PARTICIÓN

Un ejercicio de programación dinámica

Considérese un conjunto X, $|X|=n$, y una función $s:X \to \mathbf{nat}^+$.

Se quiere decidir si existen conjuntos $A,B \subseteq X$ tales que:

$A \cup B = X$

$A \cap B = \varnothing$

$(+x|\ x\in A : s.x) = (+x|\ x\in B : s.x)$

Sea $\sigma = (+x|\ x\in X : s.x)$.

Si σ es impar, la respuesta es negativa, porque las sumas sobre los subconjuntos deben ser iguales y el total es igual a σ.

Si σ es par, se puede plantear un algoritmo de programación dinámica que determine la respuesta:

1 LENGUAJE

Sea $X = \{x_1, \ldots, x_n\}$

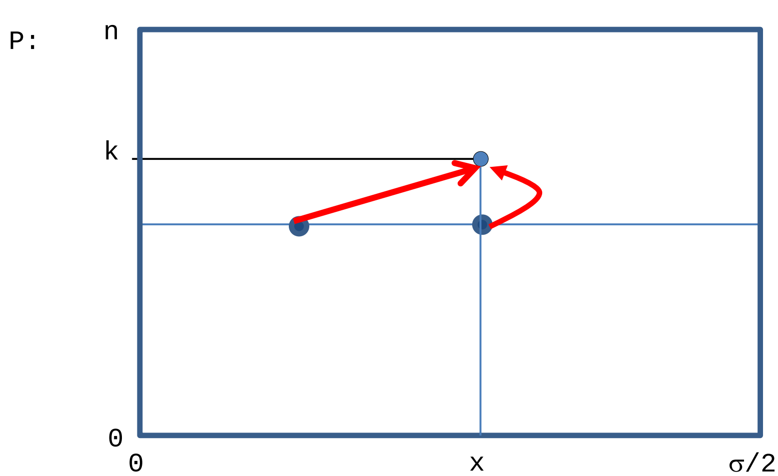
$p(x,k)$ $\approx$ "x es construible como suma de algunos $s.x_1, \ldots, s.x_k$" , con $0\le x\le\sigma/2$, $0\le k\le n$.

$p(\sigma/2,n)$ $\equiv$?

2 RECURRENCIA

$$
\begin{array}{lll}
p(x,k) & \equiv (x=0) & ,\ 0=k,\ 0\le x\le\sigma/2 \\
 & \equiv p(x,k-1) & ,\ 0<k\le n,\ 0\le x<x_k\le\sigma/2 \\
 & \equiv p(x,k-1) \vee p(x-x_k,k-1) & ,\ 0<k\le n,\ 0\le x_k\le x\le\sigma/2
\end{array}
$$

3 DIAGRAMA DE NECESIDADES



4 ESTRUCTURA DE DATOS + INVARIANTE

El patrón es similar al del problema del morral (pero: abscisas intercambiadas con las ordenadas). Se puede usar como estructura de datos auxiliar una fila que se llena de arriba abajo.

```
P[1..n]: bool

Inv PP:   0<k≤n+1 ∧ (∀y| x < y ≤ σ/2 : P[y] ≡ p(y,k))
                  ∧ (∀y| 0 < y ≤ x   : P[y] ≡ p(y,k-1))
```

5 COMPLEJIDAD

```
T(n)  = (nσ/2) * θ(1)         // nσ/2 iteraciones de costo θ(1)
      = θ(nσ)

S(n)  = θ(σ/2)
      = θ(σ)
```

6 SOLUCIÓN

```
[Ctx : X={x_1,x_2,…,x_n} ∧ n>0 ∧ (∀k| 1≤k≤n: s.x_k:nat)

for x:= 0 to σ/2 → P[x]:= (x=0) rof;

x,k:= σ/2,1;

{Inv PP}

do k≠n+1
  → if x<s.x_k   then x,k:= σ/2,k+1
                 else p[x]:= p[x] ∨ p[x-s.x_k];
                      x:= x-1
     fi
od

{Pos R: P[σ/2] = p(σ/2,n)}
]
```

DETERMINACIÓN DE LA PARTICIÓN, CUANDO EXISTE

Para saber cómo formar los conjuntos `A` y `B`, cuando se puede hacer la partición, hay que recordar las decisiones en una estructura de datos matricial, i.e., guardar la información de si $s.x_k$ se usa para construir `x` (la rama `else` del condicional en el cuerpo del ciclo).

Si se usa una matriz

```
M[1..n,0..σ/2]:bool
```

para recordar las decisiones, se puede proceder así:

```
[Ctx : X={x1,x2,…,xn} ∧ n>0 ∧ (∀k| 1≤k≤n: s.xk:nat)

M:= false;

for x:= 0 to σ/2 → P[x]:= (x=0) rof;

x,k:= σ/2,1;

{Inv PP ∧ "M recuerda las decisiones"}

do k≠n+1
   → if x<s.xk    then x,k:= σ/2,k+1
                  else p[x]:= p[x] ∨ p[x-s.xk];
                       x:= x-1;
                       M[x,k]:= true
      fi
od

{R0: P[σ/2] = p(σ/2,n) ∧ "M recuerda las decisiones"}

A= ∅;
x:= σ/2;
for k:= n downto 0
   → if M[x,k]  then   A:= A∪{xk};
                       x:= x-xk
      fi
rof

{R1: (+y| y∈A: y) = σ/2}

B:= X\A

{Pos: (+y| y∈A: y) = (+y| y∈B: y) ∧ A∪B=X ∧ A∩B=∅}

]
```

Las complejidades del nuevo algoritmo son

$T1(n) = \theta(n\sigma) + \theta(n)$

$\quad\quad = \theta(n\sigma)$

$S1(n) = \theta(\sigma/2)$