

Fundamentos de análisis y diseño de algoritmos

Programación voraz

Selección de actividades

Características de la programación voraz

Programación voraz

- La programación dinámica puede resultar costosa. (Pocos subproblemas repetidos)
- Otra **estrategia** para resolver **problemas de optimización**: en cada estado de la búsqueda de una solución al problema, tomar el camino (la decisión) que es el mejor en ese momento (óptima), sin tener en cuenta las soluciones a subproblemas

Programación voraz

- Un algoritmo voraz toma decisiones con rapidez sobre vistas locales → toma decisiones óptimas locales. Espera que llegue a una solución óptima global
- Un algoritmo voraz no siempre encuentra la solución óptima global

Programación voraz

Problema de selección de actividades

- Suponga que se tiene un conjunto de actividades S etiquetadas con números de $a_1 \dots a_n$. $S = \{a_1, \dots, a_n\}$
 - Todas las actividades necesitan acceder a un mismo recurso
 - Cada actividad a_i tiene asociada dos valores:
 - s_i : tiempo inicial
 - f_i : tiempo final
- estos son los tiempos entre los cuales la actividad *debería* acceder al recurso

Programación voraz

Problema de selección de actividades

$$S = \{1, 2, 3\}$$

$(0, 5)$, $(1, 2)$, $(2, 3)$ son los tiempos para las 3 actividades

$$S_1 = \{1\} \quad S_2 = \{2, 3\}$$

Programación voraz

Problema de selección de actividades

$S=\{1,2,3\}$

$(0,5)$, $(1,2)$, $(2,3)$ son los tiempos para las 3 actividades

¿Cuáles son las diferentes formas de planificar las actividades?

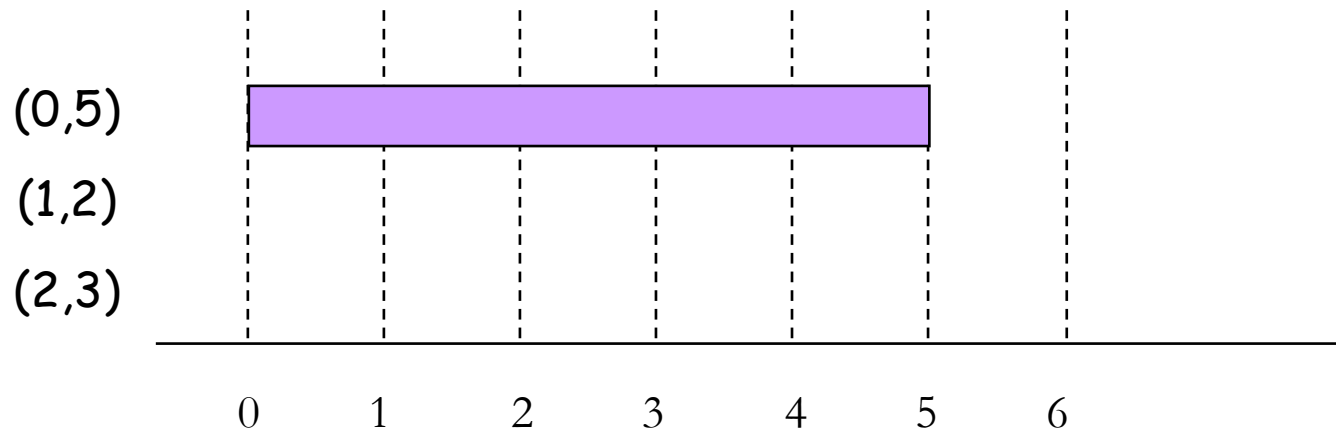
Programación voraz

Problema de selección de actividades

$S=\{1,2,3\}$

$(0,5)$, $(1,2)$, $(2,3)$ son los tiempos para las 3 actividades

Asignar el recurso a la actividad 1



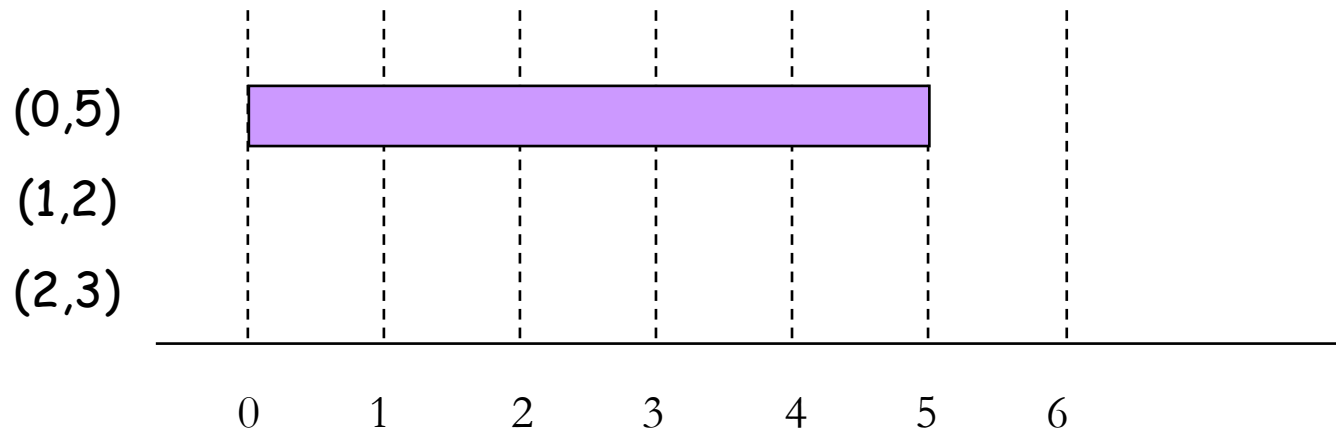
Programación voraz

Problema de selección de actividades

$S=\{1,2,3\}$

$(0,5)$, $(1,2)$, $(2,3)$ son los tiempos para las 3 actividades

Asignar el recurso a la actividad 1



Las actividades 2 y 3 no se podrían atender

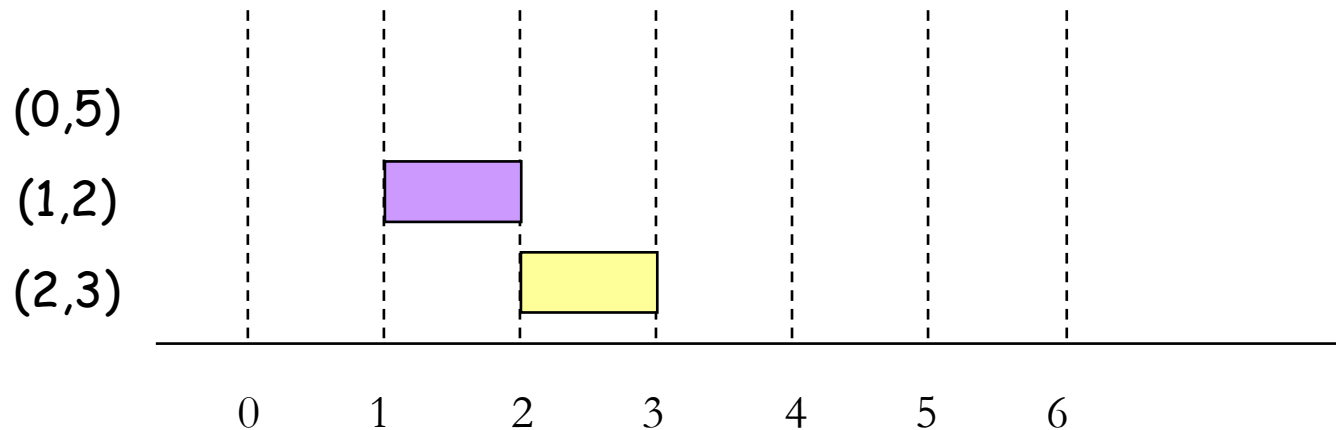
Programación voraz

Problema de selección de actividades

$S=\{1,2,3\}$

$(0,5)$, $(1,2)$, $(2,3)$ son los tiempos para las 3 actividades

Asignar el recurso a las actividades 2 y 3



La actividad 1 no se podría atender

Programación voraz

Problema de selección de actividades

Entrada: $S = \{a_1, \dots, a_n\}$

Salida: $A \subseteq S$, tal que $|A|$ es máxima

¿Cómo es la solución ingenua?

- 1) Enumerar todos los posibles subconjuntos de actividad
- 2) Buscar el de mayor tamaño

$O(2^n)$

Programación voraz

Problema de selección de actividades

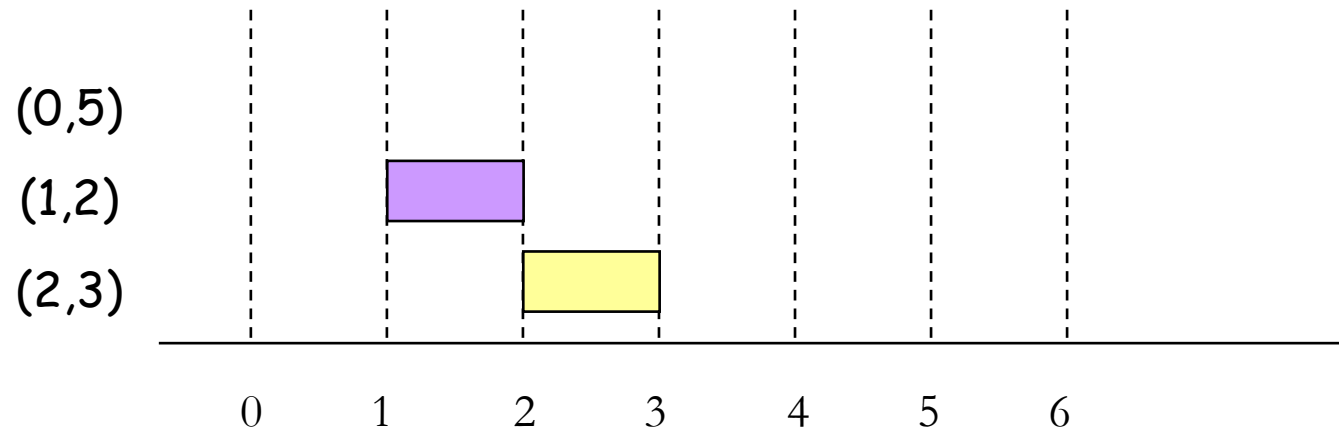
Entrada: $S = \{a_1, \dots, a_n\}$

Salida: $A \subseteq S$, tal que $|A|$ es máxima

(maximizar la cantidad de actividades que van a usar el recurso)

Programación voraz

Problema de selección de actividades

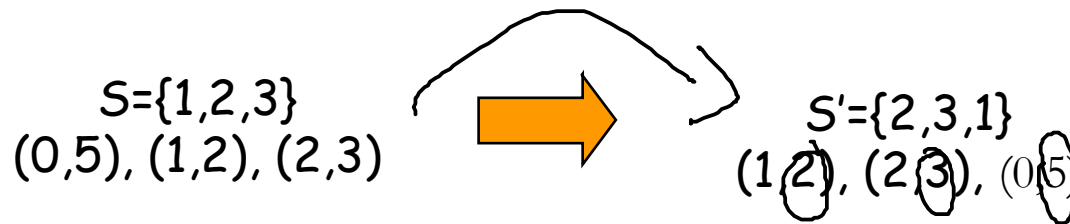


$A=\{2,3\}$ es la solución óptima

Programación voraz

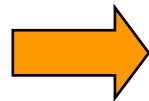
Solución:

- Ordenar las actividades ascendentemente según los tiempos de finalización f_i



- Coloque en la solución el primer recurso en la lista ordenada

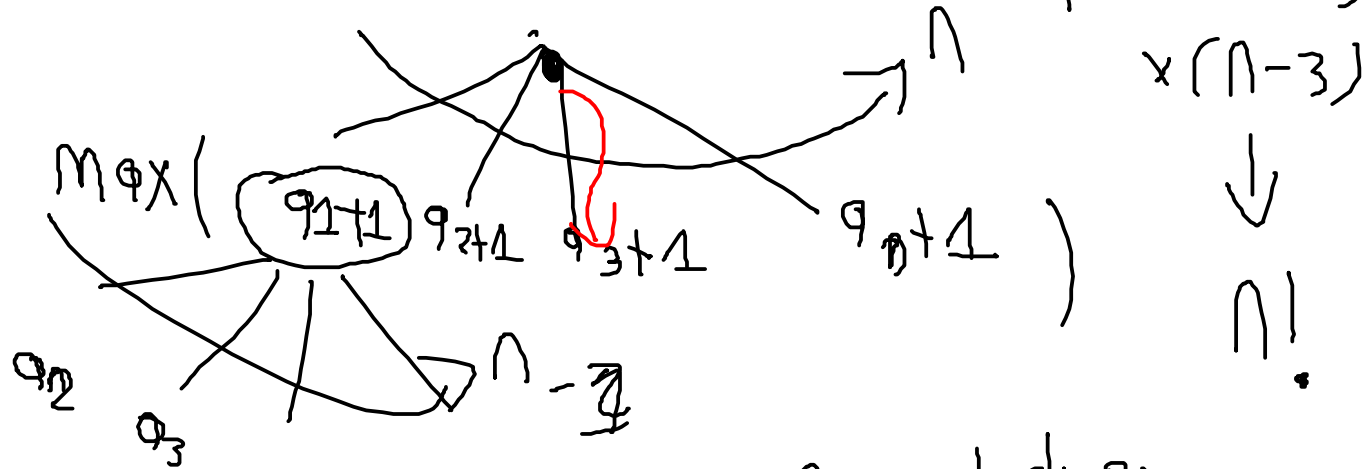
$S' = \{2, 3, 1\}$
 $(1, 2), (2, 3), (0, 5)$



$A = \{2\}$

$O(n \log n)$

$$Q = \{q_1, q_2, q_3, q_4, \dots, q_n\}$$



* element d.

num of dispo
* elemento E_{i+1}

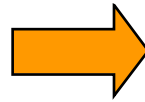
$$M[i, j] = \begin{cases} 0 & i=0 \\ \min_{k \in Q} [M[i-1, j+1]] & \text{num frag. d.} \end{cases}$$

Programación voraz

Solución:

- Coloque en la solución A , el recurso en S' que tiene tiempo de inicio menor o igual que el tiempo final del recurso que se acaba de planificar

$S'=\{2,3,1\}$
 $(1,2), (2,3), (0,5)$

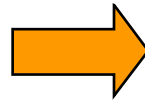


Programación voraz

Solución:

- Coloque en la solución A , el recurso en S' que tiene tiempo de inicio menor o igual que el tiempo final del recurso que se acaba de planificar

$S'=\{2,3,1\}$
 $(1,2), (2,3), (0,5)$

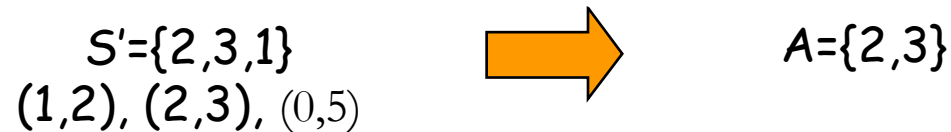


$A=\{2,3\}$

Programación voraz

Solución:

- Coloque en la solución A , el recurso en S' que tiene tiempo de inicio menor o igual que el tiempo final del recurso que se acaba de planificar



¿Por qué es una estrategia voraz?

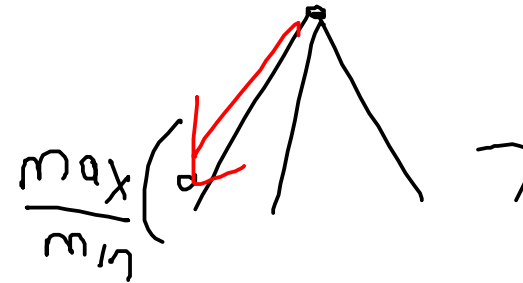
- Se toma una decisión óptima local en cada estado de la solución
- La decisión no depende de solucionar primero subproblemas relacionados

Programación voraz

¿Cuándo utilizar una estrategia voraz?

Cuando el problema exhiba:

- *Propiedad de escogencia voraz*
- *Subestructura óptima* ↪



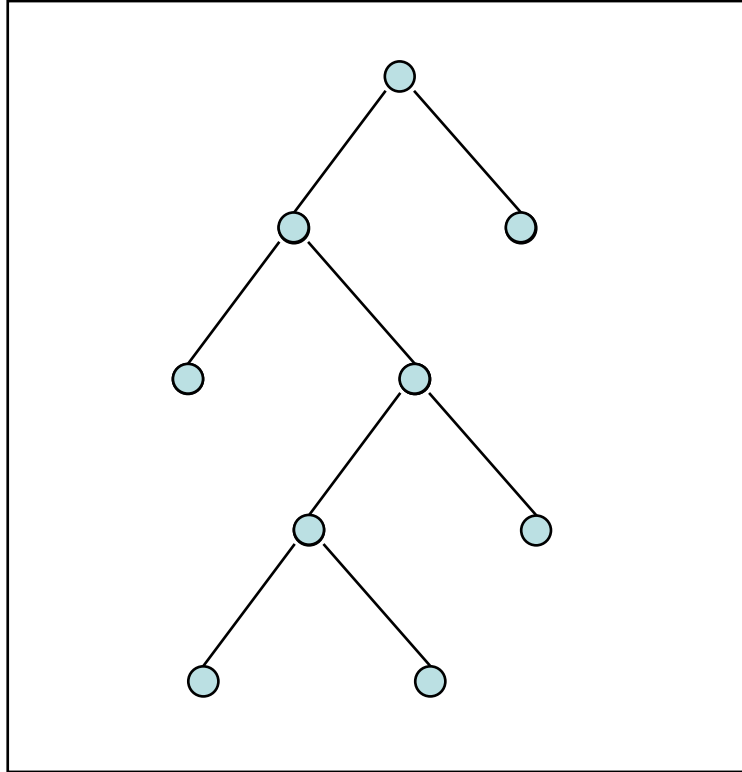
Programación voraz

¿Cuándo utilizar una estrategia voraz?

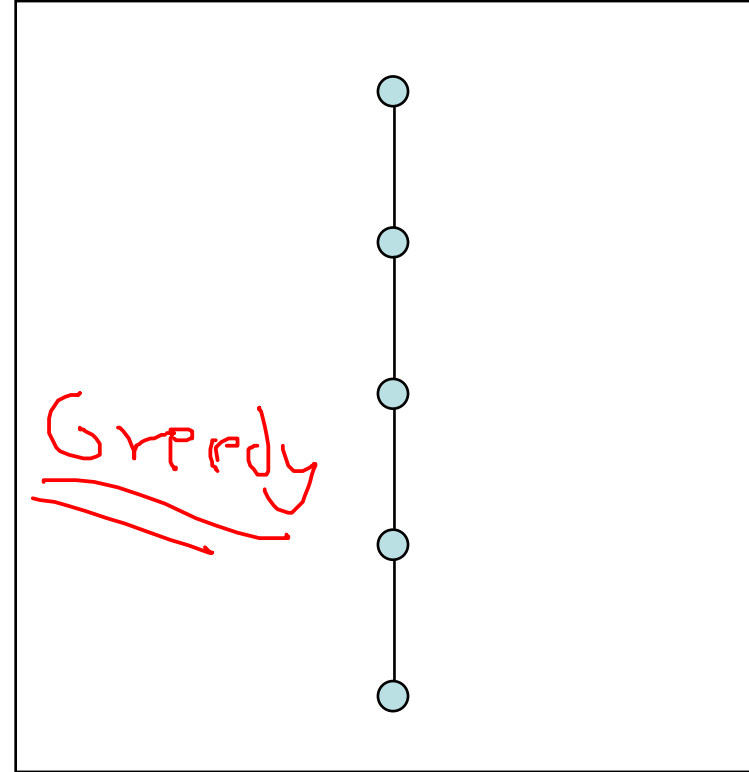
Cuando el problema exhiba:

- *Propiedad de escogencia voraz*: una solución óptima se puede hallar a partir de soluciones óptimas locales
- *Subestructura óptima*: igual que en programación dinámica

Programación voraz



Programación dinámica



Programación voraz

Programación voraz

Problema: Mochila 0-1

Se tienen N objetos y una mochila de capacidad (de peso) M , cada objeto tiene un peso w_i , $1 \leq i \leq N$. Cada objeto puede estar, o no, en la mochila. Además, se tiene un beneficio b_i por cada objeto

El problema consiste en maximizar el beneficio. La solución se representa indicando para cada objeto si se debe colocar o no en la mochila

Programación voraz

De manera formal, el problema consiste en encontrar $\langle x_1, x_2, \dots, x_n \rangle$ tal que:

$$\sum_{1 \leq i \leq N} b_i x_i \text{ sea máximo, sujeto a}$$

$$\sum_{1 \leq i \leq N} w_i x_i \leq M$$

$x_i \in \{0,1\}$, donde 0 significa que el objeto i no se coloca en la mochila y 1 que si

Programación voraz

$N=3$, $M=9$, $b=\langle 10,6,8 \rangle$, $w=\langle 3,4,5 \rangle$

$\langle 1,0,1 \rangle$ es una solución que indica colocar en la mochila los objetos 1 y 3, esto implica un beneficio de 18

$\langle 1,1,0 \rangle$ es una solución que indica colocar en la mochila los objetos 1 y 2, esto implica un beneficio de 16

$\langle 0,1,1 \rangle$ es una solución que indica colocar en la mochila los objetos 2 y 3, esto implica un beneficio de 14

Programación voraz

Estrategia voraz: seleccionar el ítem que tiene mayor beneficio por peso, esto es, b_i/w_i sea mayor

Programación voraz

Estrategia voraz: seleccionar el ítem que tiene mayor beneficio por peso, esto es, b_i/w_i sea mayor

$N=3, M=9, b=\langle 10, 6, 8 \rangle, w=\langle 3, 4, 5 \rangle$

Beneficio/peso = $\langle 10/3, 6/4, 8/5 \rangle = \langle 3.3, 1.5, 1.6 \rangle$

Seleccionar el item1, luego el item3 y por último el item2 (si caben)

Dinámica
 $O(mn)$

Voraz
 $O(n \log(n))$

Programación voraz

Estrategia voraz: seleccionar el ítem que tiene mayor beneficio por peso, esto es, b_i/w_i sea mayor

$N=3, M=9, b=\langle 10,6,8 \rangle, w=\langle 3,4,5 \rangle$

Beneficio/peso = $\langle 10/3, 6/4, 8/5 \rangle = \langle 3.3, 1.5, 1.6 \rangle$

Seleccionar el item1, luego el item3 y por último el item2 (si caben)

Solución: $\langle 1,0,1 \rangle$

Beneficio = $10+8$

Programación voraz

Estrategia voraz: seleccionar el ítem que tiene mayor beneficio por peso, esto es, b_i/w_i sea mayor

$N=3$, $M=50$, $b=\langle 60, 100, 120 \rangle$, $w=\langle 10, 20, 30 \rangle$

Beneficio/peso = $\langle 60/10, 100/20, 120/30 \rangle = \langle 6, 5, 4 \rangle$

Seleccionar el item1, luego el 2

Solución: $\langle 1, 1, 0 \rangle$

Beneficio = $60 + 100 = 160$

Programación voraz

Estrategia voraz: seleccionar el ítem que tiene mayor beneficio por peso, esto es, b_i/w_i sea mayor

$N=3, M=50, b=\langle 60, 100, 120 \rangle, w=\langle 10, 20, 30 \rangle$

$\text{Beneficio/peso} = \langle 60/10, 100/20, 120/30 \rangle = \langle 6, 5, 4 \rangle$

La solución óptima es: $\langle 0, 1, 1 \rangle$

Beneficio = $100 + 120 = 220$