

Patrones en algoritmos golosos avanzados

Agustín Santiago Gutiérrez

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Training Camp 2021

Contenidos

1 Heurísticas

- Heurística “Pasito seguro”
- Heurística “Continua \Rightarrow no hay repetidos”
- Heurística “Conjunto = Ordenar”
- Ejemplo: Cardboard Box

2 Golosos de ordenar

- Cinta magnética
- CIIC 2020: Codefoxes
- Práctica IOI 2019
- Dos máquinas

3 Scheduling de tareas

- Scheduling común (matroide)
- Trabajos en un disco (mundial ICPC recargado)

4 Golosos usando “reemplazo equivalente”

- Máxima suma sin consecutivos
- Super Game

5 Bonus si hay tiempo

- Artem and Array
- Bonus: Cardboard Box para todo k

Thinking to get at once all the gold the goose could give, he killed it and opened it only to find nothing.

Aesop, "The Goose with the Golden Eggs", Aesop's Fables, as translated by Joseph Jacobs (1894)

Greedy folk have long arms.

English proverb. As noted in J. Kelly's Complete Collection of Scottish Proverbs (1721).

Contenidos

1 Heurísticas

- Heurística "Pasito seguro"
- Heurística "Continua ⇒ no hay repetidos"
- Heurística "Conjunto = Ordenar"
- Ejemplo: Cardboard Box

2 Golosos de ordenar

- Cinta magnética
- CIIC 2020: Codefoxes
- Práctica IOI 2019
- Dos máquinas

3 Scheduling de tareas

- Scheduling común (matroide)
- Trabajos en un disco (mundial ICPC recargado)

4 Golosos usando "reemplazo equivalente"

- Máxima suma sin consecutivos
- Super Game

5 Bonus si hay tiempo

- Artem and Array
- Bonus: Cardboard Box para todo k

Explicación de la heurística

- **Casi todas las demostraciones golosas la usan**
- Idea: describir el goloso como secuencia de “pasitos”
- Al comenzar el pasito X , todavía es posible alcanzar solución
- Demostrar que **luego del pasito X** , sigue siendo posible
 - Forma típica: suponer que existe una solución inalcanzable
 - Esa solución se forma haciendo **otro** pasito Y
 - (Razonar acá, magia magia)
 - “Puedo hacer un cambiazo así y así”
 - (Más magia magia)
 - Y tengo ahora otra solución, pero que **sí** hizo el pasito X

Contenidos

1 Heurísticas

- Heurística "Pasito seguro"
- **Heurística "Continua ⇒ no hay repetidos"**
- Heurística "Conjunto = Ordenar"
- Ejemplo: Cardboard Box

2 Golosos de ordenar

- Cinta magnética
- CIIC 2020: Codefoxes
- Práctica IOI 2019
- Dos máquinas

3 Scheduling de tareas

- Scheduling común (matroide)
- Trabajos en un disco (mundial ICPC recargado)

4 Golosos usando "reemplazo equivalente"

- Máxima suma sin consecutivos
- Super Game

5 Bonus si hay tiempo

- Artem and Array
- Bonus: Cardboard Box para todo k

Explicación de la heurística

- La **respuesta es continua** en función de n valores de entrada x_i
- Si hay repetidos, puede molestar a los razonamientos
- $x_i \in \mathbb{R}$ y no asumimos $x_i \in \mathbb{Z}$
- Perturbando δ los x_i , podríamos hacerlos todos distintos
- La respuesta cambia $\epsilon(\delta)$: despreciable
- **Puedo razonar asumiendo que no hay repetidos**
- En el código desempato arbitrariamente (si hace falta)

Contenidos

1

Heurísticas

- Heurística “Pasito seguro”
- Heurística “Continua ⇒ no hay repetidos”
- **Heurística “Conjunto = Ordenar”**
- Ejemplo: Cardboard Box

2

Golosos de ordenar

- Cinta magnética
- CIIC 2020: Codefoxes
- Práctica IOI 2019
- Dos máquinas

3

Scheduling de tareas

- Scheduling común (matroide)
- Trabajos en un disco (mundial ICPC recargado)

4

Golosos usando “reemplazo equivalente”

- Máxima suma sin consecutivos
- Super Game

5

Bonus si hay tiempo

- Artem and Array
- Bonus: Cardboard Box para todo k

Enunciado de la heurística

“Cuando al pensar un problema encuentres frente a ti un **conjunto (o multiconjunto)**, sean lo que sean sus elementos, sabrás que permutar libremente el orden en que los tienes puedes, sin que la respuesta se vea afectada para nada. **Debes ordenarlos, pequeño saltamontes: ordenarlos sabiamente debes.**”

Kwai Chang Caine, monje shaolín



Ejemplo trivial: sumar n números

- Dados n números de entrada, calcular la suma
- Observación: es un **conjunto** de números
- O sea: el orden en que vienen en la entrada no cambia nada
- Nuevo problema: sumar una **lista ordenada** de n números
- No ganamos nada... pero tampoco perdemos

Contenidos

1

Heurísticas

- Heurística “Pasito seguro”
- Heurística “Continua \Rightarrow no hay repetidos”
- Heurística “Conjunto = Ordenar”

Ejemplo: Cardboard Box

2

Golosos de ordenar

- Cinta magnética
- CIIC 2020: Codefoxes
- Práctica IOI 2019
- Dos máquinas

3

Scheduling de tareas

- Scheduling común (matroide)
- Trabajos en un disco (mundial ICPC recargado)

4

Golosos usando “reemplazo equivalente”

- Máxima suma sin consecutivos
- Super Game

5

Bonus si hay tiempo

- Artem and Array
- Bonus: Cardboard Box para todo k

Ejemplo: Cardboard Box (enunciado)

- Usar exactamente k estrellas y obtener máximo beneficio
- n niveles numerados $1 \leq i \leq n$
- El nivel i da beneficio 0 con **cero** estrellas
- El nivel i da beneficio A_i con **una** estrella
- El nivel i da beneficio B_i con **dos** estrellas

Ejemplo: Cardboard Box (observaciones)

- Observación: es un **conjunto** de niveles
 - O sea: el orden en que vienen en la entrada no cambia nada
- Observación: la respuesta es continua en los (A, B)
 - Podemos razonar como si los A, B fueran todos distintos

Ejemplo: Cardboard Box (ordenando por A)

ORDENADO POR A_i DECRECIENTE



ABSURDO!

Ejemplo: Cardboard Box (ordenando por A_i)

ORDENADO POR A_i DECRECIENTE



Ejemplo: Cardboard Box (ordenando por B)

ORDENADO POR B_i : DECRECIENTE



ABSURDO!

Ejemplo: Cardboard Box (ordenando por B)

ORDENADO POR B_i : DECRECIENTE



Contenidos

1 Heurísticas

- Heurística “Pasito seguro”
- Heurística “Continua \Rightarrow no hay repetidos”
- Heurística “Conjunto = Ordenar”
- Ejemplo: Cardboard Box

2 Golosos de ordenar

- Cinta magnética
- CIIC 2020: Codefoxes
- Práctica IOI 2019
- Dos máquinas

3 Scheduling de tareas

- Scheduling común (matroide)
- Trabajos en un disco (mundial ICPC recargado)

4 Golosos usando “reemplazo equivalente”

- Máxima suma sin consecutivos
- Super Game

5 Bonus si hay tiempo

- Artem and Array
- Bonus: Cardboard Box para todo k

¿Qué es una cinta magnética?

Magnetic tape is a medium for [magnetic recording](#), made of a thin, magnetizable coating on a long, narrow strip of [plastic film](#). It was developed in Germany in 1928, based on [magnetic wire recording](#). Devices that record and playback audio and video using magnetic tape are [tape recorders](#) and [video tape recorders](#) respectively. A device that stores computer data on magnetic tape is known as a [tape drive](#).

Magnetic tape revolutionized [sound recording and reproduction](#) and [broadcasting](#). It allowed [radio](#), which had always been broadcast live, to be recorded for later or repeated airing. It allowed [gramophone records](#) to be recorded in multiple parts, which were then mixed and edited with tolerable loss in quality. It was a key technology in early computer development, allowing unparalleled amounts of data to be mechanically created, stored for long periods, and rapidly accessed. The [Videotape recorder](#) which used magnetic tape allowed TV stations to gather news, timeshift and record content without having to use or develop relatively expensive and single-use film stock while allowing for the tape to be reused.



7-inch reel of 1/4-inch-wide audio recording tape, typical of consumer use in the 1950s–70s



Problema

- Guardamos n programas en una cinta magnética
- El i -ésimo ocupa S_i bytes
- El i -ésimo se carga f_i veces por día
- Si los ponemos en orden i_1, i_2, \dots, i_n (permutación)
- Costo: $\sum_{k=1}^n (\sum_{j=1}^k S_{i_j}) \cdot f_{i_k}$

$$N = 6$$



$$S_{i_1} + S_{i_2} + S_{i_3} + S_{i_4}$$

Heurística: intercambio de consecutivos

- Idea clave: analizar el caso $N = 2$
 - ¡Solo dos permutaciones!
 - Calcularlas y compararlas
- Más en general, considerar dos elementos consecutivos
 - Calcular qué pasa al intercambiarlos
 - Ver si el costo cambia de manera predecible



vs



Heurística: intercambio de consecutivos

$$C_A - C_B = \cancel{S_1 f_1} + (S_1 + S_2) f_2 - \brace{- S_2 f_2 - (S_2 + S_1) f_1}$$

$$C_A - C_B = S_1 f_2 - S_2 f_1$$

luego $C_B > C_A$ si y solo si:

$$0 > S_1 f_2 - S_2 f_1$$

$$\frac{S_1}{f_1} < \frac{S_2}{f_2}$$

Heurística: intercambio de consecutivos

$\star = \text{COSTO NO CAMBIA}$

$$S_{izq} = \sum_{j=1}^{k-1} s_{ij}$$

A



B



$C_B - C_A$ ES IGUAL QUE ANTES

$S_{izq} \cdot (f_{i_k} + f_{i_{k+1}})$ ES UN COSTO FIJO EN A Y B

Contenidos

1 Heurísticas

- Heurística “Pasito seguro”
- Heurística “Continua \Rightarrow no hay repetidos”
- Heurística “Conjunto = Ordenar”
- Ejemplo: Cardboard Box

2 Golosos de ordenar

- Cinta magnética
- **CIIC 2020: Codefoxes**
- Práctica IOI 2019
- Dos máquinas

3 Scheduling de tareas

- Scheduling común (matroide)
- Trabajos en un disco (mundial ICPC recargado)

4 Golosos usando “reemplazo equivalente”

- Máxima suma sin consecutivos
- Super Game

5 Bonus si hay tiempo

- Artem and Array
- Bonus: Cardboard Box para todo k

CIIC 2020: Codefoxes

- <https://omegaup.com/arena/ciic-2020-Publico/practice#problems/ciic2020-codefoxes>

Descripción del problema

Túrist es un exitoso competidor en el sitio web de competencias de programación Codefoxes. Guepardo es otro competidor también muy exitoso, aunque no tanto como Túrist. Necesita tu ayuda para obtener el mayor puntaje posible en las competencias de Codefoxes.

En las reglas de Codefoxes, se conocen N problemas al comenzar la prueba. Cada problema tiene un cierto puntaje P_i , y un cierto factor F_i de reducción del puntaje. Por cada segundo de tiempo que pasa desde el comienzo de la prueba, el puntaje del problema se reduce en F_i puntos. Por lo tanto, si el problema se envía correctamente a los t segundos de empezada la competencia, se obtienen por él $\max(0, P_i - t \times F_i)$ (nunca puede restar puntos enviar un problema).

Al ser uno de los mejores competidores del mundo, Guepardo siempre resuelve correctamente los problemas al primer intento, y además lee todos los enunciados al comenzar la prueba en un tiempo despreciable. Una vez leídos, sabe exactamente el tiempo T_i que tardará en programar y enviar cada problema, en segundos.

Debes escribir un programa que dados los valores P_i , F_i y T_i de cada uno de los N problemas, así como la duración total T de la prueba en segundos: le indique a Guepardo cuál es el máximo puntaje que puede obtener, si elige adecuadamente qué problemas resolver y en qué orden.

Contenidos

1 Heurísticas

- Heurística “Pasito seguro”
- Heurística “Continua \Rightarrow no hay repetidos”
- Heurística “Conjunto = Ordenar”
- Ejemplo: Cardboard Box

2 Golosos de ordenar

- Cinta magnética
- CIIC 2020: Codefoxes
- Práctica IOI 2019
- Dos máquinas

3 Scheduling de tareas

- Scheduling común (matroide)
- Trabajos en un disco (mundial ICPC recargado)

4 Golosos usando “reemplazo equivalente”

- Máxima suma sin consecutivos
- Super Game

5 Bonus si hay tiempo

- Artem and Array
- Bonus: Cardboard Box para todo k

Práctica IOI 2019



31ST INTERNATIONAL
OLYMPIAD IN INFORMATICS
4-11 AUGUST 2019
BAKU • AZERBAIJAN

job
Practice Tasks
English (EN)

Job Scheduling

The "Land of Fire" is famous for its "Temple of Fire" - Ateshgah. To accomodate more visitors, as a master architect, you are planning to build more temples. You have one builder and n temples to build. Temples are numbered from 0 to $n - 1$. According to the plan, each temple i has a prerequisite temple $p[i]$ that should be built before temple i . Only temple 0 has $p[0] = -1$, which means this temple can be built right away at time 0. Temple i takes $d[i]$ seconds to build, and finishing it at time t costs $t * u[i]$.

Find the minimum cost to build all the temples.

Práctica IOI 2019: observaciones

- Es el problema de la cinta, pero solo algunas permutaciones valen
- Debe ser un orden topológico del árbol
- Es continua, podemos asumir todos distintos
- Idea 0: Si el de mínimo $\frac{d_i}{u_i}$ se puede usar, es **pasito seguro**

Práctica IOI 2019: observaciones

- Es el problema de la cinta, pero solo algunas permutaciones valen
- Debe ser un orden topológico del árbol
- Es continua, podemos asumir todos distintos
- Idea 0: Si el de mínimo $\frac{d_i}{u_i}$ se puede usar, es **pasito seguro**
- Idea 1: si no se puede usar, fusionarlo con su padre
- Repetir eso n veces produce la respuesta
- Implementación delicada $O(n \lg n)$
 - Cola de prioridad o multiset global
 - Small to large / union find

Práctica IOI 2019: observaciones

- Es el problema de la cinta, pero solo algunas permutaciones valen
- Debe ser un orden topológico del árbol
- Es continua, podemos asumir todos distintos
- Idea 0: Si el de mínimo $\frac{d_i}{u_i}$ se puede usar, es **pasito seguro**
- Idea 1: si no se puede usar, fusionarlo con su padre
- Repetir eso n veces produce la respuesta
- Implementación delicada $O(n \lg n)$
 - Cola de prioridad o multiset global
 - Small to large / union find
- Curiosidad: sea X , mínimo en el subárbol de su padre
 - Fusionar con el padre es **pasito seguro**
 - Se demuestra imaginando una ejecución del algoritmo anterior

Patrones generales

- El problema anterior ilustra algunos patrones usuales
- Analizar qué pasa al poner elementos o ejemplos **extremos**
- Principio “máximo global / máximo local”
 - Vale elección golosa con el máximo global (1)
 - Luego usando (1), se demuestra que vale elección golosa local
 - A veces esto permite un algoritmo más fácil de programar
 - A veces permite bajar la complejidad temporal

Contenidos

1

Heurísticas

- Heurística “Pasito seguro”
- Heurística “Continua \Rightarrow no hay repetidos”
- Heurística “Conjunto = Ordenar”
- Ejemplo: Cardboard Box

2

Golosos de ordenar

- Cinta magnética
- CIIC 2020: Codefoxes
- Práctica IOI 2019

● Dos máquinas

3

Scheduling de tareas

- Scheduling común (matroide)
- Trabajos en un disco (mundial ICPC recargado)

4

Golosos usando “reemplazo equivalente”

- Máxima suma sin consecutivos
- Super Game

5

Bonus si hay tiempo

- Artem and Array
- Bonus: Cardboard Box para todo k

Dos máquinas

- Hay n remeras
- La i -ésima toma tiempo A_i para lavar
- La i -ésima toma tiempo B_i para secar
- Cada remera hay que lavarla, y **luego** secarla
- Hay una máquina lavadora y una secadora
- Pueden trabajar ambas máquinas a la vez
- Cada máquina procesa una remera por vez
- Dar el mínimo tiempo posible para terminar todo

Dos máquinas (cont.)

- Elegir el orden en que se lavan determina todo
- Probamos $N = 2$
- Lavar la 1 y luego la 2 toma $A_1 + A_2 + \max(0, B_1 - A_2) + B_2$

Dos máquinas (cont.)

- Elegir el orden en que se lavan determina todo
- Probamos $N = 2$
- Lavar la 1 y luego la 2 toma $A_1 + A_2 + \max(0, B_1 - A_2) + B_2$
- Al revés toma $A_1 + A_2 + \max(0, B_2 - A_1) + B_1$
- $C_B - C_A = \max(0, B_2 - A_1) + B_1 - \max(0, B_1 - A_2) - B_2$
- 1 antes que 2 es bueno, si $C_B - C_A > 0$

Dos máquinas (cont.)

- Elegir el orden en que se lavan determina todo
- Probamos $N = 2$
- Lavar la 1 y luego la 2 toma $A_1 + A_2 + \max(0, B_1 - A_2) + B_2$
- Al revés toma $A_1 + A_2 + \max(0, B_2 - A_1) + B_1$
- $C_B - C_A = \max(0, B_2 - A_1) + B_1 - \max(0, B_1 - A_2) - B_2$
- 1 antes que 2 es bueno, si $C_B - C_A > 0$
- 4 casos:
 - Si $B_2 - A_1 > 0$ y $B_1 - A_2 > 0$: $C_B - C_A = A_2 - A_1$, quiero $A_2 > A_1$
 - Si $B_2 - A_1 < 0$ y $B_1 - A_2 > 0$: $C_B - C_A = A_2 - B_2$, quiero $A_2 > B_2$
 - Si $B_2 - A_1 > 0$ y $B_1 - A_2 < 0$: $C_B - C_A = B_1 - A_1$, quiero $B_1 > A_1$
 - Si $B_2 - A_1 < 0$ y $B_1 - A_2 < 0$: $C_B - C_A = B_1 - B_2$, quiero $B_1 > B_2$
- Buscamos un criterio para ordenar, que garantice lo anterior

Dos máquinas (cont.)

- Vimos que $A_i > B_i$ y $A_i < B_i$ son condiciones importantes
- Si $A_i < B_i$, decimos que la remera i es tipo 1
- Si $A_i > B_i$, decimos que la remera i es tipo 2
- Veamos qué pasa si son ambas tipo 2

Dos máquinas (cont.)

- Vimos que $A_i > B_i$ y $A_i < B_i$ son condiciones importantes
- Si $A_i < B_i$, decimos que la remera i es tipo 1
- Si $A_i > B_i$, decimos que la remera i es tipo 2
- Veamos qué pasa si son ambas tipo 2
- Observación: el caso 1 nunca ocurre al comparar dos tipo 2

Dos máquinas (cont.)

- Vimos que $A_i > B_i$ y $A_i < B_i$ son condiciones importantes
- Si $A_i < B_i$, decimos que la remera i es tipo 1
- Si $A_i > B_i$, decimos que la remera i es tipo 2
- Veamos qué pasa si son ambas tipo 2
- Observación: el caso 1 nunca ocurre al comparar dos tipo 2
- Observación: el caso 2 no puede fallar al comparar dos tipo 2

Dos máquinas (cont.)

- Vimos que $A_i > B_i$ y $A_i < B_i$ son condiciones importantes
- Si $A_i < B_i$, decimos que la remera i es tipo 1
- Si $A_i > B_i$, decimos que la remera i es tipo 2
- Veamos qué pasa si son ambas tipo 2
- Observación: el caso 1 nunca ocurre al comparar dos tipo 2
- Observación: el caso 2 no puede fallar al comparar dos tipo 2
- Observación: el caso 4 no puede fallar si ponemos $B_1 > B_2$
- Observación: el caso 3 nunca ocurre si ponemos $B_1 > B_2$

Dos máquinas (cont.)

- Vimos que $A_i > B_i$ y $A_i < B_i$ son condiciones importantes
- Si $A_i < B_i$, decimos que la remera i es tipo 1
- Si $A_i > B_i$, decimos que la remera i es tipo 2
- Veamos qué pasa si son ambas tipo 2
- Observación: el caso 1 nunca ocurre al comparar dos tipo 2
- Observación: el caso 2 no puede fallar al comparar dos tipo 2
- Observación: el caso 4 no puede fallar si ponemos $B_1 > B_2$
- Observación: el caso 3 nunca ocurre si ponemos $B_1 > B_2$
- Conclusión: **si ambas son tipo 2 queremos $B_1 > B_2$**

Dos máquinas (cont.)

- De forma análoga vemos que:
- **Si ambas son tipo 1 queremos $A_1 < A_2$**
- **Si una es tipo 1 y la otra tipo 2, queremos primero la tipo 1**
- Este ordenamiento es óptimo incluso con $N > 2$
- Se puede demostrar reduciéndolo a lo que ya hicimos
- Solución $O(N \lg N)$:
 - Ordenar con este criterio
 - Simular eficientemente
- Por continuidad, es correcto asignar las $A_i = B_i$ a cualquier tipo

Contenidos

1 Heurísticas

- Heurística “Pasito seguro”
- Heurística “Continua \Rightarrow no hay repetidos”
- Heurística “Conjunto = Ordenar”
- Ejemplo: Cardboard Box

2 Golosos de ordenar

- Cinta magnética
- CIIC 2020: Codefoxes
- Práctica IOI 2019
- Dos máquinas

3 Scheduling de tareas

- **Scheduling común (matroide)**
- Trabajos en un disco (mundial ICPC recargado)

4 Golosos usando “reemplazo equivalente”

- Máxima suma sin consecutivos
- Super Game

5 Bonus si hay tiempo

- Artem and Array
- Bonus: Cardboard Box para todo k

Problema

- Se deben realizar n trabajos
- Todos toman una unidad de tiempo
- Cada uno tiene un deadline en el que debe estar terminado
- ¿Es posible cumplir todos?

Problema

- Se deben realizar n trabajos
- Todos toman una unidad de tiempo
- Cada uno tiene un deadline en el que debe estar terminado
- ¿Es posible cumplir todos?
 - Ordenamos por deadline, y simulamos

Condición mágica

- Para cada $t \geq 0$, llamo c_t a la cantidad de trabajos con deadline t
- Sea $C_t = \sum_{i=0}^t c_t$ (cantidad con deadline hasta t)
- (1) Claramente si se puede cumplir todo, $C_t \leq t$ para todo $t \geq 0$

Condición mágica

- Para cada $t \geq 0$, llamo c_t a la cantidad de trabajos con deadline t
- Sea $C_t = \sum_{i=0}^t c_t$ (cantidad con deadline hasta t)
- (1) Claramente si se puede cumplir todo, $C_t \leq t$ para todo $t \geq 0$
 - Observación: por el goloso anterior, (1) es condición suficiente

¡Matroide!

- Sea A un conjunto de trabajos posibles
- Un $S \subseteq A$ es independiente, si cumple la condición mágica
- Estos conjuntos independientes forman un matroide

Máxima cantidad de trabajos realizables

- Como es un matroide, basta correr el mismo goloso de antes
- Cuando fallamos con un elemento, lo descartamos y seguimos

Máximo beneficio conseguible

- Cada trabajo tiene ahora también un beneficio
- Cumplir subconjunto de trabajos con máximo beneficio total
- Como es un matroide, recorremos en orden de beneficio:
 - Si agregarlo preserva la condición mágica, lo agregamos
 - Si no, lo descartamos
 - Notar que ya **no** recorremos en orden por deadline
- Verificar la condición mágica eficientemente permite $O(n \lg n)$

Idea para problemsetters

- Al ser un matroide, se podrían plantear variantes más avanzadas
- Problema con dos procesadores (suma / intersección de matroides)
- Se pueden resolver en tiempo polinomial con algoritmos de matroides

Contenidos

1

Heurísticas

- Heurística “Pasito seguro”
- Heurística “Continua \Rightarrow no hay repetidos”
- Heurística “Conjunto = Ordenar”
- Ejemplo: Cardboard Box

2

Golosos de ordenar

- Cinta magnética
- CIIC 2020: Codefoxes
- Práctica IOI 2019
- Dos máquinas

3

Scheduling de tareas

- Scheduling común (matroide)
- Trabajos en un disco (mundial ICPC recargado)

4

Golosos usando “reemplazo equivalente”

- Máxima suma sin consecutivos
- Super Game

5

Bonus si hay tiempo

- Artem and Array
- Bonus: Cardboard Box para todo k

Enunciado

- Hay n trabajos sobre un disco rígido
- Inicialmente, el disco tiene E de espacio libre
- Un trabajo solamente se puede ejecutar si queda al menos e_i libre
- Luego de ejecutar un trabajo, la cantidad de espacio libre aumenta en d_i
 - $d_i > 0$ en los trabajos que terminan liberando espacio
 - $d_i = 0$ en los trabajos que no alteran el espacio libre
 - $d_i < 0$ en los trabajos que terminan ocupando más espacio

Verificar si todos se pueden ejecutar

- Ejecutar un trabajo con $d_i \geq 0$ (si se puede) es pasito seguro
- Si no se puede ejecutar ningún $d_i \geq 0$, es claro que nunca se podrá
- ¿En qué orden ir ejecutando los trabajos con $d_i < 0$?

Verificar si todos se pueden ejecutar

- Ejecutar un trabajo con $d_i \geq 0$ (si se puede) es pasito seguro
- Si no se puede ejecutar ningún $d_i \geq 0$, es claro que nunca se podrá
- ¿En qué orden ir ejecutando los trabajos con $d_i < 0$?
 - Solución: en orden decreciente de $e_i + d_i$

Interpretación como scheduling

- Si identificamos:
 - $-d_i$ con el tiempo que toma un trabajo
 - $-e_i$ con el tiempo de inicio máximo permitido de un trabajo
- El problema es equivalente al de scheduling
- Estamos ordenando crecientemente por deadline ($-e_i - d_i$)

Este caso no es matroide

- Ya no todos los trabajos tardan lo mismo
- Deja de ser matroide

Completar la máxima cantidad de trabajos

- Podemos recorrer en el mismo orden, por deadline
- Si un trabajo encaja, lo agregamos
- Si no encaja, no lo descartamos directamente:
 - Lo agregamos al conjunto solución
 - Ahora los trabajos elegidos suman más que este deadline
 - Sacamos al trabajo **más largo** que haya

Contenidos

1 Heurísticas

- Heurística “Pasito seguro”
- Heurística “Continua \Rightarrow no hay repetidos”
- Heurística “Conjunto = Ordenar”
- Ejemplo: Cardboard Box

2 Golosos de ordenar

- Cinta magnética
- CIIC 2020: Codefoxes
- Práctica IOI 2019
- Dos máquinas

3 Scheduling de tareas

- Scheduling común (matroide)
- Trabajos en un disco (mundial ICPC recargado)

4 Golosos usando “reemplazo equivalente”

- Máxima suma sin consecutivos
- Super Game

5 Bonus si hay tiempo

- Artem and Array
- Bonus: Cardboard Box para todo k

Máxima suma sin consecutivos, tranqui

- Se tiene un arreglo de N elementos enteros
- Se eligen exactamente k
- Se eligen maximizando la suma
- Está prohibido elegir dos elementos consecutivos
- Clásico DP $O(Nk)$, u $O(N)$ sin restricción de k

Máxima suma sin consecutivos, hard

- Ahora lo queremos en $O(N \lg N)$
- Queremos la respuesta para **todos** los k
- Es continua, no nos preocupamos por empates
- Veamos un caso extremo:
 - Con $k = 1$ es simplemente elegir el máximo
 - Si máximo es gigantesco, elegirlo es pasito seguro (salvo $k = \frac{n+1}{2}$)
 - ¿Por qué no elegiríamos al máximo?

Posibilidades sin sentido

Cambiar otro elemento por el máximo M , mejoraría

$\text{b} = \text{NO ELEGIDO}$

$\text{M} = \text{ELEGIDO}$

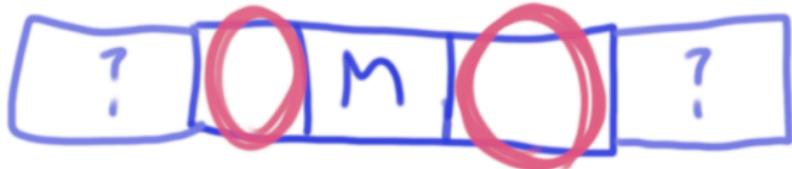
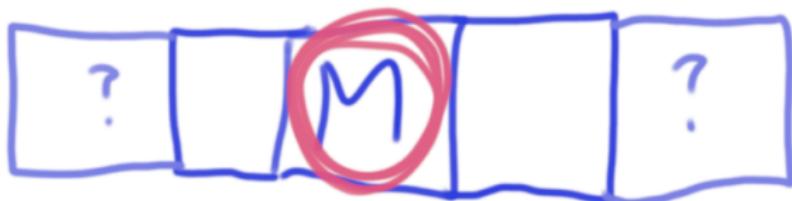


Posibilidades con sentido

Unicas dos opciones si M no está en un borde del arreglo

$\text{m} = \text{NO ELEGIDO}$

$\text{M} = \text{ELEGIDO}$



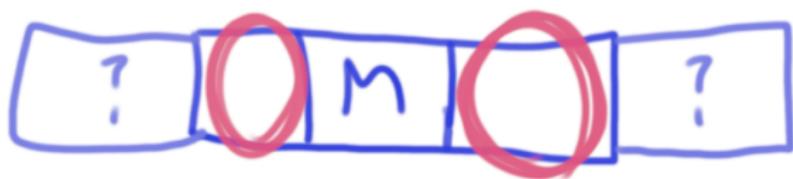
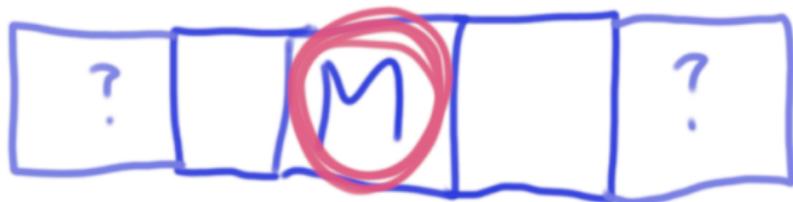
Idea: reemplazo por elemento equivalente

Miremos fuerte estas dos posibilidades.

¿Qué opciones dejan para el futuro?

$\text{b} = \text{NO ELEGIDO}$

$\text{M} = \text{ELEGIDO}$

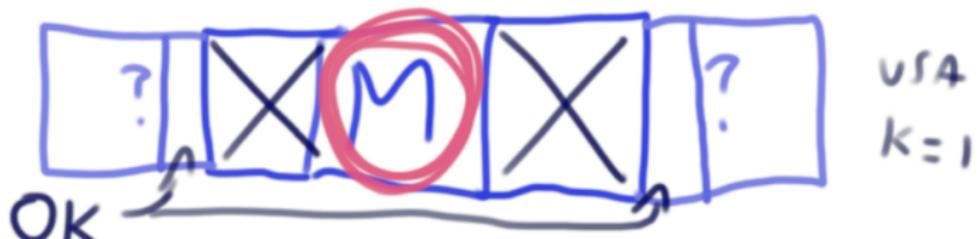


Idea: reemplazo por elemento equivalente

Opciones:

$\text{---} = \text{NO ELEGIDO}$

$\text{M} = \text{ELEGIDO}$

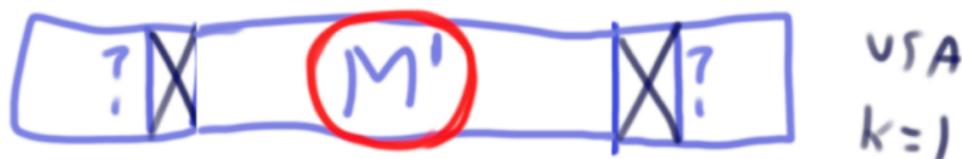


Idea: reemplazo por elemento equivalente

Las anteriores **equivalentes** a la siguiente situación, con $M' = a - M + b$:

$\text{b} = \text{NO ELEGIDO}$

$\text{M} = \text{ELEGIDO}$



"YA GASTÉ" $k=1$

"YA GANÉ" M

Algoritmo

- Inicializar acumulador a 0 (solución $k = 0$)
- En cada paso:
 - Buscar el máximo
 - Sumarlo al acumulador (se obtiene solución para el siguiente k)
 - Si está en un borde, borrarlo junto a su vecino y listo
 - Si no, borrarlo con sus vecinos, y dejar allí $a - M + b$
- Implementación simple con dos sets: $O(N \lg N)$

Contenidos

1 Heurísticas

- Heurística “Pasito seguro”
- Heurística “Continua \Rightarrow no hay repetidos”
- Heurística “Conjunto = Ordenar”
- Ejemplo: Cardboard Box

2 Golosos de ordenar

- Cinta magnética
- CIIC 2020: Codefoxes
- Práctica IOI 2019
- Dos máquinas

3 Scheduling de tareas

- Scheduling común (matroide)
- Trabajos en un disco (mundial ICPC recargado)

4 Golosos usando “reemplazo equivalente”

- Máxima suma sin consecutivos
- Super Game

5 Bonus si hay tiempo

- Artem and Array
- Bonus: Cardboard Box para todo k

Enunciado

- Dos jugadores sacan números de los extremos de un arreglo
- Un jugador va restando los que agarra al total
- El otro va sumando los que agarra al total
- El que suma busca maximizar, y el que resta minimizar
- ¿Cómo jugar de forma óptima?
- Es continua, podemos asumir que no hay repetidos

Teorema

- Si el máximo está en un borde, elegirlo es pasito seguro



Teorema 2

- Si no está en un borde, se puede reemplazar $aMb \rightarrow (a - M + b)$
- Demostración: por el teorema anterior, no bien se agarre a o b , el otro agarra M , que queda expuesto en un borde
- $M > a$ y $M > b$, luego ambos prefieren que el otro juegue primero
- Por lo tanto si vas a tomar a , el rival toma M , y ahora jugar en la otra punta, para eso mejor jugar en la otra punta directamente, y con suerte el rival vendrá a jugar primero acá. Y si terminás jugando a más tarde, no perdiste nada.

Algoritmo

- Estos teoremas ya permiten un algoritmo $O(N \lg N)$
- Se pueden usar los anteriores y un análisis con cuidado para demostrar:
- Teorema local: si $a < b > c$, se puede reemplazar localmente abc por $(a - b + c)$
- Permite un algoritmo $O(N)$, y mucho más fácil de programar
- Más difícil de demostrar

Referencia

- <https://www.mimuw.edu.pl/~idziaszek/termity/termity.pdf>

Contenidos

1 Heurísticas

- Heurística “Pasito seguro”
- Heurística “Continua \Rightarrow no hay repetidos”
- Heurística “Conjunto = Ordenar”
- Ejemplo: Cardboard Box

2 Golosos de ordenar

- Cinta magnética
- CIIC 2020: Codefoxes
- Práctica IOI 2019
- Dos máquinas

3 Scheduling de tareas

- Scheduling común (matroide)
- Trabajos en un disco (mundial ICPC recargado)

4 Golosos usando “reemplazo equivalente”

- Máxima suma sin consecutivos
- Super Game

5 Bonus si hay tiempo

- Artem and Array
- Bonus: Cardboard Box para todo k

Contenidos

1 Heurísticas

- Heurística “Pasito seguro”
- Heurística “Continua \Rightarrow no hay repetidos”
- Heurística “Conjunto = Ordenar”
- Ejemplo: Cardboard Box

2 Golosos de ordenar

- Cinta magnética
- CIIC 2020: Codefoxes
- Práctica IOI 2019
- Dos máquinas

3 Scheduling de tareas

- Scheduling común (matroide)
- Trabajos en un disco (mundial ICPC recargado)

4 Golosos usando “reemplazo equivalente”

- Máxima suma sin consecutivos
- Super Game

5 Bonus si hay tiempo

- Artem and Array
- Bonus: Cardboard Box para todo k