

Greedy

Inicial

Federico Nahuel Quijada

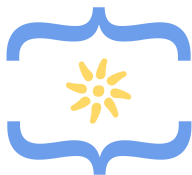
Universidad Tecnológica Nacional – Facultad Regional Santa Fe

Training Camp 2026



Gracias Sponsors!

Organiza



Facultad de
INFORMÁTICA



UNIVERSIDAD
NACIONAL
DE LA PLATA

Diamond Sponsor



- ➊ Algoritmos Greedy
- ➋ ¿Cómo identificarlos?
- ➌ ¿Cómo (intentar) demostrar un Greedy?
- ➍ Algunos casos que fallan
- ➎ Greedys Clásicos
- ➏ Conclusión

Competidor



Coach



Outline

- ➊ Algoritmos Greedy
- ➋ ¿Cómo identificarlos?
- ➌ ¿Cómo (intentar) demostrar un Greedy?
- ➍ Algunos casos que fallan
- ➎ Greedys Clásicos
- ➏ Conclusión

¿Qué es un algoritmo Greedy?

Idea

Un algoritmo **greedy** toma, en cada paso, la decisión que parece ser la mejor utilizando únicamente la información disponible en ese momento.

- No reconsidera decisiones tomadas anteriormente.
- Espera que una secuencia de decisiones óptimas locales produzca una solución óptima global.

Ejemplos

- Elegir el intervalo que termina primero.
- Comprar siempre el objeto más barato.
- Agregar la arista de menor peso disponible.

Ventajas y desventajas

Ventajas

- La idea suele estar a la vista.
- En general, es facil de programar.
- Se utiliza como "parte" de muchos problemas dificiles

Ejemplos

- Es facil creerse un greedy erroneo.
- Probar completamente un claim greedy suele ser dificil.

Outline

- ① Algoritmos Greedy
- ② ¿Cómo identificarlos?
- ③ ¿Cómo (intentar) demostrar un Greedy?
- ④ Algunos casos que fallan
- ⑤ Greedys Clásicos
- ⑥ Conclusión

¿Cómo detectar un posible Greedy?

Muchas veces no es evidente si un problema admite una solución greedy.

Una buena estrategia es hacerse algunas preguntas:

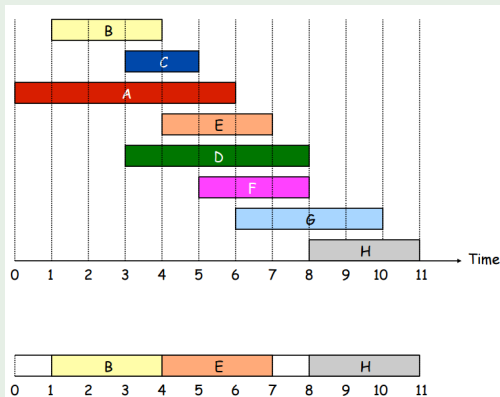
- ¿Me sirve ordenar los elementos?
- ¿Qué ocurre si siempre elijo el más chico?
- ¿Y si elijo el más grande?
- ¿Qué pasa si tomo el que termina primero?
- ¿O el que cuesta menos?
- ¿Puedo demostrar que esta elección nunca es peor que otra?

Importante

No existe una receta para descubrir un greedy. Estas preguntas ayudan, pero la práctica y la demostración son fundamentales.

Example (Programación de intervalos)

Dada una lista de intervalos $[L_i, R_i]$, queremos seleccionar la mayor cantidad de intervalos que no se superpongan.



Example (Programación de intervalos)

Idea greedy

- Ordenar los intervalos por tiempo de finalización.
- Mientras haya intervalos disponibles, elegir el que termina primero y eliminar los que se superponen con él.

¿Por qué funciona?

- Elegir el intervalo que termina antes deja la mayor cantidad posible de tiempo libre para seleccionar intervalos en el futuro.
- Si dos intervalos terminan en el mismo instante, cualquiera de ellos deja exactamente el mismo espacio hacia adelante.
- Por lo tanto, esta elección nunca puede ser peor que cualquier otra disponible.

Outline

- ① Algoritmos Greedy
- ② ¿Cómo identificarlos?
- ③ ¿Cómo (intentar) demostrar un Greedy?
- ④ Algunos casos que fallan
- ⑤ Greedys Clásicos
- ⑥ Conclusión

¿Cómo demostrar un Greedy?

Encontrar una estrategia greedy no alcanza.

Debemos demostrar que **siempre** produce una solución óptima.

Las técnicas de demostración más comunes son:

- Exchange Argument
- Greedy Choice Property
- Staying Ahead

Exchange Argument

Idea

Tomamos una solución óptima cualquiera.

Si esa solución no toma la decisión greedy, demostramos que podemos **intercambiar** algunos elementos para obtener otra solución igual de buena que sí la toma.

Repetimos el argumento hasta transformar completamente la solución óptima en la solución greedy.

Greedy Choice Property

Idea

Existe al menos una solución óptima que comienza realizando la decisión greedy.

Una vez fijada esa primera decisión, el resto del problema vuelve a ser el mismo problema, pero más pequeño.

Esto permite aplicar el mismo razonamiento de manera recursiva.

Idea

Comparar paso a paso la solución greedy con cualquier otra solución. Demostrar que, en cada instante, la solución greedy nunca está en peor situación.

Si nunca queda por detrás, tampoco podrá terminar siendo peor.

Outline

- ① Algoritmos Greedy
- ② ¿Cómo identificarlos?
- ③ ¿Cómo (intentar) demostrar un Greedy?
- ④ Algunos casos que fallan
- ⑤ Greedys Clásicos
- ⑥ Conclusión

El Greedy no siempre funciona

Problema

Disponemos de monedas con valores:

$$\{1, 3, 4\}$$

Queremos formar el valor:

6

utilizando la menor cantidad de monedas posible.

El Greedy no siempre funciona

Problema

Disponemos de monedas con valores:

$$\{1, 3, 4\}$$

Queremos formar el valor:

$$6$$

utilizando la menor cantidad de monedas posible.

Idea Greedy

Siempre tomar la moneda más grande posible.

$$4 + 1 + 1$$

Total: **3 monedas**

Solución óptima

$$3 + 3$$

Total: **2 monedas**

El Greedy no siempre funciona

Teorema de la fruta golosa

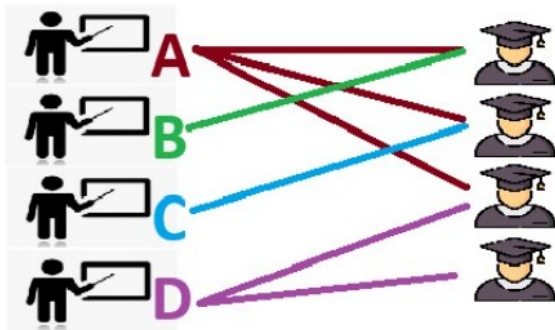
Toda persona que conoce el concepto de algoritmo goloso, en algún momento, propone una solución greedy para un problema que en realidad se resuelve con programación dinámica.

— *Taravilse, Gutiérrez, Piñero et al.*

El Greedy no siempre funciona

Ejemplo

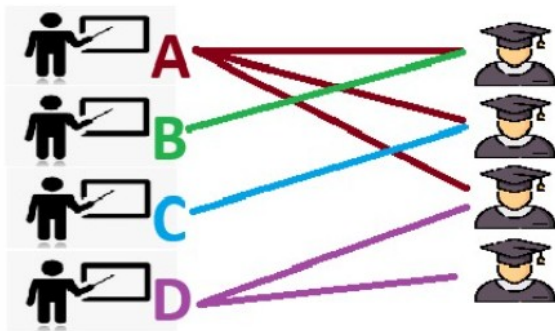
¿Qué estrategia greedy utilizarían para resolver este problema?



El Greedy no siempre funciona

Ejemplo

¿Qué estrategia greedy utilizarían para resolver este problema?



¿Realmente alcanza un greedy?

En muchos problemas de asignación, una decisión local puede impedir encontrar la mejor solución global.

Outline

- ① Algoritmos Greedy
- ② ¿Cómo identificarlos?
- ③ ¿Cómo (intentar) demostrar un Greedy?
- ④ Algunos casos que fallan
- ⑤ Greedys Clásicos
- ⑥ Conclusión

Problema: Ferris Wheel (CSES)

Enunciado

- Hay n personas con pesos w_i .
- Cada góndola soporta como máximo x kilos.
- En cada góndola pueden viajar una o dos personas.

Objetivo

Minimizar la cantidad de góndolas utilizadas.

¿Conviene emparejar personas al azar?

¿Conviene emparejar personas al azar?

No. Por ejemplo, si $X = 6$, y tenemos dos personas con peso 5, entonces, no vamos a poder emparejarlos.

¿Qué hacemos con la persona más pesada?

¿Conviene emparejar personas al azar?

No. Por ejemplo, si $X = 6$, y tenemos dos personas con peso 5, entonces, no vamos a poder emparejarlos.

¿Qué hacemos con la persona más pesada?

- Si puede viajar con alguien...
- Conviene que sea con la de menor peso.

Idea Greedy

- 1 Ordenar los pesos.
- 2 Mantener dos punteros.
- 3 Intentar emparejar el más liviano con el más pesado.

1 2 3 5 6 8



Implementación

C++

```
sort(v.begin(), v.end());

int l = 0, r = n-1;
int ans = 0;

while(l <= r){
    if(v[l] + v[r] <= x)
        l++;
    r--;
    ans++;
}
```

Complejidad:

$$O(n \log n)$$

Problema: Fractional Knapsack

Disponemos de una mochila de capacidad W .

Cada objeto posee

- peso
- valor

A diferencia del problema clásico,

podemos partir los objetos.

Objetivo:

Maximizar el valor total.

¿Qué harían?

Supongamos

Objeto	Peso	Valor
A	10	60
B	20	100
C	30	120

Capacidad:

$$W = 50$$

¿Qué harían?

Supongamos

Objeto	Peso	Valor
A	10	60
B	20	100
C	30	120

Capacidad:

$$W = 50$$

¿Qué criterio elegirían?

Ordenar por

$$\frac{\text{valor}}{\text{peso}}$$

de mayor a menor.

Mientras haya espacio:

- Tomar el objeto completo.
- Si no entra, tomar únicamente la fracción restante.

Implementación

C++

```
sort(items.begin(), items.end(),  
[] (const Item &a, const Item &b){  
    return a.value * b.weight > b.value * a.weight;  
});  
  
for(auto x: items){  
    if(cap >= x.weight){  
        ans += x.value;  
        cap -= x.weight;  
    }else{  
        ans += cap *  
            x.value / x.weight;  
        break;  
    }  
}
```

Complejidad:

Ejemplo: Un Greedy menos evidente

Dado un arreglo cuya suma total es al menos X .

Antes de comenzar, podemos realizar **un único swap** entre dos posiciones.

Luego queremos encontrar el prefijo más corto cuya suma sea al menos X .

Pregunta

¿Qué intercambio conviene realizar?

Supongamos que estamos evaluando un prefijo fijo.

Si realizamos un swap:

- Nunca conviene sacar del prefijo un elemento grande.
- Nunca conviene ingresar al prefijo un elemento pequeño.

Por lo tanto, el mejor intercambio posible es:

mínimo del prefijo $\longleftrightarrow$ máximo del sufijo

siempre que el elemento del sufijo sea mayor.

Algoritmo Greedy

Recorremos todos los posibles prefijos.

Para cada uno mantenemos:

- la suma del prefijo,
- el mínimo del prefijo,
- el máximo del sufijo restante.

El mejor valor que puede alcanzar ese prefijo es

$$\text{suma} - \text{mínimo} + \text{máximo}.$$

Si ese valor es al menos X , encontramos una respuesta válida.

Como conocemos el mínimo y el máximo en tiempo constante, el algoritmo es lineal.

$$O(N)$$

Ejemplo: Un Greedy menos evidente

Tenemos medias de cuatro tipos:

- Rojas
- Verdes
- Azules
- Especiales

Queremos formar pares maximizando el puntaje.

Par formado	Puntaje
Al menos una media especial	5
Dos medias del mismo color	2
Dos medias de distinto color	1

Pregunta

¿Qué estrategia greedy usarían?

Paso seguro

Si un color tiene una cantidad impar de medias, necesariamente una de ellas terminará emparejada con una media de otro tipo.

Por lo tanto:

- Si todavía quedan medias especiales, conviene usar una de ellas para formar ese par.
- De esa manera obtenemos **5 puntos** en lugar de **1 punto**.

Esta decisión nunca puede perjudicar una solución óptima.

Algoritmo Greedy

- 1 Para cada color con cantidad impar:
 - Si hay una media especial disponible, emparejarla con esa media.
- 2 Mientras sobren medias especiales:
 - Emparejarlas con medias de un mismo color.
- 3 Emparejar todas las medias restantes del mismo color entre sí.
- 4 Si quedan medias sin emparejar (a lo sumo una por color), emparejarlas arbitrariamente para obtener el último punto posible.

Outline

- ① Algoritmos Greedy
- ② ¿Cómo identificarlos?
- ③ ¿Cómo (intentar) demostrar un Greedy?
- ④ Algunos casos que fallan
- ⑤ Greedys Clásicos
- ⑥ Conclusión**

Existe una gran cantidad de problemas que pueden resolverse con algoritmos greedy, y cada uno tiene sus propias particularidades. En general, la mejor forma de mejorar la técnica, tanto para resolverlos como para detectarlos, es la práctica.

Además, recomiendo hacerse la siguiente pregunta:

"¿Existe algún otro algoritmo que resuelva este problema de manera similar a la estrategia greedy que estoy pensando?"

Muchas veces, comparar la idea con otros algoritmos conocidos ayuda a detectar errores o encontrar una estrategia más adecuada.

Por cualquier duda relacionada a esta clase, pueden consultarme a:

- fedenahuel07@gmail.com
- Telegram: FedeNQ