Algoritmos $\underbrace{\text{Codiciosos}}_{\text{Voraz, ávido, goloso ?}}$ ("greedy")

PARADIGMA: Construimos una solución iterativamente mediante una sucesión de elecciones "miopes" (locales) esperando que la "miopía" de la selección no produzca una solución demasiado mala. La dificultad radica en encontrar situaciones en las que un algoritmo "greedy" produzca la solución correcta.

Ejemplo: [Scheduling] Tenemos una serie de trabajos que deben ser realizados en un "recurso" común, tenemos una tabla de estos trabajos

trabajo	1	2	3	...	n
duración	ℓ_1	ℓ_2	ℓ_3		ℓ_n
Prioridad >0	w_1	w_2	w_3		w_n

más grande lleva a demora más costosa así que queremos que los de mayor prioridad se cumplan más pronto.

debemos realizarlos todos y queremos hacerlo de la forma más eficiente posible, en el sentido en que:

PROBLEMA OPTIMIZACIÓN:

X = "Conjunto de órdenes posibles σ"

Dado un orden σ y una tarea j definimos

$C_j(\sigma)$ = "Tiempo usado hasta completar la tarea j siguiendo el orden σ"

$$t(\sigma) = \sum_{j=1}^{n} w_j C_j(\sigma)$$

$$t^* := \min \{ t(\sigma) : \sigma \in X \}$$

Problema de optimización

PARA INTENTAR DESCUBRIR UN ALGORITMO VAMOS A PENSAR EN DOS CASOS ESPECIALES:

(1) Caso de longitudes iguales (pesos arbitrarios)
(2) Caso de pesos iguales (longitudes arbitrarias).

Caso especial (1)

(1) Si la longitud es ℓ y los pesos w están dados entonces:

(i) Ordenamos $w_{\sigma(1)} \geq w_{\sigma(2)} \geq \dots \geq w_{\sigma(n)}$ *

(ii) El tiempo $t(\sigma) = w_{\sigma(1)} + 2\, w_{\sigma(2)} + \dots + 2\, w_{\sigma(n)}$ es mínimo.

Obs: Si τ es cualquier orden que no cumple esto, buscamos el primer cambio: intercambio

$$w_{\tau(1)} \geq w_{\tau(2)} \geq w_{\tau(3)} \geq \dots \geq w_{\tau(n-1)} < w_{\tau(n)}$$

$$w_{\sigma(1)} \geq w_{\sigma(2)} \geq \dots \geq w_{\sigma(n-1)} \geq w_{\sigma(n)}$$

$$t(\sigma) = w_{\sigma(n-1)}(n-1) + w_{\sigma(n)}\, n + \text{"común"}$$

$$t(\tau) = w_{\tau(n-1)}(n-1) + w_{\tau(n)}\, n + \text{"común"}$$

el tiempo de τ baja invirtiendo $\tau(n)$ y $\tau(n-1)$

$$(n-1)\, w_{\tau(n)} + n\, w_{\tau(n-1)} < (n-1)\, w_{\tau(n-1)} + n\, w_{\tau(n)}$$

así que τ no es óptimo a menos que la condición * se cumpla.

Obs: Insertando los items en $n\log(n)$ en un max-priority queue y llamando el máximo n veces realizamos un cómputo total $2n\log(n) \in O(n\log(n))$.
MUY EFICIENTE.

(2) Caso especial de pesos iguales w

(i) Ordenamos longitudes σ^*: $\ell_{\sigma(1)} \leq \ell_{\sigma(2)} \leq \dots \leq \ell_{\sigma(n)}$

(ii) $t(\sigma) = w\,\ell_{\sigma(1)} + w(\ell_{\sigma(1)} + \ell_{\sigma(2)}) + \dots + w(\ell_{\sigma(1)} + \dots + \ell_{\sigma(n)})$

$$= w\left[n\,\ell_{\sigma(1)} + (n-1)\,\ell_{\sigma(2)} + \dots + 1\cdot \ell_{\sigma(n)}\right]$$

es mínimo.

Obs: Si σ es un orden cualquiera y
$\exists j : \ell_{\sigma(j)} > \ell_{\sigma(j+1)}$

Consideremos el orden $\mathfrak{I} = (\sigma(j), \sigma(j+1)) \circ \sigma$ ← intercambia el output de $\sigma(j)$ y $\sigma(j+1)$

$$t(\mathfrak{I}) = \sum_{i<j} (n-i)\ell_{\sigma(i)} + \underbrace{(n-j)\ell_{\sigma(j+1)} + (n-(j+1))\ell_{\sigma(j)}}_{\wedge} + \sum_{i>j+1} (n-i)\ell_{\sigma(i)}$$

$$\wedge$$

$$t(\sigma) = \quad + (n-j)\ell_{\sigma(j)} + (n-(j+1))\ell_{\sigma(j+1)} +$$

todo óptimo cumple la desigualdad de arriba y esto determina el valor óptimo completamente.

Obs: Se puede calcular en $O(n\log(n))$ usando un MIN-priority queue. (MUY EFICIENTE).

Qué hacer para el caso general?

Queríamos $w_{\sigma(1)} \geq \dots \geq w_{\sigma(n)}$ y
$\ell_{\sigma(1)} \leq \dots \leq \ell_{\sigma(n)}$

pero eso NO ES POSIBLE AL TIEMPO EN GENERAL, como se ve en el siguiente

Ejemplo:

Trabajo	1	2
Longitud	$\ell_1 = 5$	$\ell_2 = 2$
Peso	$w_1 = 3$	$w_2 = 1$

$\ell_2 \leq \ell_1$
$w_2 \leq w_1$

AMBOS para el mismo lado, qué hacer?

Queremos asignar a cada trabajo un "puntaje" (SCORE) S_j numérico de tal forma que un orden σ^* con $S_{\sigma(1)} \geq S_{\sigma(2)} \geq \dots \geq S_{\sigma(n)}$ sea óptimo.

Idea: $S_j := \dfrac{w_j}{\ell_j}$

Para longitud ℓ, σ^* tiene pesos decrecientes

Para peso fijo σ^* tiene longitudes CRECIENTES

Ok* en casos especiales.

Teorema: Si definimos $S_j := \frac{w_j}{\ell_j}$ entonces todo orden J con $S_{J(1)} \geq S_{J(2)} \geq \dots \geq S_{J(n)}$ es óptimo para el problema de scheduling (es decir

$$t(J) := \sum_{j=1}^{n} w_j C_j(J) = \min\{t(\sigma) : \sigma \text{ orden}\} \Big)$$

ASUMIMOS QUE LOS NÚMEROS $\frac{w_j}{\ell_j}$ SON DISTINTOS...

Dem: [Utiliza "intercambio", la marca de algoritmos greedy por excelencia]

Suponga que σ no satisface la propiedad de arriba y sea $j = \min\{n \in \mathbb{N} : S_{\sigma(j)} < S_{\sigma(j+1)}\}$. Cambiando el orden inicial de los trabajos (relabeling) podemos asumir que

$$S_1 > S_2 > \dots > S_{j-1} > S_j < S_{j+1}.$$

Sea J el orden que resulta de intercambiar los trabajos j y $j+1$. Qué pasa con la función objetivo?

$$\ell(\sigma) = \sum_{i<j} w_i \ell_{\le i} + w_j \ell_{\le j} + w_{j+1} \ell_{\le j+1} + \sum_{k \geq j+2} w_k \ell_{\le k}$$

con $\ell_{\le t} = \ell_1 + \ell_2 + \dots + \ell_t$.

$$\ell(J) = \sum_{i<j} w_i \ell_{\le i} + \underbrace{w_{j+1}(\ell_{\le j-1} + \ell_{j+1}) + w_j(\ell_{\le j-1} + \ell_{j+1} + \ell_j)} + \sum_{k \geq j+2} w_k \ell_{\le k}$$

porque reordenan las ℓ's no cambia la suma

Luego $\ell(\sigma) - \ell(J)$ es igual a:

$$w_j(\cancel{\ell_{\le j-1}} + \cancel{\ell_j}) + w_{j+1}(\cancel{\ell_{\le j-1}} + \ell_j + \cancel{\ell_{j+1}}) -$$

$$w_{j+1}(\cancel{\ell_{\le j-1}} + \cancel{\ell_{j+1}}) + w_j(\cancel{\ell_{\le j-1}} + \ell_{j+1} + \cancel{\ell_j}) =$$

$$\ell(\sigma) - \ell(J) = w_{j+1}\ell_j - w_j \ell_{j+1} > 0$$

ver página sig.

equiv a

$$\begin{array}{ccc} \dfrac{w_j}{\ell_j} & < & \dfrac{w_{j+1}}{\ell_{j+1}} \\ \| & & \| \\ s_j & < & s_{j+1} \end{array}$$

Así que $\ell(J) < \ell(\sigma)$ y podemos "mejorar" a σ por lo tanto un minimizador σ^* DEBE SATISFACER

$$S_{\sigma^*(1)} > S_{\sigma^*(2)} > \dots > S_{\sigma^*(n)}$$

y como hay sólo un orden con esa propiedad tiene que ser el óptimo ■