21-211 Diseño de Algoritmos
Semestre 2002-2 - Parcial 2
Noviembre 7, 2002
Prof. Rodrigo Cardoso

---

1 [60/100] *Cantidad no representable con n monedas*

En Estados Unidos hay monedas de 1,5,10,25,50 y 100 centavos. Es posible representar un dólar - 100 centavos - con 1 moneda (la de 100), con 2 monedas (2 de 50), con 3 monedas (1 de 50, 2 de 25), etc. Nótese que *no es posible* representar un dólar con 101 monedas. Se quiere determinar n, el mínimo número de monedas con el que *no se puede* representar un dólar.

Más generalmente, suponga que hay monedas de denominaciones $d_1<d_2, \ldots <d_m$, con $d_k \in$ **nat⁺**. Dada una cantidad C, desarrolle y analice un algoritmo de programación dinámica para determinar el mínimo n para el que C *no sea representable*.

a [10/50] Defina un lenguaje que permita formalizar la respuesta del problema como un cálculo que deba realizarse.

b [10/50] Establezca una recurrencia para efectuar el cálculo requerido.

c [10/50] Determine un diagrama de necesidades correspondiente.

d [10/50] Diseñe un invariante y un orden de evaluación para efectuar el cálculo.

e [10/50] Estime la complejidad temporal y espacial de su algoritmo.

f [10/50] Escriba el algoritmo.

---

2 [50/100] *Frascos y medidas*

Suponga tres frascos A, B, C con capacidades a, b y c litros, respectivamente. Se tiene que a = b+c. El frasco A está lleno de vino, mientras que B y C están vacíos. Se quiere desarrollar y analizar un algoritmo para trasvasar el vino entre los frascos, de modo que se deje la mitad en el frasco A y la otra mitad repartida entre B y C.

Del vaso X al vaso Y la operación de transvasar puede tener dos resultados:
- el vaso X se vacía y el contenido que estaba allí se añadió al vaso Y, o bien,
- el vaso Y se llena y en el vaso X queda el remanente que no se pudo transvasar.

a [20/50] Represente como un problema de grafos el problema anterior.

b [30/50] Explique un método para resolver el problema de grafos correspondiente y estime las complejidades espacial y temporal de su solución. No es necesario escribir el algoritmo.

21-211 Diseño de Algoritmos
Semestre 2002-2 - Parcial 2
Noviembre 7, 2002
Prof. Rodrigo Cardoso

---

1 **[60/100]**

a *Lenguaje:* **[10/10]**

Sea

`α(n,k) = true   ≡  "k es representable con n monedas"` [7/10]

Se quiere calcular

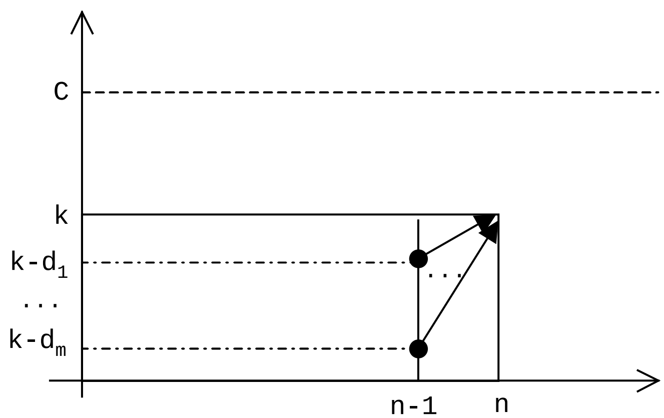
$$N = (\min n \mid : \neg\alpha(n,C)).$$

[3/10]

Recurrencia: **[10/10]**

```
α(n,k)    = true                              , si n=0, k=0
          = false                             , si n=0, 0<k≤C
          = (∃i| 1≤i≤m ∧ d_i≤k : α(n-1,k-d_i))  , si 0<n, 0<k≤C
```

Diagrama de necesidades: **[10/10]**



Invariante:

[10/10]

Se utiliza un vector `alfa[0..C]` **`of bool`**, así:

```
Inv:  0≤n ∧ 0≤k≤C ∧ alfa[0..k]=α(n-1,0..k) ∧ alfa[k+1..C]=α(n,k+1..C) ∧
      (∀n1| 0≤n1<n : α(n1,C))
```

El algoritmo recorre de izquierda a derecha y de abaja arriba la franja del plano descrita en el diagrama de recurrencia, buscando el primer n para el que `α(n,C)` sea falso.

El proceso se detiene porque `α(C+1,C)` es falso.

Complejidad: **[10/10]**

`T(n,m,C) = O(mC²)` : cuando hay que recorrer toda la franja hasta `(C+1,C)`.

`S(n,m,C) = O(C)` : el tamaño del vector `alfa`.

El algoritmo: (usa técnica de terminación forzada; el invariante se modifica ligeramente): **[10/10]**

```
[ Ctx: alfa: array[0..C] of bool, d: array[1..m] of nat;
       d[1..m] = (d1,…,dm) ∧  d1<d2, …<dm

     {Pre: T}

     if C=0
          then n:= 1
          else for i:= 1 to C → alfa[i]:= false rof;
               alfa[0]:= true;
               n,k:= 1,C;
               ncent:= C+1;

               {Inv:  0 ≤ n ≤ ncent ≤ C+1  ∧  0 ≤ k ≤ C  ∧
                         alfa[0..k] = α(n-1,0..k) ∧  alfa[k+1..C] = α(n,k+1..C) ∧
                         (∀n1| 0≤n1<n : α(n1,C)) ∧ (ncent=C+1 cor ¬α(n,C))
               }

               do   n≠ncent → j:= 1;
                              jcent:= m+1;
                              alfa[k]:= false;
                              do j≠jcent
                                 →   if k≥d[j]
                                        then alfa[k]:= alfa[k] ∨ alfa[k-d[j]];
                                             j:= j+1
                                        else  jcent:= j
                                     fi
                              od;

                              if k=C   then if ¬alfa[n,C]
                                                  then   ncent:= n
                                             fi
                                       else if k=0
                                                  then n,k:= n+1,C
                                                  else k:=k-1
                                              fi
                                       fi
                              fi
               od
     fi
     {Pos: n = (min n1|: ¬α(n1,C)) }
]
```

Una variante, usando BLC: **[10/10]**

```
[ Ctx: alfa: array[0..C] of bool, d: array[1..m] of nat;
       d[1..m] = (d1,…,dm) ∧  d1<d2, …<dm

     {Pre: T}
```

```
for i:= 1 to C → alfa[i]:= false rof;
alfa[0]:= true;
n,k:= 1,C;

alfa[n,k]:= false;
do j≠jcent
     →   if k≥d[j]
             then alfa[k]:= alfa[k] ∨ alfa[k-d[j]];
                  j:= j+1
             else  jcent:= j
         fi
od;

{Inv:  0 ≤ n ≤ C+1  ∧  0 ≤ k ≤ C  ∧
        alfa[0..k] = α(n-1,0..k) ∧  alfa[k+1..C] = α(n,k+1..C)  ∧
        (∀n1| 0 ≤ n1 < n : α(n1,C))
}

do  ¬alfa[n,C]
        →   if k=0
                then n,k:= n+1,C
                else k:=k-1
            fi;

            j:= 1;
            jcent:= m+1;
            alfa[k]:= false;
            do j≠jcent
                →   if k≥d[j]
                        then alfa[k]:= alfa[k] ∨ alfa[k-d[j]];
                             if alfa[k]    then jcent:= j
                                           else j:= j+1
                             fi
                        else jcent:= j
                    fi
            od;
        od
{Pos: n = (min n1|: ¬α(n1,C)) }
]
```

2 **[50/100]**

Variante 2.1

a1 Sean:

[20/50]

V = {(x,y,z):nat^3 | 0≤x≤a ∧ 0≤y≤b ∧ 0≤z≤c ∧ x+y+z=a} [7/20]

E = {⟨(x_1,y_1,z_1),(x_2,y_2,z_2)⟩:VxV | Existe op: (x_1,y_1,z_1) $\rightarrow_{op}$ (x_2,y_2,z_2)} [5/20]

Los *nodos* en V son triplas de números naturales, que representan el estado de los frascos en cuanto a la cantidad de vino que contienen. Sea N el número de nodos.

Con $\rightarrow_{op}$ se denota una relación binaria entre nodos que representa las transiciones posibles por transvasar líquidos entre frascos. Solo hay 12 posibles transiciones de un nodo hacia otros:

[4/20]

$(x_1,y_1,z_1) \rightarrow_{AB} (x_1+y_1-b,b,z_1)$, si $0 \le x_1+y_1-b \le a$
$(x_1,y_1,z_1) \rightarrow_{AB} (0,x_1+y_1,z_1)$, si $0 \le x_1+y_1 \le b$
$(x_1,y_1,z_1) \rightarrow_{AC} (x_1+z_1-c,y_1,c)$, si $0 \le x_1+z_1-c \le a$
$(x_1,y_1,z_1) \rightarrow_{AC} (0,y_1,x_1+z_1)$, si $0 \le x_1+z_1 \le c$

$(x_1,y_1,z_1) \rightarrow_{BA} (a,x_1+y_1-a,z_1)$, si $0 \le x_1+y_1-a \le b$
$(x_1,y_1,z_1) \rightarrow_{BA} (x_1+y_1,0,z_1)$, si $0 \le x_1+y_1 \le a$
$(x_1,y_1,z_1) \rightarrow_{BC} (x_1,y_1+z_1-c,c)$, si $0 \le y_1+z_1-c \le b$
$(x_1,y_1,z_1) \rightarrow_{BC} (x_1,0,y_1+z_1)$, si $0 \le y_1+z_1 \le c$

$(x_1,y_1,z_1) \rightarrow_{CA} (a,y_1,x_1+z_1-a)$, si $0 \le x_1+z_1-a \le c$
$(x_1,y_1,z_1) \rightarrow_{CA} (x_1+z_1,y_1,0)$, si $0 \le x_1+z_1 \le a$
$(x_1,y_1,z_1) \rightarrow_{CB} (x_1,b,y_1+z_1-b)$, si $0 \le y_1+z_1-b \le c$
$(x_1,y_1,z_1) \rightarrow_{CB} (x_1,y_1+z_1,0)$, si $0 \le y_1+z_1 \le b$

En realidad, se puede ser más preciso, observando –por ejemplo- que cuando el frasco A se llena, en los otros no puede haber vino.

[4/20]

El problema se resuelve afirmativamente si existe un camino $(a,0,0) \rightarrow^* (a/2,x,y)$ en el grafo.

b1 **[30/50]**

De cada nodo (x_1,y_1,z_1) salen, a lo sumo, 12 arcos. En total, hay 12N arcos. [5/30]

Se observa que $|V| = N \le (a+1)(b+1)(c+1)$. Es decir, el número de nodos es $N = O(a^3)$. [10/30]

En realidad, se pueden estimar mejor de manera exacta:

$$
\begin{aligned}
& |V| \\
= \quad & (+i|\ 0\le i\le a : (+j|\ 0\le j\le \min(a-i,b) : (+k|\ 0\le k\le \min(a-i-j,c) : 1))) \\
\le \quad & (+i|\ 0\le i\le a : (+j|\ 0\le j\le a-i : (+k|\ 0\le k\le a-i-j : 1))) \\
= \quad & \qquad \langle \text{Cambio de variable: } p:= 1+a-i-j,\ 1\le p\le 1+a-i \rangle \\
& (+i|\ 0\le i\le a : (+p|\ 1\le p\le 1+a-i : \min(p,1+c))) \\
= \quad & (+i|\ 0\le i\le a : (1+a-i)(2+a-i)/2) \\
= \quad & \qquad \langle \text{Cambio de variable: } q:= 1+a-i,\ 1\le q\le 1+a \rangle \\
& (+q|\ 1\le q\le 1+a : q(q+1)/2) \\
= \quad & (+q|\ 1\le q\le 1+a : T_q) \\
= \quad & \qquad \langle \text{Suma de los primeros } 1+a \text{ números triangulares} \rangle \\
& (1+a)(2+a)(3+a)/6
\end{aligned}
$$

En cualquier caso, $N = \theta(a^3)$.

Variante b1.1: Usar Floyd-Warshall para calcular conectividad $(a,0,0) \rightarrow^* (a/2,x,y)$

$$\begin{aligned} T(a,b,c) &= T_{FW}(N) \\ &= O(N^3) \\ &= O(a^9). \end{aligned}$$ [4/30]

$$\begin{aligned} S(a,b,c) &= S_{FW}(N) \\ &= O(N^3) \\ &= O(a^9). \end{aligned}$$ [4/30]

Variante b1.2: Usar Dijkstra para calcular conectividad $(a,0,0) \to^* (a/2,x,y)$

$$\begin{aligned} T(a,b,c) &= T_D(N) \\ &= O(N \log N + e) \\ &= O(N \log N + 12N) \\ &= O(N \log N) \\ &= O(a^3 \log a^3) \\ &= O(a^3 \log a) \end{aligned}$$ [10/30]

$$\begin{aligned} S(a,b,c) &= S_D(N) \\ &= O(N) \\ &= O(a^3). \end{aligned}$$ [5/30]

Variante 2.2

a2 Sean:

[25/50]

$V = \{(y,z):nat^2 \mid 0 \le y \le b \wedge 0 \le z \le c\}$ [10/20]

$E = \{\langle (y_1,z_1),(y_2,z_2)\rangle : V\times V \mid \text{ Existe op: } (y_1,z_1) \to_{op} (y_2,z_2)\}$ [7/20]

Los *nodos* en V son parejas de números naturales, que representan el estado de los frascos B y C en cuanto a la cantidad de vino que contienen. El resto del vino está en el frasco A. Sea N el número de nodos.

Con $\to_{op}$ se denota una relación binaria entre nodos que representa las transiciones posibles por transvasar líquidos entre frascos. Solo hay, a lo más, 8 posibles transiciones de un nodo hacia otros:

[4/20]

$$\begin{aligned}
(y_1,z_1) &\to_{AB} (b,z_1) \\
(y_1,z_1) &\to_{AB} (a-z_1,z_1) \\
(y_1,z_1) &\to_{AC} (y_1,c) \\
(y_1,z_1) &\to_{AC} (y_1,y_1-a)
\end{aligned}$$

$$\begin{aligned}
(y_1,z_1) &\to_{BA} (0,z_1) \\
(y_1,z_1) &\to_{BC} (y_1+z_1-c,c) && \text{, si } c \le y_1+z_1 \\
(y_1,z_1) &\to_{BC} (0,y_1+z_1) && \text{, si } y_1+z_1 \le c
\end{aligned}$$

$$\begin{aligned}
(y_1,z_1) &\to_{CA} (y_1,0) \\
(y_1,z_1) &\to_{CB} (b,y_1+z_1-b) && \text{, si } b \le y_1+z_1 \\
(y_1,z_1) &\to_{CB} (y_1+z_1,0) && \text{, si } y_1+z_1 \le b
\end{aligned}$$

[4/20]

El problema se resuelve afirmativamente si existe un camino $(0,0) \to^* (a/2+x,\ a/2-x)$ en el grafo.

b2 [30/50]

De cada nodo (y_1, z_1) salen, a lo sumo, 8 arcos. En total, hay 8N arcos. [5/30]

Se observa que $|V| = N = (b+1)(c+1)$. Es decir, el número de nodos es $N = O(b^2)$. [10/30]

Variante b1.1: Usar Floyd-Warshall para calcular conectividad $(0,0) \rightarrow^* (a/2+x,\ a/2-x)$

$$
\begin{aligned}
T(b,c) &= T_{FW}(N) && [4/30]\\
&= O(N^3)\\
&= O(b^6).
\end{aligned}
$$

$$
\begin{aligned}
S(b,c) &= S_{FW}(N) && [4/30]\\
&= O(N^3)\\
&= O(b^6).
\end{aligned}
$$

Variante b1.2: Usar Dijkstra para calcular conectividad $(a,0,0) \rightarrow^* (a/2,x,y)$

$$
\begin{aligned}
T(b,c) &= T_D(N) && [10/30]\\
&= O(N \log N + e)\\
&= O(N \log N + 8N)\\
&= O(N \log N)\\
&= O(b^2 \log b^2)\\
&= O(b^2 \log b)
\end{aligned}
$$

$$
\begin{aligned}
S(b,c) &= S_D(N) && [5/30]\\
&= O(N)\\
&= O(b^2).
\end{aligned}
$$