Programación dinámica I
(métodos top-down)

Un "algoritmo" para programar dinámicamente:
(ie, Agenda)

1) Especificar un problema Π de optimización / búsqueda / decisión / control / etc sobre un espacio combinatorio S

2) Describir una función recursiva f que resuelva Π tal que...

...3) f tiene la propiedad de superposición de subproblemas

4) Proponer una estructura de memoización M apropiada.

5) Diseñar un algoritmo A para f usando M.

6) Demostrar que A tiene la complejidad deseada.

7) Demostrar que f resuelve Π.

<u>Agenda</u>

- contar subconjuntos (calentamiento)
- problema del CD (básico)
- diseño de párrafos (1R2C 2020)

•) Contar subconjuntos (coleccionamiento)

Dados $m, k \in \mathbb{N}$, determinar la cantidad de subconjuntos de $\{1, \dots, m\}$ que tienen exactamente k elementos

1) Especificación (espacio combinatorio)

$$|S_{mk}| \text{ donde } S_{mk} = \{ S \subseteq \{1, \dots, m\} \text{ tq } |S| = k \}$$

es un problema de conteo

2) Función recursiva $f(m,k) = |S_{mk}|$

$$f : \mathbb{N} \times \mathbb{Z} \to \mathbb{N}$$

$$f(m,k) = \begin{cases} 1 & \text{si } m = 0 \text{ y } k = 0 \\ 0 & \text{si } k < 0 \text{ o } k > m \\ f(m-1,k) + f(m-1,k-1) & \text{cc.} \end{cases}$$

Semántica: $f(m,k) = |S_{mk}|$ es la cantidad de formas de tomar k elementos de $\{1, \dots, m\}$

Justificación: dos formas de tomar k elementos para $S \subseteq \{1, \dots, m\}$

•) $m \notin S \Rightarrow S$ es una forma de tomar k eltos de $\{1, \dots, m-1\}$

•) $m \in S \Rightarrow S - \{m\}$ es una forma de tomar $(k-1)$ eltos de $\{1, \dots, m-1\}$

Estas formas son excluyentes $\Rightarrow$ para contarlas alcanza con sumarlas

Obs: a $f(m,k)$ se lo suele denotar $\binom{m}{k}$

3) Propiedad de superposición de subproblemas.

\# llamadas recursivas $= \Omega\left(\binom{m}{\tilde{k}}\right)$ $\quad (\tilde{k} = \max\{0, \min\{n, k\}\})$

\# subinstancias $= O(n\,\tilde{k})$

Como $\dfrac{\#\ \text{llamadas}}{\#\ \text{subinstancias}} \xrightarrow[\substack{n\to\infty \\ 3 \le n-k \le n-3}]{} \infty \Rightarrow$ conviene memoizar

4) Memoización (básico)

Diccionario (representación subinstancia $\jmath$) $\rightarrow$ (solución $\jmath$)

representación

$\quad\quad\quad\quad \| \quad\quad\quad\quad\quad\quad\quad\quad \|$

par (n, k) $\quad\quad\quad\quad\quad\quad f(n, k)$

$M(n,k) \stackrel{\Delta}{=} f(n,k)$ si $0 < k \le n \Rightarrow$

Matriz ($O(1)$ para definir, definido? $\hookrightarrow$ significado)

5) Algoritmo top-down (5.6)

"Receta" inicial: - escribir el algoritmo recursivo, pero

- antes del caso recursivo verificar si ya fue computado usando M

combinatorio(n, k):

Sea M matriz de $n+1 \times \tilde{k}+1$ con valor inicial $\perp$ $\quad O(n\tilde{n})$

Retornar $f(n, k)$ donde: $\quad O(n\,\tilde{n})$

$f(i, j)$ {

si $i = j = 0 \Rightarrow$ retornar 1 $\quad O(1)$

si $j < 0$ o $j > i \Rightarrow$ retornar 0 $\quad O(1)$

$O(1)$ x llamado $\quad O(1)$ si $M[i, j] = \perp$ poner $M[i,j] = f(i-1, j-1) + f(i-1, j)$

retornar $M[i, j]$. $O(1)$

}

7) Demostrar que f resuelve, i.e. $f(m,k) = |S_{mk}|$

<u>Teorema</u>: $f(m,k) = |S_{mk}|$ para todos $0 \leq k \leq m$

<u>Demostración</u>: por inducción en m

Caso base $m=0$) $f(0,0) = 1 = |\{\emptyset\}| = |\{S \subseteq \overbrace{\{1,\ldots,0\}}^{\emptyset} \text{ tq } |S| = 0\}| = |S_{00}|$

Paso inductivo $m>0$) Sean $S^+, S^- \subseteq S_{mk}$ tales que $S \in S^+$ sii $m \in S$, y $S \in S^-$ sii $m \notin S$.
Claramente, $S_{mk} = S^+ \uplus S^- \Rightarrow |S_{mk}| = |S^+| + |S^-|$.

Ciertamente, $S \in S^-$ sii $S \in S_{m-1,k}$ y
$S \in S^+$ sii $S - \{m\} \in S_{m-1,k-1}$ y $m \in S$ $\Rightarrow$
$|S^-| = |S_{m-1,k}|$ y $|S^+| = |S_{m-1,k-1}|$. $\Rightarrow$

Por HI:

$$|S_{mk}| = |S^+| + |S^-| = |S_{m-1,k}| + |S_{m-1,k-1}| = f(m-1,k) + f(m-1,k-1)$$

$$|S_{mk}| = f(m,k) \quad \blacksquare$$

•) Problema del CD (básico)

Dado un CD de K minutos y m canciones con duraciones $w_1, \dots, w_m$, determinar la máxima cantidad de minutos que podemos grabar sin cortar ni repetir canciones.

1) Especificación

función objetivo — espacio combinatorio

$$cd(w, k) = \max\left\{\sum_{i \in S} w_i \mid S \in S_{wm}\right\}, \quad S_{wm} = \left\{S \subseteq \{1, \dots, |w|\} \ , \ \sum_{i \in S} w_i \leq K\right\}$$

problema de optimización.

2) Función recursiva $f_w(m, k) = cd(w, k)$

$$f_w : \mathbb{N} \times \mathbb{Z} \to \mathbb{N} \cup \{-\infty\}$$

$$f_w(m, k) = \begin{cases} -\infty & k < 0 \\ 0 & m = 0 \ , \ k \geq 0 \\ \max\{f(m-1, k), w_m + f(m-1, k-w_m)\} & \text{c.c.} \end{cases}$$

Semántica: $f_w(m, k) = cd(w, k)$ es el máximo tiempo grabable con las canciones $1, \dots, m$ en el CD de capacidad K.

Justificación: dos opciones para la canción m:

•) descartarla $\Rightarrow$ tengo que maximizar grabación $\leq K$ con $1, \dots, m-1$

•) grabarla $\Rightarrow$ tengo que maximizar grabación $\leq K - w_m$ con $1, \dots, m-1$ y yo ocupé w_m espacio.

De ambas opciones excluyentes, me quedo con la que consume más espacio.

3) Propiedad de superposición de subproblemas.

$\#$llamadas $= \Omega(2^n)$ $\#$subinstancias $= O(nk)$ $\Rightarrow$ vale la pena memorizar cuando $2^n \gg nk$. Sino, no vale el esfuerzo.

4) Memorización cuando $k \ll n^{-1} 2^n$.

M matriz de $n \times k$ tq $M[i,j] \stackrel{\Delta}{=} f_w(i,j)$ para todo $1 \leq i \leq n$, $0 \leq j \leq k$.
$\Rightarrow$ operaciones en $O(1)$

5) Algoritmo top-down

```
cd({w_1,...,w_n}, k)
  Inicializar M de n+1 x k+1 con ⊥
  retornar f(n,k) tq:
   f(i,j) {
        si j<0 retornar -∞
        si i=0 retornar 0
        si M[i j]=⊥ poner M[i,j] =
             max{ f(i-1,j), w_i + f(i-1, j) }
        retornar M[i,j].
   }
```

6) Complejidad: $O(nk)$ tiempo y espacio.

7) Teorema: $cd(w,k) = f_w(n,k)$ para todo $k \geq 0$ con $n = |w|$

Demostración. Por inducción en n

- $n=0 \Rightarrow cd(\phi,0) = \max\{\overbrace{\sum_{i\in S} w_i}^{0} \text{ tq } \overbrace{S}^{\phi} \subseteq \overbrace{\{1,\dots,0\}}^{\phi}, \overbrace{\sum_{i\in S} w_i}^{0} \leq 0\} = 0 = f_w(0,0)$

- $n>0 \Rightarrow$ Recordemos que $cd(w,k) = \max\{\sum_{i\in S} w_i \text{ tq } S \in S_{wk}\}$

donde $S_{wk} = \{S \subseteq \{1,\dots,|w|\}, \sum_{i\in S} w_i \leq k\}$.

Sean $S^+ = \{S \in S_{wk} \text{ tq } n \in S\}$, $S^- = \{S \in S_{wk} \text{ tq } n \notin S\}$

Claramente (a) $S_{wk} = S^+ \oplus S^-$. Mas aún,

$w' = w - w_n$

b) $S \in S^- \iff S \subseteq \{1,\dots,|w|-1\}$ y $\sum_{i\in S} w_i \leq k \iff S \in S_{w'k}$

$k' = k - w_n$

c) $S \in S^+ \iff S - \{n\} \subseteq \{1,\dots,|w|-1\}, \sum_{i\in S} w_i \leq k', n \in S \iff S \in S_{w'k'}, n \in S$

$(S^w_{n-1\,j} = \phi \ \forall j < 0)$

$\Rightarrow cd(w,k) = \max\{\sum_{i\in S} w_i \text{ tq } S \in S_{wk}\} =$

$\overset{(a)}{=} \max\{\max\{\sum_{i\in S} w_i \text{ tq } S \in S^-\}, \max\{\sum_{i\in S} w_i \text{ tq } S \in S^+\}\}$

$\overset{(b+c)}{=} \max\left\{\begin{array}{l} \max\{\sum_{i\in S} w_i \text{ tq } S \in S_{w'k}\}, \\ \max\{w_n + \sum_{i\in S} w_i \text{ tq } S \in S_{w'k'}\} \end{array}\right\} =$

$= -\infty$ si $w_n > k$
porque $S = \phi$ en este caso

$= \max\{cd(n-1,k), \overbrace{cd(n-1,k')}^{-\infty \text{ si } w_n > k} + w_n\} =$

$\overset{HI}{=} \max\{f_w(n-1,k), f_w(n-1,k') + w_n\} = f_w(n,k)$ ▪

•) Diseño de párrafos

2) Un problema común en los procesadores de texto es decidir dónde particionar un *string* S que representa un párrafo para formar los distintos reglones. En este problema, el objetivo es maximizar la belleza visual del texto resultante. Para ello, se define una matriz de belleza B con $|S| \times |S|$ entradas naturales, donde $B[i,j]$ representa la belleza que tendría un reglón que empieza en $S[i+1]$ y termina en $S[j]$ ($B[i,j] = 0$ si $j \leq i$). Luego, el problema de particionado en reglones consiste en determinar una secuencia de puntos de partición $0 = p_0 < p_1 < \ldots < p_{k+1} = |S|$ que maximice $\sum_{i=0}^{k} B[p_i, p_{i+1}]$. **Ayuda:** notar que para partir S en k reglones tenemos que elegir k de las $n-1$ posibles puntos de partición. Luego, existen $\sum_{k=0}^{n-1} \binom{n-1}{k}$ formas de partir un párrafo.

obs: B indexo desde 0 a m y S desde 1 a m.

Espacio: $\mathbb{P}_B = \{ P \subseteq \{1, \ldots, |B|-1\} \}$, i.e, cada $P \in \mathbb{P}_B$ elige la posición en la que empieza una línea (|B| = cantidad letras)

$$\text{belleza}(B) = \max_{\text{tq}} \sum_{i=0}^{k} B[p_i, p_{i+1}]$$

$$p_1 < \ldots < p_k \in \mathbb{P}_S$$

$$p_0 = 0, \quad p_{k+1} = |B|.$$

función recursiva.

$f_B : \mathbb{N} \to \mathbb{N}$

$$f_B(m) = \begin{cases} 0 & \text{si } m = 0 \\ \max\{ f_B(i) + B[i,m] \text{ tq } 0 \leq i < m \} & \text{cc} \end{cases}$$

Semántica

a) Definir la función ~~$belleza_S$~~ f_B : $\mathbb{N} \to \mathbb{N}$ tal que ~~$belleza_S$~~ $f_B(j)$ es la máxima belleza posible para el *substring* de S que empieza en $S[1]$ y termina en $S[j]$.

Justificación: el último caracter de S va en la última línea. luego, tengo que decidir la posición i donde empiezo la línea y tengo que partir el texto hasta la posición i.

$\#$ llamadas $= \Omega(2^n)$ $\Big\}$ $\Rightarrow$ tiene propiedad de superposición
$\#$ subinstancias $= O(n)$ $\Big\}$ de subproblemas.

Uso un vector m de n posiciones para memoizar donde
$m[j] = f_B(j) \;\forall\; 1 \leq j \leq n$

belleza(B):
poner $m[j] = \perp \;\forall\; 0 \leq j \leq n = |B|$
retornar $f(n)$ donde
$f(j)$ {
si $j = 0$ retornar 0
si $m[j] = \perp$ poner $m[j] =$
$\max\{f(i) + B[i,j],\; 0 \leq i < j\}$
retornar $m[j]$.

Complejidad: $O(n^2)$.

Teorema: belleza$(B) = f_B(n)$ $\quad n = |B|$

Demostración: por inducción en n.

• $n = 0$) belleza$(B) = \max\{\overset{0}{\underline{\sum_P \dots}}$ tq $\overset{\emptyset}{P} \in \mathbb{P}_B = \{\emptyset\}\} = 0 = f_B(0)$ ✓

• $n > 0$) Para cada $i = 0, \dots, n-1$, sea
$\mathbb{P}_i = \{P \in \mathbb{P}_B$ tq $\max\{P \cup \{0\}\} = i\}$.
ie, $\mathbb{P}_i$ tiene las "secuencias" cuyo máximo valor es i

Claramente, $\mathbb{P}_B = \mathbb{P}_0 \oplus \dots \oplus \mathbb{P}_{n-1}$. Más aún,
$P \in \mathbb{P}_i$ sii $P - i \in \{1, \dots, i-1\}, i \in P$ sii $P \in \mathbb{P}_{B|i}, i \in P$

($B|_i$ es la submatriz de belleza hasta i)

$$\text{belleza}(B) = \max_{\substack{\text{tq} \\ 0=p_0,\ p_1<\dots<p_k \in \mathbb{P}_B,\ p_{k+1}=m.}} \sum_{i=1}^{k} B[p_i, p_{i+1}]$$

$$= \max \left\{ \max_{\substack{\text{tq} \\ 0=p_0,\ p_1<\dots<p_k \in \mathbb{P}_k,\ p_{k+1}=m}} \left\langle B[p_k, m] + \sum_{i=1}^{k-1} B[p_i, p_{i+1}] \right\rangle \text{ para todo } p_k = 0, \dots, m-1 \right\}$$

$$= \max \left\{ \left\langle B[p_k, m] + \max_{\substack{\text{tq} \\ 0=p_0,\ p_1<\dots<p_{k-1} \in \mathbb{P}_{B|p_k}}} \left\{ \sum_{i=1}^{k-1} B[p_i, p_{i+1}] \right\rangle \right. \text{ para todo } p_k = 0, \dots, m-1 \right\}$$

$$= \max \{ B[p_k, m] + \text{belleza}(B|p_k) \ \forall\, p_k = 0, \dots, m-1 \}$$

$$= \max \{ B[i, m] + \text{belleza}(B|i) \quad \text{tq } 0 \le i \le m-1 \}$$

$$\overset{\text{HI}}{=} \max \{ B[i, m] + f_B(i),\ 0 \le i \le m-1 \}$$

$$= f_B(m) \quad \blacksquare$$

Conclusión: las demostraciones en estos casos obvios son poco interesantes y una buena justificación es más que suficiente para poder aplicar una inducción.