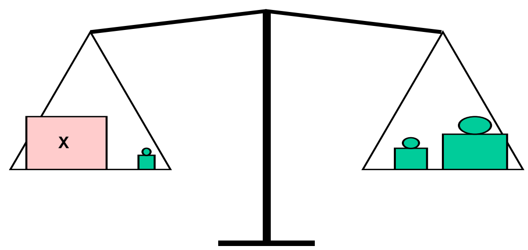
ISIS 211 Diseño de Algoritmos
Semestre 2002-2 - Parcial 3
Noviembre 21, 2002
Prof. Rodrigo Cardoso

Supóngase un conjunto P de n pesas, n≥0:

$$P = \{p_0, \ldots, p_{n-1}\}$$

tal que, para $i \in 0..n-1$, $p_i > 0$, $p_i \in$ **nat**.

Se dispone de una balanza de dos platos, en cada uno de los cuales pueden situarse algunas de las pesas de P, así como un peso desconocido X∈ **nat**, llamado el *objetivo*. Una situación típica tiene la forma



Cuando la balanza está equilibrada, el peso del objetivo es igual a la suma de los valores de las pesas que están en el plato opuesto menos la suma de los valores de las pesas que están en el mismo plato. El peso X es P-*pesable* si existe una manera de colocar el peso X y algunas de las pesas de P en la balanza, de manera que la balanza quede equilibrada. Hay dos preguntas para resolver, dados P y X:

a ¿Es X P-pesable?
b Si X es P-pesable ¿cuánto pesa X?

Se quiere desarrollar un algoritmo de agenda para resolver los problemas. Conviene tener una representación para lo que es un *ensayo de pesada* (escoger algunas de las pesas y ponerlas en los platos de la balanza), fácil de manipular. Con un arreglo:

`e : `**`array`**`[0..n-1] `**`of`**` {0,1,-1}`

se representa un ensayo de pesada. Se supone que el peso X se coloca en el plato izquierdo de la balanza y, para $0 \le i < n$:

- `e[i] =  0` : la pesa `i` no se coloca en ningún plato de la balanza;
- `e[i] =  1` : la pesa `i` se coloca en el plato izquierdo de la balanza;
- `e[i] = -1` : la pesa `i` se coloca en el plato derecho de la balanza.

1 **`[35/100]`** Exprese los diferentes elementos de una solución del problema **a** con algoritmo de agenda (`SOLPOS`, `sat`, ...) utilizando la notación anterior. Indique cómo puede, al mismo tiempo, resolver el problema **b**.

2 **`[10/100]`** Argumente si su algoritmo amerita o no:
- **2a** Manejo de nodos marcados
- **2b** Predicado dominó

3 **`[10/100]`** Estime, en términos de n, el orden de complejidad de la verificación del predicado de satisfacción.

4 **`[10/100]`** Estime, en términos de n, el orden de complejidad del paso

`AGENDA:=AGENDA ∪ SUC(x)`

5 **[15/100]** Si es posible, estime, en términos de n, el orden de complejidad de su algoritmo.

---

6 **[20/100]** Para cada una de las siguientes afirmaciones, juzgue la veracidad de la misma, y explique su respuesta:

6a [5/10] El problema de saber si un mapa se colorea con 5 colores está en P.
6b [5/10] Si un problema está en NP, entonces no puede estar en P.
6c [10/10] El problema de saber si un grafo de n vértices es conexo está en NP.

TAREA FINAL: Para entregar (impreso) el 3 de diciembre, 9 a.m., en Secretaría:

Escriba un artículo sobre el tema ***Búsqueda en grafos, como técnica de solución de problemas***. Escriba sus opiniones sobre:

- Para qué clase de problemas cree Usted que puede usarse esta técnica.
- Por qué, teniendo algoritmos rápidos para buscar, v.gr., Dijkstra, sigue siendo interesante estudiar búsquedas más generales (Algoritmos B, A, A*, heurísticas, ...).
- La dificultad de la búsqueda en grafos, i.e., ¿es un problema difícil? (... ¿está en P? ¿en NP?)

Algunas condiciones de forma:

Estructura:

Título
Autores
    Grupo de hasta 3 personas
Resumen
    Presentación resumida de la problemática y de los resultados
Introducción
    Presentación del tema
Desarrollo
    Secciones que desarrollan las ideas del artículo
Conclusiones
Bibliografía
    Fuentes consultadas.

Longitud

Entre 6 y 10 páginas, tamaño carta. Hojas numeradas en pie de página

Fuente

Arial 10, Interlineado sencillo.

Márgenes

Inferior	: 2.5 cm
Superior	: 2.5 cm
Derecho	: 2.1 cm
Izquierdo	: 2.5 cm

ISIS 211 Diseño de Algoritmos
Semestre 2002-2 - Parcial 3
Noviembre 21, 2002
Prof. Rodrigo Cardoso

---

1 ***[35/100]*** *Exprese los diferentes elementos de una solución del problema* ***a*** *con algoritmo de agenda (SOLPOS, sat, …) utilizando la notación anterior. Indique cómo puede, al mismo tiempo, resolver el problema* ***b****.*

```
SOLPOS = {e | e : array[0..n-1] of {0,1,-1}}                [ 5/10]
sat e  ≡ (+i| 0≤i<n : e[i]*pᵢ) = X                           [ 3/10]
SOL    = {e:SOLPOS | (+i| 0≤i<n : e[i]*pᵢ) = X}              [ 2/10]
```

El espacio de búsqueda es el mismo SOLPOS:
BUSQ = SOLPOS. [5/10]

La relación de sucesión entre nodos del espacio de búsqueda se puede definir así:
e → e' , si e' es sucesor en orden lexicográfico de e [15/10]
(considerando a e como una secuencia en $\{0,1,-1\}^n$).

En esta solución cada elemento tiene exactamente un sucesor. La agenda se reduce a un elemento.

La búsqueda empieza en:
s = (0,...,0). [2/30]

Al encontrar una solución e, el peso X es igual a $(+i|\ 0\le i<n : e[i]*p_i)$
[3/30]

Variantes:
Cualquier enumeración de $\{0,1,-1\}^n$ logra el mismo resultado (agenda de un elemento).
Ejemplos:

- Pensando que hay 3 posibilidades para colocar cada pesa, las pesadas de e se pueden representar con un número en base 3. Por ejemplo, 102_3 puede representar a e= (0,-1,1). Estra correspondencia es biunívoca, de manera que uno puede contar en base 3, decodificar el número a una secuencia que representa una pesada y, de esta manera, pasar por todo el conjunto de pesadas.

- Continuando con la idea anterior, se puede considerar a 0,1,-1 como dígitos de un sistema numérico. Como notación, $\underline{1}$ es el dígito para el −1. Un arreglo e representa el número
$$ve = (+i|\ 0\le i<n : e[i]*3^i)$$
Por ejemplo $0\underline{1}1$ representa el número 0*27 - 1*9 + 1*1 = -8.

  Entonces, se empieza en e = $\underline{1...1}$ y se cuenta en este sistema numérico.

Otras posibilidades (agenda de más de un elemento):

- Enumerar primero las pesadas con 0 pesas, después las de 1 pesa, ..., después las de n pesas. Y ordenar las pesadas de k pesas decidiendo cuáles de las k van a la izaquierda y cuáles van a la derecha.
  Da un grafo de ramificación finita, algo difícil de seguir (para calcular SUC).

- etc. En todo caso, se debe garantizar que se recorre todo BUSQ

[15/10]*

2 ***[10/100]*** *Argumente si su algoritmo amerita o no:*

2a *Manejo de nodos marcados*

No es necesario: nunca se repite un e en la enumeración. [5/10]

Cuando se usa una enumeración como base para la relación del grafo (el cálculo de sucesores), la respuesta es la misma: no hay que marcar, porque no hay repeticiones.

Más generalmente: sólo hay que marcar los nodos visitados si $.\!\rightarrow\!.$ es una relación en la que puede haber ciclos.

2b *Predicado dominó*

En el caso de la enumeración, un predicado dominó eliminaría pesadas de un punto en adelante de la enumeración. Para la solución planteada en **1** no se ve, a priori, un predicado dominó que funcione. [5/10]

3 ***[10/100]*** *Estime, en términos de n, el orden de complejidad de la verificación del predicado de satisfacción.*

`O(sat(e)) = O(n)` [10/10]

4 ***[10/100]*** *Estime, en términos de n, el orden de complejidad del paso*

AGENDA:=AGENDA $\cup$ *SUC(x)*

El algoritmo corresponde a calcular el sucesor de número natural, en notación ternaria. En el peor caso, deben cambiarse n dígitos. Así las cosas, la sucesión es $O(n)$. No se descarta que haya enumeraciones más baratas (como, por ejemplo, usar el código Gray en la notación binaria cambia un bit entre dos números consecutivos). [10/10]

5 ***[15/100]*** *Si es posible, estime, en términos de n, el orden de complejidad de su algoritmo.*

El algoritmo itera sobre $\{-1,0,1\}^n$, efectuando cada vez $O(n)$ operaciones (verificación de `sat` y cálculo de sucesores).

En total $O(n3^{n-1})$. O bien: $O(n3^n)$. [15/10]

6 ***[20/100]*** *Para cada una de las siguientes afirmaciones, juzgue la veracidad de la misma, y explique su respuesta:*

6a **[5/10]** *El problema de saber si un mapa se colorea con 5 colores está en P.*

Verdadero. [5/10]

4 colores bastan: la respuesta es "sí", calculada en $O(1)$.

6b **[5/10]** *Si un problema está en NP, entonces no puede estar en P.*

Falso. [5/10]

$P \subseteq NP$. Todo problema que se sepa que está en P está también en NP.

6c **[10/10]** *El problema de saber si un grafo de n vértices es conexo está en NP.*

Verdadero. [10/10]

Con Floyd-Warshall, por ejemplo, se decide en $O(n^3)$.