Semana 8.- Algoritmos voraces

Sirven para encontrar una solución a ciertos problemas de optimización de manera directa, eligiendo en cada paso la mejor opción localmente, para encontrar una solución óptima. De una serie de candidatos decidimos cuál aceptamos o rechazamos. Cada candidato se elige únicamente una vez y no se vuelve a seleccionar.

Problema del cambio de monedas

Tenemos que pagar de manera exacta una cierta cantidad utilizando el mínimo número de monedas: contamos con monedas desde un céntimo hasta dos euros.

La solución pasará por considerar las monedas de manera decreciente, de tal manera que se utilizará la mayor moneda que se pueda usar en cualquier momento:

Imaginemos que tenemos que pagar 1,47 euros:

1. Miramos la moneda de 2 euros: no es válida.
2. Miramos la moneda de 1 euro, es válida→ restante 47 céntimos
3. Miramos la moneda de 1 euro, no es válida.
4. Miramos la moneda de 50 céntimos, no es válida.
5. Miramos la de 20 céntimos, es valida→ restante 27 céntimos.
6. Miramos la de 20 céntimos, es valida→ restante 7 céntimos.
7. Miramos la de 20 céntimos, no es válida.
8. Miramos la de 10 céntimos, no es válida.
9. Miramos la de 5 céntimos, es válida→ 2 céntimos restantes.
10. Miramos la de 5 céntimos, no es válida.
11. Miramos la de 2 céntimos, es válida→ 0 céntimos restantes.

Para que un algoritmo voraz se pueda considerar como tal, en cada paso la elección debe ser:

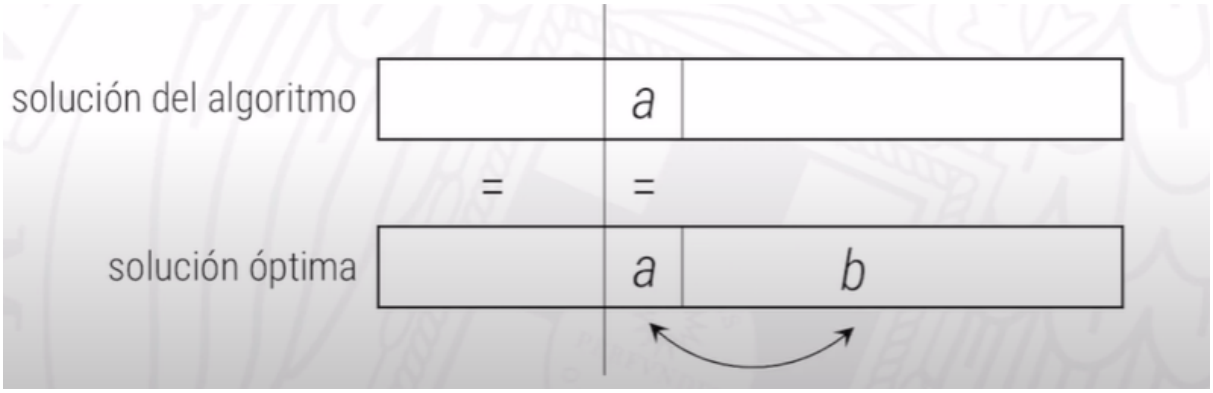
- **Factible.** Que satisfaga las restricciones del problema.
- **Óptima localmente**. La mejor opción local entre todas las disponibles.
- I**rrevocable.** No se puede cambiar en los pasos posteriores del algoritmo.

Características generales

- Para construir una solución de un conjunto de candidatos se forman dos conjuntos: los candidatos seleccionados y los rechazados.
- Existe una función de selección que indica que candidatos es prometedor entre los no seleccionados.
- Existe un test de factibilidad que comprueba si un candidatos es factible en una solución parcial.

Demostración de corrección

- Método de **reducción de diferencias.** Comparar una solución óptima con la obtenida por el algoritmo voraz. Si ambas soluciones no son iguales, se va transformando la óptima de manera que se mantenga óptima pero siendo más parecida a la del algoritmo voraz.



Problemas de la mochila

Especifícaciones

- Hay n objetos, cada uno con un peso pi>0 y con un valor vi>0
- La mochila soporta un peso máximo M.

- Hay que maximizar el valor que metemos en la mochila teniendo en cuenta que:
    - El peso total de los objetos metidos en la mochila no puedes sobrepasar M.

Ejemplo:

$M = 100, n = 5$

	1	2	3	4	5
p_i	10	20	30	40	50
v_i	20	30	66	40	60

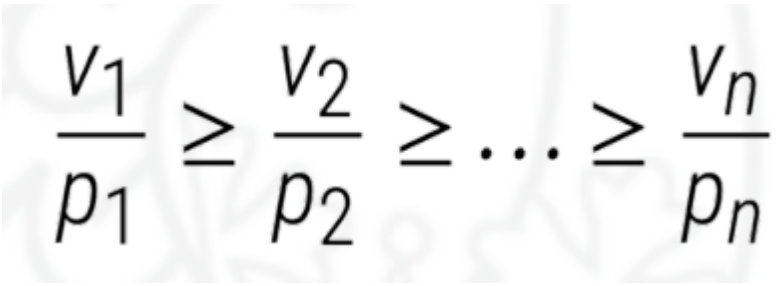
seleccionar	x_i					valor
máx. v_i	0	0	1	0,5	1	146

$M = 100, n = 5$

	1	2	3	4	5
p_i	10	20	30	40	50
v_i	20	30	66	40	60
$\frac{v_i}{p_i}$	2	1,5	2,2	1	1,2

seleccionar	x_i					valor
máx. v_i	0	0	1	0,5	1	146
mín. p_i	1	1	1	1	0	156
máx. $\frac{v_i}{p_i}$	1	1	1	0	0,8	164

Se meterá por tanto el objeto que mayor **vi/pi** y cuyo peso sea posible de meter, es decir, debemos ordenarlos de mayor a menor valor por unidad de peso.



$$\frac{v_1}{p_1} \geq \frac{v_2}{p_2} \geq \ldots \geq \frac{v_n}{p_n}$$

Hay que tener en cuenta que en este problema jugamos con objetos que se pueden partir, es decir, no tenemos que meter el objeto entero.

Tiene un coste de: **O(NLogN)**

Preguntas de test

- La solución ofrecida por el algoritmo de la mochila NO es único.
- El algoritmo de Dijkstra y Kruskal son voraces.
- El algoritmo de la mochila NO da soluciones óptimas para todos los sistemas monetarios.