Greedy

23 de abril de 2019

Problema I

Figura: En una economía hay monedas con denominaciones: 20,10 y 5

Problema I

Figura: En una economía hay monedas con denominaciones: 20,10 y 5

Problema I

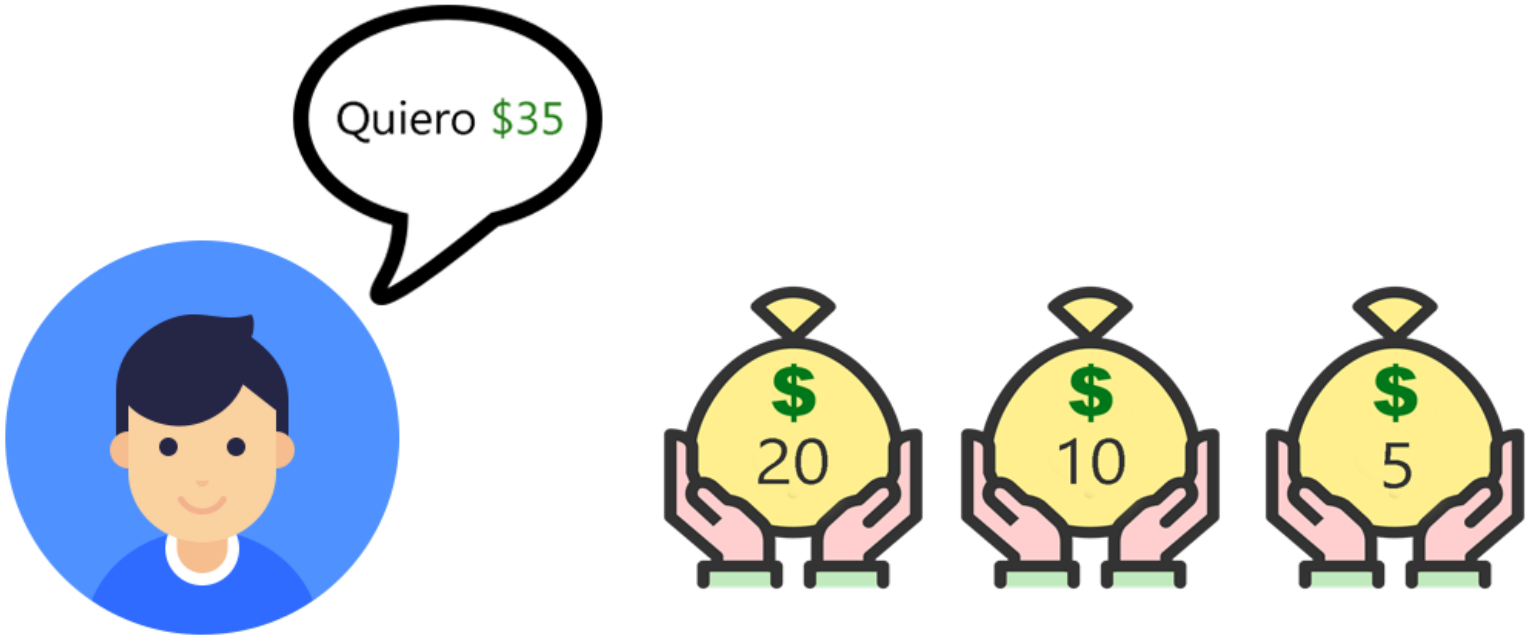

Figura: ¿Cómo dar $35 en la menor cantidad posible de monedas?

Problema I - Posibles soluciones

- 5+5+5+5+5+5+5
- 10+5+5+5+5+5
- 20+5+5+5
- ...

Problema I - Posibles soluciones

- 5+5+5+5+5+5+5
- 10+5+5+5+5+5
- 20+5+5+5
- ...
- **Solución óptima:** 20+10+5

Problema I - Idea

¿Qué algoritmo nos permitiría obtener un determinado monto a pagar en la **menor** cantidad de monedas posibles?

Problema I - Solución

Solución

En cada iteración, añadir a la solución la moneda de valor **más alto** tal que al agregarla no se pase del monto a pagar.

Problema I - Solución

Solución

En cada iteración, añadir a la solución la moneda de valor **más alto** tal que al agregarla no se pase del monto a pagar.

Este algoritmo es un algoritmo **Greedy** (o en español: Goloso)

¿Qué es un algoritmo Greedy?

- Los algoritmos greedy suelen resolver problemas de optimización. Por ejemplo:
  - **Minimizar un cierto costo.**
  - **Maximizar una cierta ganancia.**

- Un algoritmo se dice **greedy** si toma una decisión en cada paso (basada en cierto criterio), reduciendo el problema a una instancia más chica.

Consideración

- Las decisiones que toman los algoritmos greedy se asumen óptimas, y **no se revisan más adelante**.

- **Importante:**
  No todos los problemas admiten una solucion greedy. Sin embargo, es muy común creer que una idea greedy (errónea) funciona, pero finalmente el problema se soluciona con una técnica más sofisticada.

Volviendo al Problema I

Figura: En una economía hay monedas con denominaciones: 4,3 y 1

Volviendo al Problema I

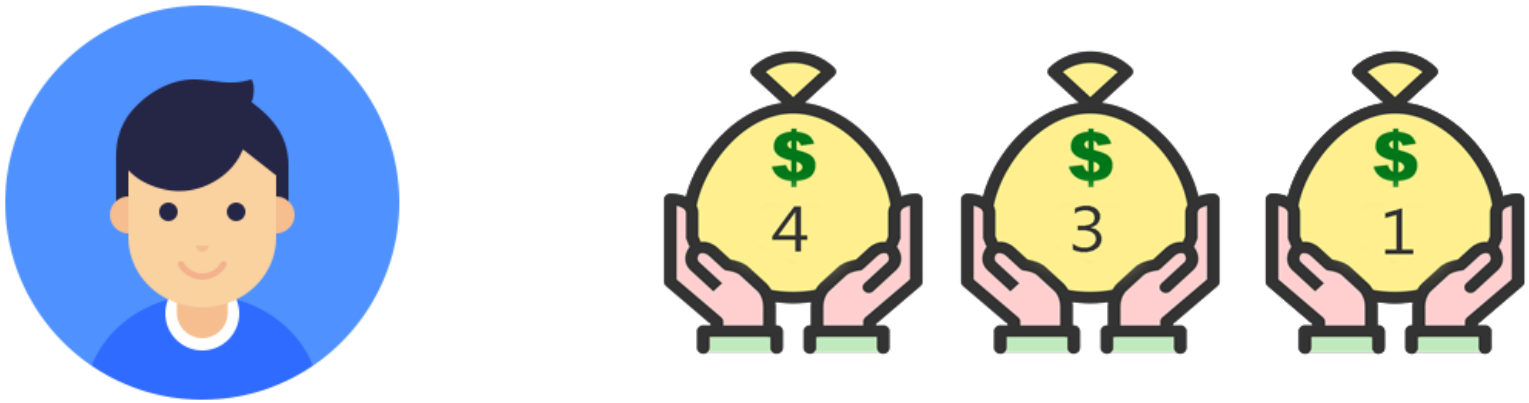

Figura: En una economía hay monedas con denominaciones: 4,3 y 1

Volviendo al Problema I

Figura: ¿Cómo dar $6 en la menor cantidad posible de monedas?

Greedy falla en Problema I

- El Greedy da como solución las monedas [4,1,1], pero la solución óptima es [3,3]. Por lo tanto, Greedy falla en este caso.
- Como veremos más adelante, este problema se resuelve con **programación dinámica.**

Problema II: Dragones

https://codeforces.com/problemset/problem/230/A

Enunciado

Kirito tiene que matar a n dragones. Inicialmente, Kirito tiene fuerza s y los dragones tienen fuerza x_i. $(1 \leqslant i \leqslant n)$. Cuando Kirito quiere matar a un dragón, si la fuerza de Kirito es mayor a la fuerza del dragón, entonces Kirito gana y adquiere y_i de fuerza, caso contrario, Kirito se muere.

Input

s: Fuerza inicial de Kirito

n: Cantidad de dragones

Siguen n líneas de la forma x_i y_i, donde x_i es la fuerza del i-ésimo dragón e y_i es la fuerza que obtiene Kirito si mata al i-ésimo dragón.

Output

YES si existe una forma de vencer a los n dragones

NO si **no** hay forma de vencer a los n dragones

Solución al problema II

Solución

Ordenar a los dragones por fuerza de menor a mayor, y en cada iteración, verificar si la fuerza de Kirito es mayor a la fuerza del dragón con el que se va a pelear. Si eso se da, se suma el y_i del dragón a la fuerza actual de Kirito y se continúa con la iteración hasta el final. Si en algún paso la fuerza de Kirito es menor o igual a la fuerza del dragón, se printea *NO* y se termina el programa. Si se llegó al final, se printea *YES*.

Problema III

`www.spoj.com/problems/BUSYMAN/`

Enunciado

Tengo planeadas n actividades. Cada una tiene un tiempo de inicio a_i y un tiempo de finalizacion b_i. Decir la mayor cantidad de actividades que puedo realizar (es decir, no se deben intersecar sus respectivos intervalos de tiempo, pero puedo empezar una actividad inmediatamente después de terminar otra).

Problema III - Solución

Solución

- Ordenar las actividades por hora de finalización de menor a mayor.
- Elegir la primer actividad y agregarla a la solución.
- Para las siguientes actividades, realizar lo siguiente:
  - Si la hora de inicio es mayor o igual a la hora de finalización de la ultima actividad elegida, elegir esta actividad y agregarla a la solución.

Problema III - Ejemplo de Solución

Ejemplo:

Actividad	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**
Inicio	0	3	1	5	5	8
Final	6	4	2	9	7	9

Cuadro: Ejemplo sin sortear

Ordenando por tiempo de finalización:

Actividad	**A3**	**A2**	**A1**	**A5**	**A6**	**A4**
Inicio	1	3	0	5	8	5
Final	2	4	6	7	9	9

Cuadro: Ejemplo sorteado

Problema III - Ejemplo de Solución

Ejemplo:

Actividad	**A1**	**A2**	**A3**	**A4**	**A5**	**A6**
Inicio	0	3	1	5	5	8
Final	6	4	2	9	7	9

Cuadro: Ejemplo sin sortear

Ordenando por tiempo de finalización:

Actividad	**A3**	**A2**	**A1**	**A5**	**A6**	**A4**
Inicio	1	3	0	5	8	5
Final	2	4	6	7	9	9

Cuadro: Ejemplo sorteado

Solución: A3 $\Rightarrow$ $A2 \Rightarrow A5 \Rightarrow A6$

Conclusión

- No hay mucha teoría o algoritmos generales detrás de greedy, sino que cada problema tiene su particularidad. Por eso el énfasis en los ejemplos.
- No hay otra forma de aprender greedy, más que resolviendo problemas que salgan con greedy :D

Greedy

23 de abril de 2019