunto de soluciones restantes es $T'_{\geqslant}(\mathbf{d}, s)$, donde $\mathbf{d}$ se obtiene de $\mathbf{c}$ suprimiendo para cada $i \in B$ r_i elementos de la clase $\mathcal{E}\left(\sum_{h=1}^{i} m_h\right)$.

c) Si $s > \tilde{c}_{\sigma(i-1)+1}$ para $1 \leq i \leq k$, se distinguen cuatro casos:

1) Si $\sum_{i=1}^{k} \tilde{c}_{\sigma(i-1)+1} < s$ entonces $T'_{\geq}(\mathfrak{c}, s) = \emptyset$

2) Si $\sum_{i=1}^{k} \tilde{c}_{\sigma(i-1)+1} = s$ entonces $T'_{\geq}(\mathfrak{c}, s) = \bigcup_{i=1}^{k} \left[m_i ; \sum_{h=1}^{i} m_h \right]$

3) Si $\sum_{i=1}^{k} \tilde{c}_{\sigma(i-1)+1} > s$ y $\sum_{i=2}^{k} \tilde{c}_{\sigma(i-1)+1} < s$:

sea $t \in \{1, 2, \ldots, m_1\}$ tal que

$$\tilde{c}_t + \sum_{i=2}^{k} \tilde{c}_{\sigma(i-1)+1} \geq s > (w_{1t}-1)\, \tilde{c}_{m_1} + \sum_{i=2}^{k} \tilde{c}_{\sigma(i-1)+1} .$$

Entonces, para $1 \leq j \leq t$,

si $I \in T'_{\geq}(\mathfrak{c}^{(m_1+1,\, n)}, s - \tilde{c}_j)$ y su multiplicidad es μ,

se tiene $\left[w_{1j} ; m_1 \right] \cup I \in T'_{\geq}(\mathfrak{c}, s)$ y su multiplicidad

es $\binom{m_1}{w_{1j}} \mu$.

No hay otras soluciones de $T'_{\geq}(\mathfrak{c}, s)$.

4) Si $\sum_{i=1}^{k} \tilde{c}_{\sigma(i-1)+1} > s$ y $\sum_{i=2}^{k} \tilde{c}_{\sigma(i-1)+1} \geq s$, entonces

para $1 \leq j \leq m_1$,

si $I \in T'_{\geq}(\mathfrak{c}^{(m_1+1,n)}, s - \tilde{c}_j)$ y su multiplicidad es μ ,

se tiene $[w_{1j} ; m_1] \cup I \in T'_{\geq}(\mathfrak{c}, s)$ y su multiplicidad

es $\binom{m_1}{w_{1j}} \mu$.

El conjunto de soluciones restantes es $T'_{\geq}(\mathfrak{c}^{(m_1+1,n)}, s)$

Prueba :

a) Evidente

b-1) Por el Teorema II.4.1, para $1 \leq t_i \leq r_i$, $i \in B$ se tiene $\left\{\sum_{h=1}^{i-1} m_h + t_i\right\} \in T_{\geq}(\tilde{\mathfrak{c}}, s)$. En consecuencia, por la Proposición II.5.4, $\left[w_{ir_i} ; \sum_{h=1}^{i} m_h\right] \in T'_{\geq}(\mathfrak{c}, s)$. Finalmente, la Proposición II.5.2 da directamente la multiplicidad.

b-2) Por el Teorema II.4.1, el subconjunto de $T_{\geq}(\tilde{\mathfrak{c}}, s)$ que queda por determinar es $T_{\geq}(\tilde{\mathfrak{d}}, s)$, donde $\tilde{\mathfrak{d}}$ se obtiene de $\tilde{\mathfrak{c}}$ suprimiendo, para cada $i \in B$, los elementos

$\tilde{c}_{\sigma(i-1)+1}$, $\tilde{c}_{\sigma(i-1)+2}$, ... , $\tilde{c}_{\sigma(i-1)+r_i}$. Como, por la Proposición II.5.4 cada solución de $T'_{\geqslant}(\mathbf{c}, s)$ está asociada a una y solo una solución de $T_{\geqslant}(\tilde{\mathbf{c}}, s)$, para determinar completamente $T'_{\geqslant}(\mathbf{c}, s)$ solo falta encontrar $T'_{\geqslant}(\mathbf{d}, s)$.

c-1) $\sum_{i=1}^{k} \tilde{c}_{\sigma(i-1)+1} = \sum_{i=1}^{k} m_i c_{\sigma(i)} = \sum_{i=1}^{n} c_i$. Si la primera de estas expresiones es menor que s, también lo será la últina y por el Teorema II.4.1, $T'_{\geqslant}(\mathbf{c}, s) = \emptyset$.

c-2) Es consecuencia directa de las Proposiciones II.5.4 y II.5.2.

c-3) Consecuencia de los Lemas II.5.1 y II.5.2 y de la Proposición II.5.2.

c-4) También consecuencia de los dos lemas precedentes y de la Proposición II.5.2.

c.q.d.

Así como el algoritmo HR se fundamenta en el Teorema II.4.1 también el teorema precedente sirve de base para desarrollar el algoritmo correspondiente, pero dadas las características constructivas de estos teoremas evitaremos en esta ocasión el escribir explícitamente dicho algoritmo.

A modo de conclusión del capítulo, el interés del Teorema II.5.1 estriba en dar un método para determinar $T_{\geqslant}(\mathbf{c}, s)$, que será tanto más eficiente que el algoritmo HR cuanto más elementos -- iguales haya en $\tilde{\mathbf{c}}$. Así, en el Ejemplo II.5.1, la obtención de I_1 es relativamente más eficiente que la de I_2 . Es evidente sin embar

go que las cotas encontradas en los Teoremas II.4.3 y II.4.4 siguen siendo válidas, y que en consecuencia será necesario utilizar el concepto de complejidad media para obtener resultados más significativos.

II.6 Demostraciones técnicas y cuestiones complementarias

A) Prueba de la cadena de reducciones $p_1 \propto p_2 \propto p_3 \propto p_4 \propto p_5$

$p_1 \propto p_2$:

Sea $F = \bigwedge_{i=1}^{n} C_i$, $C_i = \bigvee_{j=1}^{m_i} a_{ij}$, con

$$a_{ij} = \{x_1, \ldots, x_p, \bar{x}_1, \ldots, \bar{x}_p\}$$

una fórmula en forma normal conjuntiva para la que se quiere estudiar el problema de la satisfactibilidad.

Sea F' la fórmula obtenida al sustituir cada cláusula C_i de cuatro o más literales por

$$(a_1 \vee a_2 \vee y_1) \wedge (a_3 \vee \bar{y}_1 \vee y_2) \wedge (a_4 \vee \bar{y}_2 \vee y_3) \wedge \ldots$$

$$\ldots (a_{m_i-2} \vee \bar{y}_{m_i-4} \vee y_{m_i-3}) \wedge (a_{m_i-1} \vee a_{m_i} \vee \bar{y}_{m_i-3}),$$

donde los literales a_{ij} se han representado sin su primer sub-índice por comodidad, y las variables $y_1, y_2, \ldots$ son variables booleanas suplementarias.

Se comprueba ahora con facilidad que F' se satisface sii F se satisface.

$\underline{p_2 \propto p_3}$:

Sea $F = \bigwedge_{i=1}^{n} C_i$, $C_i = \bigvee_{j=1}^{t} a_{ij}$, con

$a_{ij} \in \{x_1, \ldots, x_p, \bar{x}_1, \ldots, \bar{x}_p\}$, $t \leq 3$.

Sea $G = (X, E)$ un grafo no orientado definido por

$$X = \{x_1, \ldots, x_p, \bar{x}_1, \ldots, \bar{x}_p\} \cup \{v_1, \ldots, v_p\} \cup \cup \{c_1, \ldots, c_n\}$$

$$E = \{(v_i, v_j) : i \neq j\} \cup \{(x_i, \bar{x}_i) : 1 \leq i \leq p\} \cup \cup \{(v_i, x_j), (v_i, \bar{x}_j) : i \neq j\} \cup \{(c_i, x_j) : : x_j \notin C_i\} \cup \{(c_i, \bar{x}_j) : \bar{x}_j \notin C_i\}$$

Finalmente, sea $k = p + 1$

Puede suponerse sin pérdida de generalidad que $p \geq 4$, - pues si no, el problema de satisfactibilidad propuesto sería decidible en tiempo polinomial.

Se tiene entonces :

a) El subgrafo de G engendrado por los vértices $\{v_1, \ldots, v_p\}$ es una clique y precisa por lo tanto de p colores.

b) Si $i \neq j$, los vértices v_i , x_j y $\bar{x}_j$ son adyacentes, lo que introduce un p + 1-ésimo color, que designaremos por Q. Con -- p+1 colores, o bien v_i y x_i son del mismo color y $\bar{x}_i$ es de color Q, o bien v_i y $\bar{x}_i$ son del mismo color y x_i es de color Q. Es claro entonces que p+1 colores bastan para los vértices $\{x_1, \ldots, x_p, \bar{x}_1, \ldots, \bar{x}_p\}$.

c) Todo vértice c_i es adyacente a $2p-3$ vértices, por lo menos. Ahora bien, $p \geq 4 \Longrightarrow 2p - 3 > p + 1$. Por lo tanto $\exists_j$, $1 \leq j \leq p$, tal que x_j y $\bar{x}_j$ figuran entre estos $2p - 3$ vértices. En consecuencia, c_i no puede ser de color Q pero sí de alguno de los otros.

De este modo, para todo i , $1 \leq i \leq n$; hay un literal $a_h \in C_i$ tal que el vértice a_h no es de color Q . En caso contrario, todos los $a_h \in C_i$ son de color Q. Pero entonces c_j es adyacente a p vértices de p colores distintos, así como por lo menos a un vértice de color Q, con lo que p + 1 colores no bastarían . Dando el valor 0 a todo vértice de color Q y 1 a los demás vértices, se concluye que F se satisface sii p + 1 colores bastan para G.

$P_3 \propto P_4$:

Sea $G = (X, E)$ un grafo no orientado, $k \in \mathbb{Z}^+$. Definimos los siguientes conjuntos:

$$S = X \cup \{(u, i) : u \in E , \quad 1 \leq i \leq k\}$$

$$S_{xi} = \{x\} \cup \{(u,i) : u \in E \text{ y } u \text{ incide en } x\},$$

$$x \in X, \quad 1 \leq i \leq k$$

$$R_{ui} = \{(u,i)\}, \quad u \in E, \quad 1 \leq i \leq k$$

Supongamos que k colores $c_1, c_2, \ldots, c_k$ bastan para G. Sean $x, y \in X$, $x \neq y$, y sean c_i, c_j sus colores respectivos. Entonces $S_{xc_i} \cap S_{yc_i} = \emptyset$; (si $(x,y) \notin E$, es evidente; si $(x,y) \in E$ entonces $c_i \neq c_j$ y el resultado es inmediato). Por otro lado, es claro que $S = \bigcup_{x,i} S_{xi} \cup \left(\bigcup_{\substack{u,i \\ (u,i) \notin S_{xi}}} R_{ui} \right)$.

Recíprocamente, supongamos que existe una subfamilia de $\{S_{xi}\} \cup \{R_{ui}\}$ que sea una partición de S. Entonces, para cada $x \in X$ existe un color c_i único tal que $x \in S_{xc_i}$, con $1 \leq i \leq k$. Si k colores no bastan para G, entonces existe $(x,y) \in E$ tal que x e y tienen el mismo color c_j. Pero en este caso, $((x,y),j) \in S_{xc_j} \cap S_{yc_j}$ y no puede haber ninguna partición de S entre los S_{xi} y R_{ui}. Por lo tanto, la partición -- existe sii G admite k colores.

$\underline{p_4 \propto p_5}$:

Sean $S = \{u_1, u_2, \ldots, u_p\}$

$$\{S_1, S_2, \ldots, S_k\} \subseteq \qquad (S)$$

y $t \geqslant 1$, dados.

Definimos :

$$\beta = k + 1 \quad ; \quad s = \frac{\beta^p - 1}{\beta - 1} \quad ; \quad \mathcal{E}_{ij} = \begin{cases} 1 & \text{si } u_i \in S_j \\ 0 & \text{si no} \end{cases}$$

$$\mathfrak{c} = (c_1, \ldots, c_k) \quad \text{con} \quad c_j = \sum_{i=1}^{p} \mathcal{E}_{ji} \beta^{i-1}$$

Probaremos que $I \in T_{=}(\mathfrak{c}, s)$ sii $\{S_i\}_{i \in I}$ es una partición de S.

En efecto, si $I \in T_{=}(\mathfrak{c}, s)$ entonces :

$$\sum_{j \in I} c_j = \sum_{j \in I} \sum_{i=1}^{p} \mathcal{E}_{ji} \beta^{i-1} = \sum_{i=1}^{p} \left(\sum_{j \in I} \mathcal{E}_{ji} \right) \beta^{i-1} =$$

$$= s = \frac{\beta^p - 1}{\beta - 1} = 1 + \beta + \beta^2 + \ldots + \beta^{p-1} .$$

Pero $\sum_{j \in I} \mathcal{E}_{ji} \leq k = \beta - 1$ y en consecuencia,

$$\sum_{j \in I} \mathcal{E}_{ji} = 1 \quad ; \quad i = 1, 2, \ldots, p .$$

Por lo tanto, para todo i en $\{1, 2, \ldots, p\}$ hay un y solo un $j \in I$ tal que $u_i \in S_j$. De lo que se desprende que $\{S_j\}_{j \in I}$

es una partición de S.

La demostración del recíproco es similar.

No es difícil ver que todas las construcciones utilizadas en las reducciones se llevan a cabo en tiempo polinomial. Además, - la pertenencia de los problemas p_i , $1 \leq i \leq 5$ a la clase NP es evidente, lo que completa la demostración.

c.q.d.

B) Prueba del Lema II.4.4

Toda solución I de $T_{\geq}$ puede representarse por un camino desde r a un vértice terminal de la estructura de árbol definida en (3.7), sección 3 de este capítulo. Por otro lado, dicha estructura es tal que todo sub-árbol T_i contiene como sub-árboles a T_{i+1}, T_{i+2}, ..., T_n . En consecuencia si un camino en T_{j_1} definido por r, c_{j_1}, c_{j_2}, ..., c_{j_p}, con $j_k < j_{k+1}$, representa una solución de $T_{\geq}$, entonces, $\forall\, k > 1$, el camino en T_{j_k} definido por r, c_{j_k}, $c_{j_{k+1}}$, ..., c_p no puede representar a solución alguna de $T_{\geq}$. Basta entonces encontrar el número máximo de caminos desde r a un vértice terminal de T que no aparezcan nunca como sub-caminos de algún camino, o equivalentemente, al número máximo de vértices - de T de igual rango ρ [5], que se alcanza precisamente para

$$\rho = \left\lfloor \frac{n}{2} \right\rfloor \text{ ó } \left\lceil \frac{n}{2} \right\rceil$$

c.q.d.

NOTAS A PIE DE PAGINA DEL CAPITULO II

(1) El problema de decidir si $T_{=}(\mathfrak{c}, s) = \emptyset$ corresponde en cierto modo al "problema de la mochila 0/1" , que en programación matemática se define en la forma que sigue:

$$\text{maximizar} \quad \sum_{i=1}^{n} b_i x_i$$

s, a

$$\sum_{i=1}^{n} c_i x_i \leq s$$

$$s, b_i, c_i, \in \mathbb{Z}^+ , \quad 1 \leq i \leq n$$

$$x_i \in \{0, 1\} , \quad 1 \leq i \leq n$$

Nuestro problema equivale al caso en que $b_i = c_i$ para $1 \leq i \leq n$. Es evidente que $T_{=}(\mathfrak{c}, s) \neq \emptyset$ sii $\max \sum_{i=1}^{n} c_i x_i = s$.

La determinación de $T_{=}(\mathfrak{c}, s)$ (o de uno de sus elementos) está muy relacionada también con el problema del establecimiento de particiones. (Ver por ejemplo E. Horowitz y S. Shani [1974]).

(2) Sea la inecuación $\mathfrak{a} \cdot \mathfrak{x} + \mathfrak{b} \cdot \bar{\mathfrak{x}} \geq s$; $\mathfrak{a}, \mathfrak{b} \in \mathbb{Z}^n$.
Definiendo

$$x'_i = \begin{cases} x_i & \text{si} \quad a_i \geq b_i \\ \bar{x}_i & \text{si} \quad a_i < b_i \end{cases} \quad ; \quad c_i = \begin{cases} a_i - b_i & \text{si} \quad a_i \geq b_i \\ b_i - a_i & \text{si} \quad a_i < b_i \end{cases}$$

$$; \quad r_i = \begin{cases} b_i & \text{si} \quad a_i \geq b_i \\ a_i & \text{si} \quad a_i < b_i \end{cases}$$

se obtiene la inecuación equivalente $\mathfrak{c} \cdot \mathfrak{x}' \geqslant s - \sum_{i=1}^{n} r_i$

(3) Hemos utilizado esta notación exclusivamente en los subíndices por razones de claridad.

(4) También es posible generalizar de forma similar el Lema II.4.2, pero no es necesario ahora. El enunciado correspondiente es :

$I \in T_{\geqslant}(\mathfrak{c}^{(\pi)}, s) \Longrightarrow I \in T_{\geqslant}(\mathfrak{c}, s)$ y la prueba es directa.

(5) Entenderemos por rango ρ de un vértice x de un árbol, al número de vértices del camino que une a x con la raiz del árbol.

C A P I T U L O III

COMPLEJIDAD ALGORITMICA Y ESTADISTICA.

III.1 Introducción

Es posible analizar algoritmos que resuelven algunos problemas estadísticos utilizando herramientas de complejidad combinatorial. Los problemas básicos de ordenamiento, entre otros, juegan un papel importante en problemas de inferencia no paramétrica, correlación y regresión (Shamos, [1976]). Pero este enfoque se hace insuficiente cuando la resolución del problema requiere un proceso iterativo -por ejemplo, la resolución numérica de un sistema de ecuaciones diferenciales en estimación paramétrica- y es preciso entonces recurrir a técnicas de complejidad analítica (ver sección 5). Por otro lado, la afirmación recíproca también es cierta, pues los métodos estadísticos son útiles para analizar el comportamiento de algoritmos, particularmente en el estudio de su complejidad media. En este sentido, con frecuencia se recurre a la experiencia práctica y a técnicas de simulación y solo en los últimos años se han publicado los primeros trabajos teóricos de cierta importancia; entre ellos citaremos la metodología general propuesta por Bentley (1981) y aplicada en particular al Problema del Viajante, la interesante discusión de técnicas útiles en este tipo de análisis que se encuentra en Lueker (1981), y la ya clásica referencia de Weide (1978), que estudia la aplicación de algunas de estas técnicas a diversos problemas concretos. En las secciones siguientes desarrollaremos diversos puntos de este área de creciente interés.

III.2 Medidas de centralidad

Tal vez uno de los aspectos primitivos más importantes en estadística es el estudio de valores centrales en torno a los cuales se encuentran distribuidas un conjunto de observaciones. Por esto dedicaremos primero nuestra atención, aunque someramente, a estas medidas.

El siguiente par de resultados es ampliamente conocido --- (Shamos [1975])

Sea $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$; entonces :

Teorema III.2.1.-

a) El tiempo necesario para calcular $\bar{x}$ es de $O(n)$.
b) El espacio necesario para calcular $\bar{x}$ es constante.

Prueba :

a) $\bar{x}$ se calcula mediante $n - 1$ sumas y 1 división
b) Basta actualizar las sumas parciales en una misma posición de - memoria.

c.q.d.

Corolario III.2.1.-

El problema de actualizar una media puede resolverse en - tiempo lineal y en espacio constante.

Prueba :

Sea $\bar{x}_{n_1}$ la media de $\left\{x_i\right\}_{i=1}^{n_1}$ y sean $\left\{x_i'\right\}_{i=1}^{n_2}$ nuevas observaciones. Entonces la media actualizada $\bar{x}_{n_1+n_2}$ puede calcularse mediante n_2+1 sumas, una multiplicación y una división si utilizamos la expresión:

$$\bar{x}_{n_1+n_2} = \frac{n_1 \bar{x}_{n_1} + \sum_{i=1}^{n_2} x_i'}{n_1 + n_2}$$

En cuanto al espacio, basta considerar una posición de memoria para el numerador y otra para el denominador.

c.q.d.

Corolario III.2.2.-

El problema de actualización continua de la media puede resolverse en tiempo lineal (en el número de observaciones) y en espacio constante.

Prueba :

Es este caso, es necesario calcular la sucesión $\left\{\bar{x}_k\right\}_{k \geq 1}$, cuyos elementos se definen mediante la relación de recurrencia :

$$\bar{x}_{k+1} = \frac{k \bar{x}_k + x_{k+1}}{k + 1}$$

Para $k = n$ dado, bastan entonces $2(n-1)$ sumas, $n-1$ multiplicaciones y $n-1$ divisiones. Por otro lado, para cada k, es suficiente un espacio constante (Corolario anterior). La continuación es evidente.

c.q.d.

En lo que respecta a la moda de un conjunto de observaciones, podemos afirmar lo siguiente:

Teorema III.2.2.-

La(s) moda(s) de $\{x_i\}_{i=1}^{n}$ puede(n) encontrarse mediante $O(n \log n)$ operaciones elementales (sumas y comparaciones).

Prueba :

Consideremos el algoritmo definido por:

1) Ordenar totalmente $\{x_i\}_{i=1}^{n}$ de mayor a menor

2) Recorrer $\{x_i\}_{i=1}^{n}$ de mayor a menor y calcular las frecuencias de los elementos distintos. Sea $\{x_i', f_i\}_{i=1}^{k}$ el resultado obtenido.

3) Determinar el(los) elemento(s) de frecuencia máxima.

El primer paso requiere $O(n \log n)$ comparaciones (ver Capítulo I).

El segundo paso necesita $n-1$ comparaciones y, a lo sumo, $n-1$ sumas (si $x_i \neq x_j$ para $i \neq j$). Finalmente bastan $n-1$ comparaciones para el tercer paso. El resto de la prueba es trivial.

c.q.d.

Finalmente, en cuanto al cálculo de la mediana y de percentiles, nos limitaresmo a citar los siguiente resultados : Shamos (1976) demuestra que la mediana de n observaciones puede encontrarse mediante $O(n \log n)$ operaciones elementales. Munro y Paterson - (1980) y Dobkin y Munro (1981) obtienen una cota superior lineal para este problema y prueban que el espacio necesario no puede ser inferior a $\left\lceil \frac{n}{2} \right\rceil + 1$. Además llegan al sorprendente resultado de la - inexistencia de un algoritmo con las anteriores caracteristicas para el cálculo de los restantes percentiles.

III.3 Distribuciones de permutaciones

Una gran variedad de problemas requieren el conocimiento de distribuciones de permutaciones. Consideremos por ejemplo el problema siguiente. Dada una población de tamaño n, determinar la proporción de muestras de tamaño k ($k \leq n$) cuya media sea superior o - igual a la media de una muestra dada. Si $A = \{a_1, a_2 \ldots, a_n\}$ es la población en cuestión, este problema puede ser resuelto encontrando la distribución de

$$\sum_{j=1}^{n} a_j x_j \qquad (3.1)$$

donde las x_j son variables aleatorias binarias tales que $\sum_{j=1}^{n} x_j = k$ y tales que cada muestra de tamaño k sea igualmente probable.

Un algoritmo puramente enumerativo requeriría $O\left(\frac{2^n}{\sqrt{n}}\right)$ operaciones elementales en el peor de los casos (que corresponde a $k = \left\lfloor \frac{n}{2} \right\rfloor$ ó $\left\lceil \frac{n}{2} \right\rceil$). ¿Es posible encontrar un algoritmo polinomial pa

ra este problema?. Pagano (1981) sostiene que un problema más complicado que éste es polinomial : determinar la distribución de $\sum_{j=1}^{n} f(a_j)\, x_j$, donde f es alguna función calculable en tiempo polinomial, de los elementos de la población. En su demostración, propone un algoritmo basado en el cálculo de la función característica de $\sum_{j=1}^{n} f(a_j)\, x_j$ en puntos equidistantes sobre $[0, 2\pi]$.

Este resultado es de importancia práctica en la utilización de diversos tests que se basan en estadísticos de la forma (3.1).

Por otro lado, esta metodología sugiere otro enfoque para tratar el problema definido en el Capítulo II. Basta observar que el problema allí definido es equivalente a n problemas del tipo definido en (3.1) con la restricción suplementaria de que para $k = 1, 2, \ldots, n$ toda submuestra de una muestra de tamaño k y de media no inferior a s dado, tenga media inferior a s. Queda así abierta la posibilidad de comparar la complejidad media de un algoritmo basado en esta idea, con la complejidad media del algoritmo basado en el Teorema II.5.1.

III.4 Estimación por punto

Dos cuestiones distintas aparecen involucradas en este tema. Por un lado, el problema de obtener una estimación de un parámetro θ mediante un estimador T dado . La complejidad computacional de este problema depende del estadístico que define a T (ver

los ejemplos estudiados en la sección 2 de este capítulo). Esto sugiere inmediatamente añadir una propiedad más al conjunto de propiedades deseables para T. Con este fin, proponemos la siguiente

Definición III.4.1.-

Sea T un estimador de θ que posea una propiedad P, y sea $O(f(n))$ su complejidad. Diremos que T es un P-estimador de θ de complejidad mínima si para todo estimador T' que posea la propiedad P y que tenga complejidad $O(g(n))$ se tiene $O(f(n)) \leqslant O(g(n))$.

La definición anterior supone conocidas cotas superiores estrictas del tiempo necesario para calcular T y T', así como siendos algoritmos capaces de alcanzar dichas cotas. En el caso en que dichas cotas fuesen desconocidas, o bien se desconocería también el P-estimador de complejidad mínima, o bien no se sabría identificar un P-estimador conocido como de complejidad mínima.

Cabe mencionar aquí que si bien en complejidad algorítmica se habla a veces también de "eficiencia" en el sentido del inverso de la complejidad (Kronsjö [1979]) y de algoritmo "optimal" en el sentido de algoritmo de complejidad mínima, no es conveniente introducir dichos términos en la definición anterior por tener un sentido propio en Inferencia Estadística.

El segundo problema que contemplaremos en estimación paramétrica es el de la obtención de una estimación de θ cuando no se conoce una expresión analítica para un estimador T, pero si se conocen ecuaciones obtenidas a partir de alguno de los métodos clási-

cos (máxima verosimilitud, momentos, etc.) . Muy poco se conoce sobre la dificultad computacional de estos problemas. Consideremos, para fijar ideas, el método de máxima verosimilitud. Si deseamos estimar un parámetro único θ, este método se reduce a resolver la ecuación

$$\frac{\partial \log L}{\partial \theta} = 0 \qquad (4.1)$$

donde $L = L(x_1, x_2, \ldots, x_n, \theta) = \prod_{i=1}^{n} f(x_i, \theta)$

$f(x, \theta)$ es la función de densidad de la población en estudio y $\{x_i\}_{i=1}^{n}$ es una muestra de la misma población.

Salvo en casos sencillos, en que la ecuación (4.1) tiene solución analítica, deberemos recurrir a métodos numéricos de resolución de ecuaciones. En general estos métodos son iterativos, y el número de iteraciones realizadas depende del orden de convergencia del método y de la precisión deseada en el resultado. Para evaluar la complejidad computacional de estos problemas es necesario tener en cuenta, entonces, los factores siguientes

1) Complejidad de cada iteración, medida en función del número de evaluaciones de funciones, de derivadas de dichas funciones, y el número de operaciones elementales entre estas funciones.

2) El orden de convergencia.

3) El error máximo permitido.

Kronsjö (1979) discute y compara diversas medidas de complejidad ba

sadas en estos factores, desde un punto de vista general. Hasta el presente, no conocemos ningún trabajo que desarrolle estas ideas - en problemas de estimación, por lo que se abre aquí una importante vía de investigación. (En cambio, existen estudios de simulación - para comparar experimentalmente la complejidad de diversos métodos para algunos problemas. Mencionaremos, por ejemplo, el trabajo de Zanakis (1977) sobre la obtención de estimaciones de máxima verosimilitud para los tres parámetros de la distribución de Weibull).

III.5 Análisis de algoritmos para selección de muestras

Consideremos el problema de obtención de una muestra de - tamaño m, de una población de tamaño n distribuida uniformemente. - Una característica deseable para esta muestra es que sea insesgada, en el sentido de que cada elemento de la población tenga la misma - probabilidad de ser seleccionado.

El algoritmo más intuitivo para este problema es, posiblemente, el siguiente:

Algoritmo M_1 :

Examina sucesivamente los elementos de la población, y los acepta con probabilidad $\frac{m}{n}$.

1.- Para cada $i \in \{1, \ldots, n\}$, asociar al elemento i-ésimo de la población de variable aleatoria x_i definida por

$$x_i = \begin{cases} 1 & \text{con probabilidad } \frac{m}{n} \\ 0 & \text{con probabilidad } 1 - \frac{m}{n} \end{cases}$$

2.- Para cada $i \in \{1, \ldots, n\}$:

Generar un valor de x_i

Aceptar el elemento i-ésimo si $x_i = 1$

3.- FIN

Teorema III.5.1.-

La muestra obtenida por el algoritmo M_1 es insesgada pero no es de tamaño fijo

Prueba :

Si llamamos X a la v.a. "número de elementos de la muestra", se tiene trivialmente que $E(X) = \sum_{i=1}^{n} E(x_i) = m$, y

$$\text{Var}(X) = \sum_{i=1}^{n} \text{Var}(x_i) = m\left(1 - \frac{m}{n}\right)$$

c.q.d.

Obsérvese que la varianza es pequeña si $m \approx n$ pero si $m \ll n$, el tamaño verdadero de la muestra obtenida puede diferir mucho de n.

Teorema III.5.2.-

La complejidad del algoritmo M_1 es de $O(n)$.

Prueba :

Trivial, pues se generan n valores

c.q.d.

Para eliminar el problema de la gran varianza del algoritmo M_1, puede utilizarse al algoritmo M_2 descrito a continuación :

Algoritmo M_2 :

Genera aleatoriamente elementos de la población hasta - aceptar exactamente m.

Utiliza las variables :

k : número de elementos seleccionados

MUESTRA : vector de elementos seleccionados; en particular, MUESTRA (i) será el i-ésimo elemento seleccionado.

C : vector característico del conjunto de elementos seleccionados. (Supondremos numerados de 1 a n los elementos de la población).

1.- [Preparación]

k ⟵ 0

C (i) ⟵ 0 para todo i ∈ {1, ..., n}

2.- [Iterar]

Mientras k < m :

2.1.- [Generar un elemento]

Generar un valor de i de una v.a. distribuída uniformemente sobre {1, 2, ..., n}

2.2.- [Aceptar un elemento]

Si C(i) = 0 entonces : k ⟵ k + 1

MUESTRA (k) ⟵ i

C(i) ⟵ 1

3.- FIN

Es evidente que la muestra obtenida mediante el algoritmo M_2 tiene exactamente m elementos y es insesgada. Pero en cambio, la complejidad de este algoritmo es superior a la de M_1, como se verá a continuación.

Sea p_k la probabilidad de generar un entero i aún no seleccionado cuando ya se han seleccionado k elementos de la población. Entonces, $p_k = \frac{n - k}{n}$, $k = 0, 1, \ldots$

Sea x_k una variable aleatoria que representa el número de enteros generados, antes de obtener uno que no esté entre los k previamente seleccionados.

Proposición III.5.1.-

x_k tiene una distribución geométrica de media $\frac{1}{p_k}$, y varianza $\frac{1 - p_k}{p_k^2}$.

Prueba :

Evidente .

c.q.d.

La proposición anterior permite estudiar la complejidad del algoritmo M_2 basándose en el número medio de enteros generados :

Teorema III.5.3.-

El número medio de enteros generados para obtener una muestra de tamaño m mediante el algoritmo M_2 está acotado superiormente por $O(n \log n)$ e inferiormente por $O(m)$.

Prueba :

Sea t el número total de enteros generados por el algoritmo M_2. Es claro que $t = \sum_{k=0}^{m-1} x_k$.

En consecuencia,

$$E(t) = \sum_{k=0}^{m-1} E(x_k) = \sum_{k=0}^{m-1} \frac{1}{p_k} = \sum_{k=0}^{m-1} \frac{n}{n-k} = n\left(\sum_{k=n-(m-1)}^{n} \frac{1}{k}\right) =$$

$$= n\,(H_n - H_{n-m+1}) \qquad (5.1)$$

donde H_n es el n-ésimo número armónico. Esta expresión es válida para $m = 1, 2, \ldots, n-1$. (Para $m = 0$ y $m = n$ se obtienen muestras "triviales" para las que el algoritmo M_2 no es aplicable), y es evidente que es monótona creciente en m. Por lo tanto basta analizar los casos en que $m = O(n)$, y en que $m << n$.

a) $m = O(n)$.

En este caso, $E(t) = O(n\,H_n)$ (5.2)

Sea $U_n = \sum_{k=1}^{n} \frac{1}{2^{\lceil \log_2 k \rceil}}$

$$V_n = \sum_{k=1}^{n} \frac{1}{2^{\lfloor \log_2 k \rfloor}}$$

Es claro que $U_n \leq H_n \leq V_n$, y que estas desigualdades son estrictas para $n > 2$.

Por otro lado, podemos escribir

$$U_n = 1 + \frac{p}{2} + r_n$$

$$V_n = q + s_n$$

donde $p = \lfloor \log_2 n \rfloor$, $q = \lfloor \log_2 (n+1) \rfloor$,

$$r_n = \frac{n - 2^p}{2^{p+1}} \quad , \quad s_n = \frac{n + 1 - 2^q}{2^{q+1}}$$

obsérvese que $0 \leq r_n < \frac{1}{2}$ y $0 \leq s_n < 1$.

Por lo tanto :

$$\lfloor \log_2 n \rfloor < H_n < \lfloor \log_2 (n+1) \rfloor$$

y $H_n = O(\log_2 n)$, lo que nos permite reescribir (5.1) en la forma :

$$E(t) = O(n \log_2 n)$$

b) $m << n$

Entonces , $E(t) = n \sum_{k=0}^{m-1} \frac{1}{n-k} = n\, O(\frac{m}{n}) = O(m)$

c.q.d.

Kennedy y Gentle (1980) analizan un método no secuencial para este problema y estudian también el caso en que se desconoce el tamaño de la población.

III.6 Otros problemas

Los problemas estudiados en las secciones anteriores no son en modo alguno los únicos que han empezado a estudiarse desde el punto de vista de su complejidad computacional. Así por ejemplo, Kunth y Yao (1976) y Goodman y Hedetniemi (1977) analizan métodos para generar números aleatorios y Shamos (1976) estudia la complejidad de algunos algoritmos para diversos tipos de regresión. Sin embargo el número de trabajos publicados en esta área hasta la fecha es pequeño. Pensamos que, debido al auge que ha experimentado últimamente el estudio de técnicas de análisis de algoritmos, y a la necesidad de disponer de métodos estadísticos computacionalmente eficientes, la situación cambiará drásticamente en los próximos años.

BIBLIOGRAFIA

ADELMAN, L. and MANDERS, K. (1977) "Reducibility, Randomness and Intractability" 9th. An. ACM Symp. on Theory of Computing. Boulder, Colo., 151-153.

AHO, A.V.; HOPCROFT, J.E. and ULLMAN, J.D. (1974) "The Design and Analysis of Computer Algorithms" Addison-Wesley.

BENTLEY, J.L. (1981). "Applications of Statistics to Applied Algorithm Design". Computer Science and Statistics: Proc. 13th. Symp. on the Interface, Pittsburgh, Pa., 59-67.

BENZAKEN, C. and HAMMER, P.L. (1977) "Linear Separation of Dominating Sets in Graphs" Rap. Rech. 72, Univ. Scientifique et Medicale de Grenoble.

BLUM, M. (1967) "A Machine-Independent Theory of the Complexity of the Recursive Functions" J. Assoc. Comp. Mach., 14, 322-336.

BORODIN, A. (1972) "Computational Complexity and the Existence of Complexity Gaps" J. Assoc. Comp. Mach., 19, 158-174.

CHURCH, A. (1936) "An Unsolvable Problem of Elementary Number Theory" Am. J. Math, 58, 345-363.

CHVATAL, V. and HAMMER, P.L. (1973) "Set Packing Problems and Threshold Graphs" Res. Rep. Corr. 73-21. Univ. of Waterloo.

COOK, S.A. (1971) "The Complexity of Theorem Proving Procedures" Proc. 3rd. Ann. ACM Symp. on Theory of Computing, 151-158.

DODGSON, C.L. (1883) in St. James's Gazette, Aug. 1, 5-6.

DOBKIN, D. and MUNRO, J.I. (1981) "Optimal Time Minimal Space Selection Algorithms" J. Assoc. Comp. Mach., 28, 454-461.

EDMONDS, I. (1965) "Paths, Trees and Flowers", Canad. J. Math., 17, 449-467.

FINE, T.L. (1973) "Computational Complexity, Random Sequences and Probability", chapter V of "Theories of Probability: An Examination of Fonndations", Academic Press.

FORD, L. and JOHNSON, S. (1959) "A Tournament Problem", Am. Math. Monthly, 66, 387-389.

GAREY, M.R. and JOHNSON, D.S. (1979) "Computers and Intractability: A Guide to the Theory of NP-Completeness", W.H. Freeman an a Co., San Francisco.

GILL, I. (1977) "Computational Complexity of Probabilistic Turing Machines" SIAM J. Comp., 6, 675-695.

GOODMAN, S.E. and HEDETNEMI, S.T. (1977) "Introduction to the Design and Analysis of Algorithms", Mc Graw Hill.

HADIAN, A. (1969) Ph. D. Thesis, Univ. of Minnesota.

HAMMER, P.L. et RUDEANU, S. (1970) "Méthodes Booléennes en R.O." Dunod.

HOPCROFT, J.E. and ULLMAN, J. (1979) "Introduction to Automata Theory, Languages and Computation" Addison-Wesley.

HOROWITZ, E. and SAHNI, S. (1974) "Computing Partitions with Applications to the Knapsack Problem" J. Assoc. Comp. Mach., 21, 277-192.

HU, Sze-Tsen (1965) "Threshold Logic" University of California Press.

KARP, R.M. (1972) "Reducibility Among Combinatorial Problems" in Complexity of Computer Computations Miller and Thatcher, eds.; Plenum Press, 85-104.

KENNEDY, W.I. and GENTLE, J.E. (1980) "Statistical Computing" Marcel Dekker.

KLEENE, S.C. (1936) "General Recursive Functions of Natural Numbers" Math. Ann., 112, 727-742.

KLEE, V., and MINTY, G.J. (1972) "How Good is the Simplex Algorithm?" in Inequalities III, ed. O. Shisha, Academic Press, 159-175.

KHACHIYAN, L.G. (1979) "A Polinomial Algorithm for Linear Programming". Dokl. Akad. Nauk. SSSR 244, 1093-1096.

KNUTH, D.E. (1973) "The Art of Computer Programming" vol. III: Sorting and Searching, Addison-Wesley.

---------- and Yao, A.C. (1976) "The Complexity of Nonuniform Random Number Generation" in Algorithms and Complexity:

New Directions and Recent Results, J.F. Traub, ed. Acad. Press N.Y.

KRONSJO, L.I. (1979) "Algorithms: Their Complexity and Efficiency", J. Wiley and Sons.

LOUIT, G. (1971) "Algorithmes de Tri" Dunod.

LOVASZ, L. (1980) "A New Linear Programming Algorithm-better or worse than the Simplex Method?". Math. Int., 141-146.

LUEKER, G.S. (1981) "Algorithms with Random Input" Computer Science and Statistics: Proc. 13th. Symp. on the Interface, Pittsburgh, Pa., 68-75.

MUNRO, J.I. and PATERSON, M. (1980) "Selection with Limited Storage" Theor. Comp. Sci. 12, 315-323.

MUROGA, S. (1971) "Threshold Logic and its Applications" Wiley-Interscience.

PAGANO, M. (1981) "Polynomial Time Algorithms for Obtaining Permutation Distributions". Computer Science and Statistics: Proc. 13th. Symp. on the Interface, Pittsburgh, Pa., 45-48.

RABIN, M.O. (1976) "Probabilistic Algorithms" in Algorithms and Complexity: New Directions and Recent Results, J.F. Traub ed., Acad. Press N.Y. 21-39.

---------- (1980) "Probabilistic Algorithms for Testing Primality" J. Number Th. 12, 128-138.

RACKOFF, C. (1982) "Relativized Questions involving Probabilistic Algorithms" J. Assoc. Comp. Mach. 29, 261-268.

REINGOLD, E. (1972) "On the Optimality of Some Set Algorithms", J. Assoc. Comp. Mach., 19, 649-658.

SAVAGE, J.E. (1976) "The Complexity of Computing". J. Wiley.

SHAMOS, I.M. (1976) "Geometry and Statistics: Problems at the Interface", in Algorithms and Complexity: New Directions and Recent Results, J.F. Traub, ed., Acad, Press N.Y., 251-279.

TURING, A.M. (1937) "On Computable Numbers, with an Application to the Ents heidungsproblem" Proc. London math. Soc. (2), 42, 230-265:

---------- (1937) "On Computable Numbers, with an Application to the Enrs heidungsproblem". A Correction". Proc. London math. Soc. (2), 43, 544-546.

VALDES, T. y AZORIN, E. (1980) "Bases para un Enfoque de la Inferencia Secuencial bajo la Teoría de Autómatas" XII Reunión Nacional de Estadística, Investigación Operativa e Informática,Jaca (manuscrito no publicado).

WEIDE, B.W. (1978) "Statistical Methods in Algorithm Design and Analysis" Carnegie-Mellon University Computer Science Report CMU-CS-78-142.

ZANAKIS, S.H. (1977) "Computational Experience with Some Nonlinear Optimatization Algorithms in Deriving Maximum Likelihood Estimates for the Three-Parameter Weibull Distribution", in Algorithms Methods in Probability, M.F. Neuts, ed.; TIMS Studies in the Management Sciences, 7, 63-77.