Algoritmos Greedy

Técnicas de Diseño de Algoritmos

1er Cuatrimestre 2024

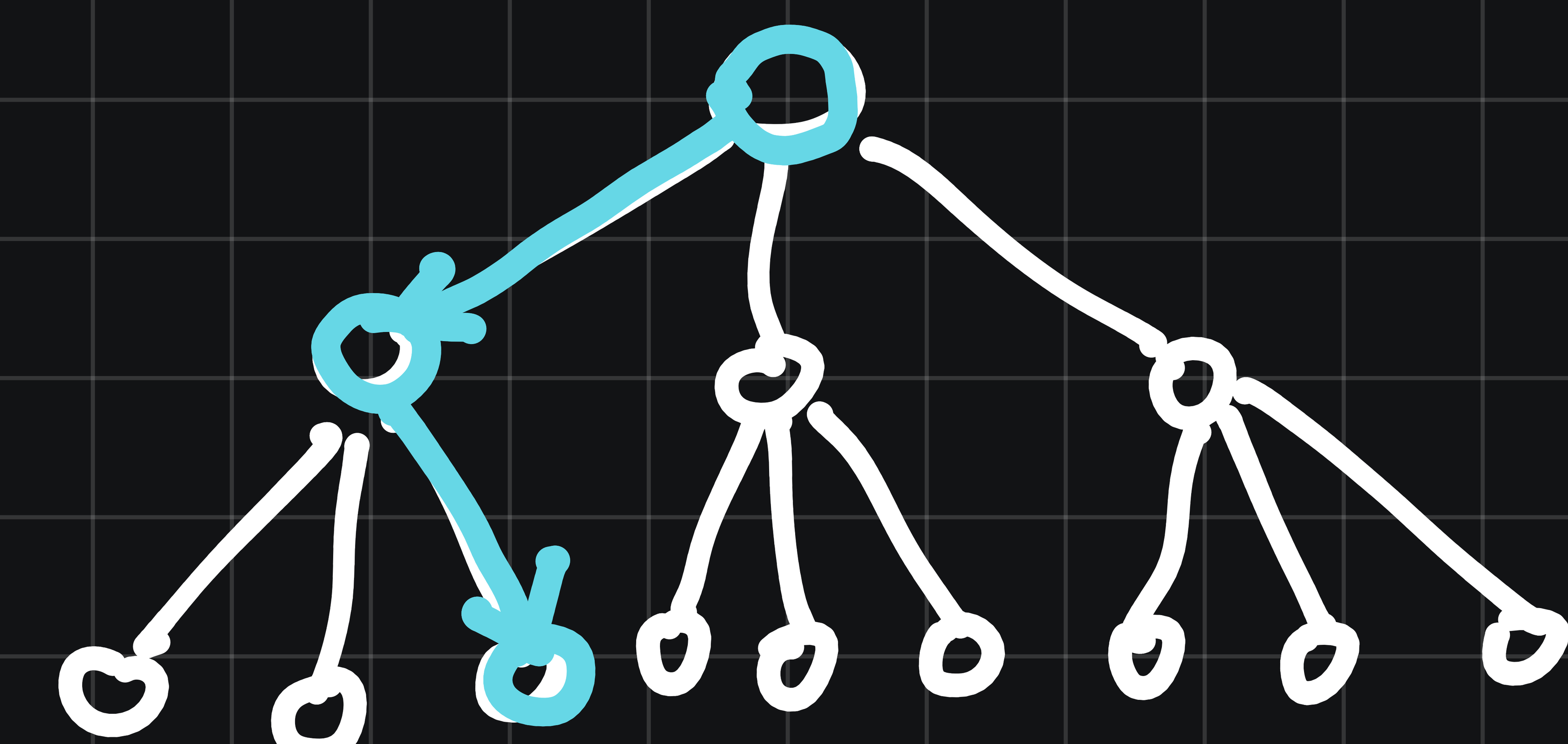
Repaso de la teoría

- Venimos estudiando algunos problemas cuyas soluciones se construyen tomando decisiones una a una.

- En cada elección, potencialmente hay una opción que localmente es la más atractiva (Ej. el valor máximo de la métrica que quiero optimizar)

Repaso de la teoría

- Es como tomar el árbol de backtracking y, en vez de comparar las ramas, elegir una según algún criterio.



- Frecuentemente implica ordenar los elementos del input de alguna manera específica.

Repaso de la teoría

- El algoritmo greedy para un problema dado nos proveerá una solución candidata
- La pregunta es si esa solución es correcta (por ejemplo, si es el óptimo)
- Es común que los algoritmos greedy no sean complejos de implementar y la dificultad esté en probar que son correctos.

Dos Problemas:

- CD (el regreso)
- Meximización

Problema del CD

Se tiene un CD que soporta hasta P minutos de música, y dado un conjunto con N canciones de duración p_i $(0 \leq i \leq N-1,$ $p_i \in \mathbb{N})$ queremos encontrar la mayor cantidad de minutos de música que podemos poner en el CD usando temas del conjunto sin repetir.

CD - Formulación Recursiva

Idea de la semántica: "¿Cuál es la máxima cantidad de minutos si mi capacidad es K y uso los temas del i en adelante?"

$$CD(i,K) = \begin{cases} -\infty & \text{si } i = N \text{ y } K < 0 \\ 0 & \text{si } i = N \text{ y } K \geq 0 \\ \max\left(CD(i+1, K),\ CD(i+1, K-P_i) + P_i\right) & \text{c.c.} \end{cases}$$

CD - Variante

- Ahora, el tema i tiene no solo una duración p_i asociada sino también un valor g_i que describe cuánto me gusto
- lo que voy a querer es obtener el compilado que maximice mi gusto

CD - Variante

- Ahora, el tema i tiene no solo una duración p_i asociada sino también un valor g_i que describe cuánto me gusta
- Lo que voy a querer es obtener el compilado que maximice mi gusto

(En esencia, estoy transformando el problema en la mochila (Knapsack))

CD - Cambio de semántica

Idea de la semántica: "¿Cuál es el máximo gusto acumulado si mi capacidad es K y uso los items del i en adelante?

¿Cómo debería modificar la formulación recursiva?

$$CD(i,k) = \begin{cases} -\infty & \text{si } i=N \text{ y } K<0 \\ 0 & \text{si } i=N \text{ y } K \geq 0 \\ \max\left(CD(i+1,K),\ CD(i+1,K-P_i)+P_i\right) & \text{c.c.} \end{cases}$$

CD - Formulación Variante

Idea de la semántica: "¿Cuál es el máximo gusto acumulado si mi capacidad es K y uso los temas del i en adelante?

$$CD(i,k) = \begin{cases} -\infty & \text{si } i = N \text{ y } K < 0 \\ 0 & \text{si } i = N \text{ y } K \geq 0 \\ \max\left(CD(i+1, K), CD(i+1, K-P_i) + g_i\right) & \text{c.c.} \end{cases}$$

CD - Variante

- Esta variante del problema admite los mismos algoritmos vistos hasta ahora, en particular la solución por PD.

CD - Variante adicional

- Ahora, Permitimos cortar un tema para completar el disco
- Tomemos un tema que dura p minutos y me gusta g. Si pongo α de los p minutos en el disco, eso me gusta

$$\frac{\alpha}{p} \cdot g$$

(ie. si un Tema me gusta g y pongo el 60%, eso me gusta $0.6g$)

CD - Variante fraccionada

- Esta variante aún podría resolverse con nuestra formulación, modificándola
- Se puede, por ejemplo, comparar una solución sin temas cortados con la mejor solución posible asumiendo cada tema como el que se corta
- Pero se puede hacer algo mejor

CD - Variante fraccionada

- En vez de determinar si pongo o no pongo cada tema, intentemos ir poniendo los que más me gustan hasta que alguno me quede cortado.

Pero...

$P=5$	$N=3$	
4	2	3
10	6	9

Eligirían el tema que me gusta 10 y 1/3 del que me gusta 9, mejor poner el que me gusta 9 y el que me gusta 6...

CD - Encontrando otra métrica

- Supongamos que los temas duran más de P minutos (puedo poner sólo parte de uno) y me gustan lo mismo

$N=6$ 8 10 → 15

20 20 → 12

¿Cuál conviene poner? el primero

CD - Encontrando otra métrica

- Supongamos que los temas duran más de P minutos (Puedo poner sólo parte de uno) y me gustan lo mismo

$N=6$ $8 \quad 10 \rightarrow 15$

$20 \quad 20 \rightarrow 12$

¿Cuál conviene poner? el primero

Podemos pensar que el primer tema "rinde más" que el segundo, pues da "más gusto por minuto".

CD - Cociente gusto por minuto

- Para el tema t, defino

$$p(t) = p_i \qquad g(t) = g_i \qquad q(t) = \frac{g(t)}{p(t)} = \frac{g_i}{p_i}$$

donde t es el i-ésimo tema en la entrada

- La mejor manera de aprovechar un minuto es poner el tema t^* tal que $q(t^*)$ es máximo.

$\Rightarrow$ Mi estrategia será ir poniendo en el disco los temas en orden de cociente decreciente hasta llenarlo

CD - Agregando temas por cociente

- Vamos a hacer inducción en $|D|$. Para ello, consideremos la siguiente formulación del problema:

$$CD(D,K) = \begin{cases} 0 & \text{si } D = \emptyset \\ \max\limits_{t \in D} \left\{ \underbrace{\frac{\min\{K, p(t)\}}{p(t)} \cdot g(t)} + \underbrace{CD(D - \{t\}, K - \min\{K, p(t)\})} \right. & \text{si no} \end{cases}$$

el máximo entre todos los temas

el gusto que me da poner este tema (o lo que pueda meter del mismo)
Obs: si el disco está lleno da 0

la solución del CD suponiendo que puse todo lo que pude del tema

- Esta formulación nos viene bien para hacer inducción

CD - Agregando temas por cociente

- Afirmamos que para la instancia $CD(D, k)$, $k \geq 0$, hay una solución óptima usando los temas de máximo cociente disponibles (cuando D tiene al menos un tema)
- Notamos con $t^*(D)$ al tema de D tal que $q(t^*(D))$ es máximo.
- Haciendo inducción en $|D|$, cuando $D = \phi$, la afirmación es trivialmente verdadera
- Para el paso inductivo, sup. que vale con $|D| = n$, tomo una instancia con $|D| = n+1$.

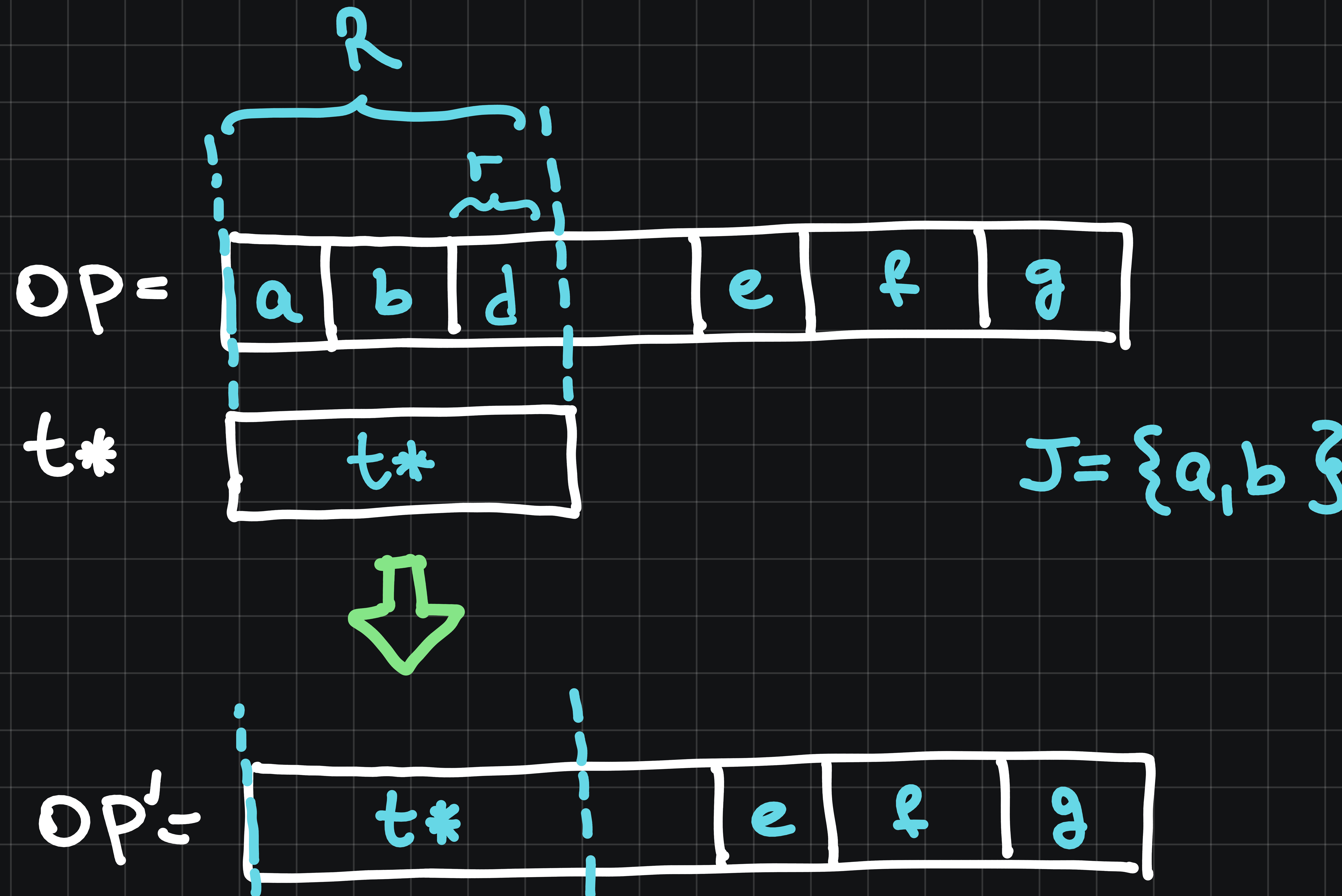
CD - Optimalidad

- Sea Op una solución óptima de $CD(D, K)$ y supongamos que no contiene a $t^* = t^*(D)$. Tomemos $R = \min(K, P(t^*))$ minutos de Op y formemos una solución alternativa Op' reemplazándolos por minutos de la canción t^*.

- Sea J el conjunto de los temas de Op que están en el segmento reemplazado, d el tema que haya quedado cortado (si hay), r la longitud de d que haya quedado en el segmento reemplazado

CD - Optimalidad

Es decir



CD - Optimalidad

Tomemos

$$x = \underset{z \in J \cup \{d\}}{\operatorname{argmax}} \left(q(z) \right)$$

Luego, tenemos que el gusto en el segmento reemplazado es:

$\forall j,\ \frac{g(j)}{P(j)} \leq \frac{g(x)}{P(x)}\ ;\ \frac{g(d)}{P(d)} \leq \frac{g(x)}{P(x)}$

$$\left(\sum_{j \in J} g(j) \right) + \frac{r}{P(d)} g(d) = \left(\sum_{j \in J} \frac{P(j) g(j)}{P(j)} \right) + \frac{r}{P(d)} g(d) \leq \left(\sum_{j \in J} \frac{g(x) P(j)}{P(x)} \right) + \frac{r}{P(x)} g(x)$$

$$= \frac{g(x)}{P(x)} \cdot \left(\sum_{j \in J} P(j) \right) + r = \frac{g(x)}{P(x)} R \leq \frac{g(t^*)}{P(t^*)} R$$

Luego, el gusto de Op' es al menos tanto como el de Op

CD - Optimalidad

- Para los $k-R$ minutos restantes del CD, el problema es $CD(D-\{t^*\}, k-\min\{k, P(t^*)\})$, donde por hipótesis inductiva, hay una solución óptima usando los temas de máximo cociente disponibles, completando la demostración.

$\blacksquare$

CD - Algoritmo

1. ordenar los temas por cociente
   gusto por minuto.

2. Mientras el espacio disponible alcance
   para poner el próximo tema, agregarlo y
   actualizar

3. Cuando el próximo tema no entre completo,
   agregarlo fraccionalmente

(si me quedo sin temas, todos entraban en el disco)

CD - Algoritmo

$\Theta(N \lg N)$ 1. ordenar los temas por cociente gusto por minuto.

$\Theta(N)$ 2. Mientras el espacio disponible alcance para poner el próximo tema, agregarlo y actualizar

3. Cuando el próximo tema no entre completo, agregarlo fraccionalmente

(si me quedo sin temas, todos entraban en el disco)

costo total: $\Theta(N \lg N)$

CD - En el problema sin fracción

¿No podría usar esto mismo en la variante sin fracción?

No funciona. Contraejemplo:

$N=2$ | 1 | 10
$P=10$ | 2 | 10

$N=2$	1	10
$P=10$	2	10
	↓ 2	↓ 1

Pone el primer tema y no logra llenar el disco

0

Sin embargo, a veces se puede usar como heurística

CD - Algoritmo greedy heurístico

Consideremos el algoritmo greedy y supongamos que agrego todos los temas hasta el j.

Sea opt el óptimo de nuestro problema y OPT* el óptimo de la versión fraccionaria.

Afirmamos que opt* $\geq$ opt y que son iguales sólo si no hay temas fraccionados.

Luego,

$$opt^* = \sum_{i=1}^{j-1} g_i + \alpha g_j > OPT$$

Proporción del tema j que uso

(Suponemos que todos los temas entran en el disco)

$$\Downarrow$$

$$\sum_{i=1}^{j-1} g_i + g_j > OPT$$

CD - Algoritmo greedy heurístico

Si

$$\sum_{i=1}^{j-1} g_i + g_j > OPT \quad ,$$

entonces

$$\max\left(\sum_{i=1}^{j-1} g_i \,,\; g_j\right) > \frac{OPT}{2}$$

(No es posible que los dos sumandos valgan menos que $\frac{OPT}{2}$ pero sumados superen OPT)

CD - Algoritmo greedy heurístico

Es decir, podemos alterar levemente el algoritmo greedy y obtener una heurística que de manera garantizada da al menos la mitad del óptimo.

(En particular, haciendo estrategia greedy hasta el primer tema que no entre y quedarse con el máximo gusto entre lo acumulado y el del tema que no pude agregar)

Este tipo de algoritmos se conoce como **Algoritmos aproximados.**

Meximización

Dado un conjunto $X \subset \mathbb{N}_0$, definimos

$$\operatorname{mex}(X) = \min\{\, j \mid j \in \mathbb{N}_0 \land j \notin X \,\}$$

Meximización

Dado un conjunto $X \subset \mathbb{N}_0$, definimos

$$mex(X) = \min \{ j \mid j \in \mathbb{N} \wedge j \notin X \}$$

$$mex(\{0,1,3\}) = 2$$

$$mex(\{1,2,3,4\}) = 0$$

Meximización

Dado un vector $A = a_1 \ldots a_n$ de naturales con el 0, encontrar la permutación $B = b_1 \ldots b_n$ de A que maximice

$$\sum_{i=1}^{n} mex(B, i) = \sum_{i=1}^{n} mex(\{b_1, \ldots, b_i\})$$

Meximización - Ejemplo

$$A = [0, 5, 2, 0, 1]$$

$$B^0 = 0, 5, 2, 0, 1$$

$$\text{mex}(B^0, 1) = \text{mex}(\{0\}) = 1$$

$$\text{mex}(B^0, 2) = 1$$

$$\text{mex}(B^0, 3) = 1$$

$$\text{mex}(B^0, 4) = 1$$

$$\text{mex}(B^0, 5) = 3$$

$$\sum \text{mex} = 7$$

$$B^1 = 0, 1, 0, 2, 5$$

$$\text{mex}(B^1, 1) = 1$$

$$\text{mex}(B^1, 2) = 2$$

$$\text{mex}(B^1, 3) = 2$$

$$\text{mex}(B^1, 4) = 3$$

$$\text{mex}(B^1, 5) = 3$$

$$\sum \text{mex} = 11$$

Meximización - Ideas

Notar que si $mex(X,i)=k$, $k \leq i$, pues X a lo sumo es una permutación del conjunto $\{0,\dots,i-1\}$.

Además, para $j>i$, $mex(X,j) \geq k$, pues El valor más grande posible es el primer faltante, todos los elementos que ya estaban seguirán estando.

Meximización - Permutación óptima

Buscamos que $mex(B^*, i)$ sea máximo para todo i

Para el primer elemento voy a querer el 0, si no está, $\sum mex$ va a dar siempre 0

Asimismo, en la segunda posición voy a querer tener un 1, y en la tercera un 2,...

Meximización - Solución greedy

Defino que una solución del problema es greedy si cumple que tiene el valor i en la i-ésima posición (de ser posible)

Por otra parte, es óptima si maximiza

$$\sum_{i=1}^{n} \operatorname{mex}(B, i)$$

Para toda permutación B.

Meximización - greedy ⇒ optima

Sea $A = a_1 \dots a_n$ input, $G = g_1 \dots g_n$ permutación greedy e $Y = y_1 \dots y_n$ una permutación cualquiera. Veamos que

$P(k) \equiv$ Si $k \leq n$, entonces $\sum_{i=1}^{k} \text{mex}(G,i) \geq \sum_{i=1}^{k} \text{mex}(Y,i)$

hacemos inducción.

$P(0) \rightarrow$ Trivial

$P(k) \Rightarrow P(k+1)$. Si $k+1 > n$, la implicación falsea el precedente y vale.

Sea $k < n$. Vemos que $m = \text{mex}(G, k+1) \geq \text{mex}(Y, k+1)$. Si $m = k+1$, esto vale porque $k+1$ es el máximo valor posible para $\text{mex}(Z, k+1)$ (el mínimo que no está no puede ser mayor).

Meximización - greedy ⇒ optima

Si $m \leq k$, entonces $0 \dots m-1$ estān en $\{g_1 \dots g_{k+1}\}$ pero m no. Luego, m no está en G y por ende no está en A ni en Y (pues por ser G greedy, no puede estar después). Por lo tanto,

$mex(Y, i) = \min\{y \in \mathbb{N} \mid y \notin Y\} \leq m = mex(G, k+1)$ para todo i, en particular k, así que $mex(G, k+1) \geq mex(Y, k+1)$. Lo que sigue es inducción, pues

$$\sum_{i=1}^{k+1} mex(G, i) = \sum_{i=1}^{k} mex(G, i) + mex(G, k+1)$$

Meximización - greedy => optima

$$\sum_{i=1}^{k+1} \mathrm{mex}(G,i) = \sum_{i=1}^{k} \mathrm{mex}(G,i) + \mathrm{mex}(G,k+1) \geq \sum_{i=1}^{k} \mathrm{mex}(G,i) + \mathrm{mex}(Y,k+1)$$

$$\overset{HI}{\geq} \sum_{i=1}^{k} \mathrm{mex}(Y,i) + \mathrm{mex}(Y,k+1) = \sum_{i=1}^{k+1} \mathrm{mex}(Y,i)$$

$\blacksquare$

Maximización - Óptimo ⇒ greedy (extra)

Sea $m = \text{mex}(O, n)$ para una permutación óptima $O = O_1, \dots, O_n$ de X y sea G una permutación golosa.

Ciertamente, $\text{mex}(O, i) \leq \min\{i, m\}$ para todo $1 \leq i \leq n$. Luego,

$$\sum_{i=1}^{n} \text{mex}(G, i) \leq \sum_{i=1}^{n} \text{mex}(O, i) \leq \underbrace{\sum_{i=1}^{m} i + m(n-m)}_{= \sum_{i=1}^{n} \min\{i, m\}} = \sum_{i=1}^{n} \text{mex}(G, i)$$

esto es precisamente lo que da la cuenta en una permutación greedy

$\blacksquare$

Meximización - Óptimo ⇒ greedy (extra)

¿No demuestra esto lo que buscamos?

No, porque podrían potencialmente haber soluciones greedy que no son óptimas. No alcanzaría con devolver una solución greedy cualquiera para garantizar el óptimo.

Meximización - Algoritmo

¿Cómo genero la permutación B con $B[i]=i$?

Posibilidad: Buscar cada valor y ubicarlo $(O(n^2))$

¿Se puede mejorar? Sí.

Meximización - Algoritmo

- Voy recorriendo A
- En cada posición i, si su valor $k < n$, swappeo la posición i con la k
- Al final, los primeros elementos forman escalera.
- Costo: $O(n)$