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 $a_1 \ldots a_n$. $S=\{a_1, \ldots, a_n\}$
- 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

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?

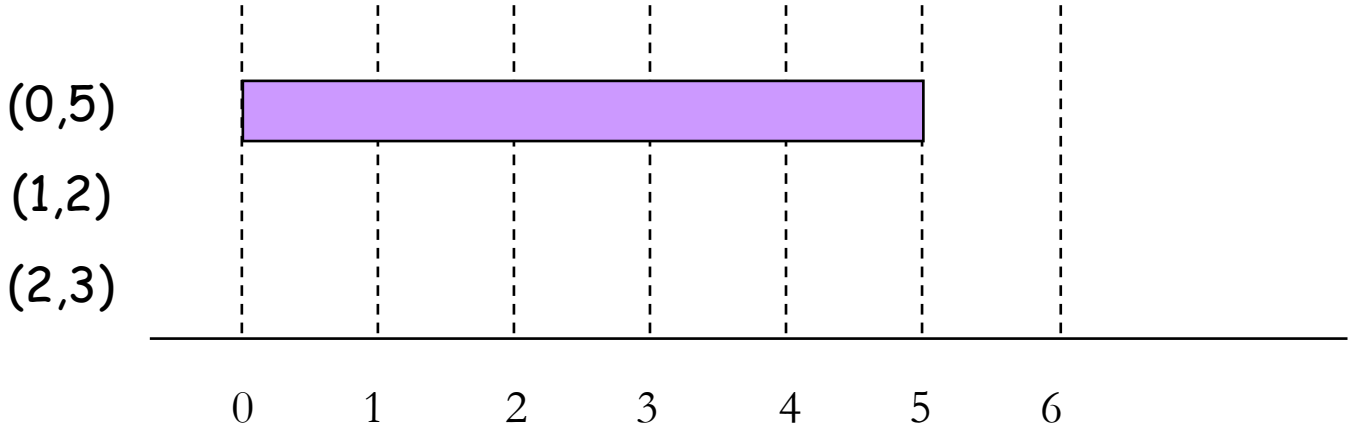
Programación voraz

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



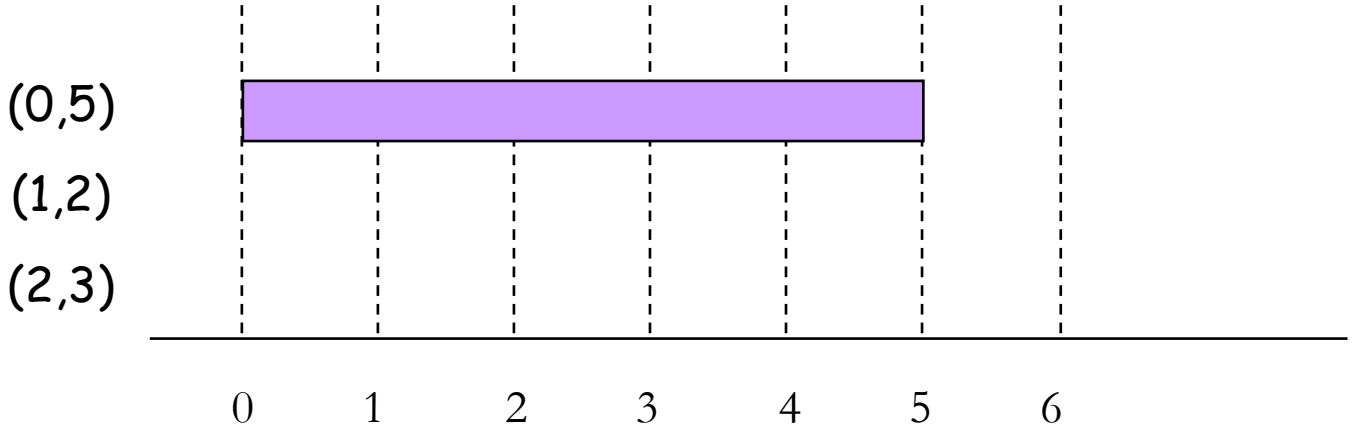
Programación voraz

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

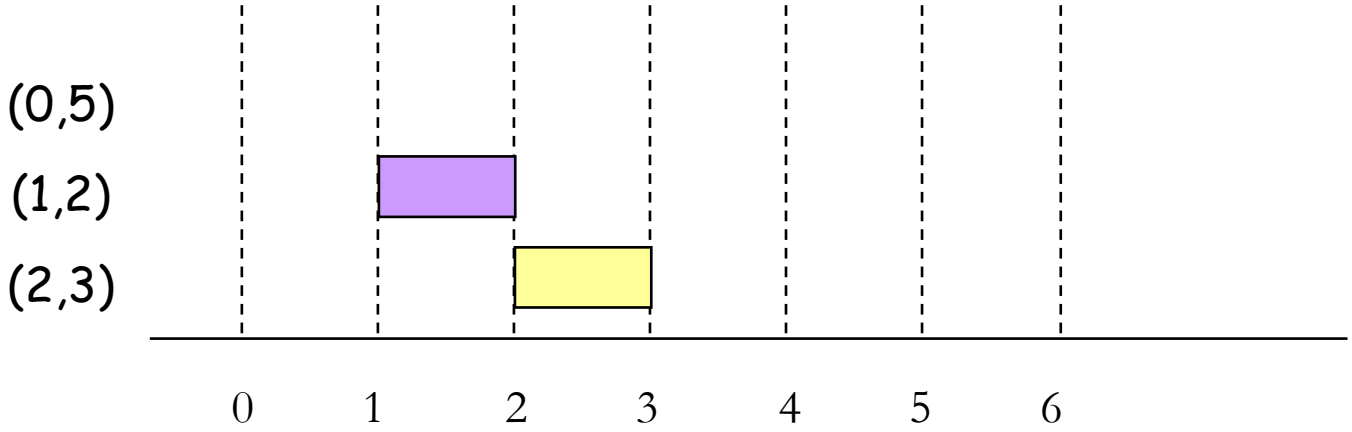
Programación voraz

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: $S=\{a_1, \ldots, a_n\}$

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

Programación voraz

Problema de selección de actividades

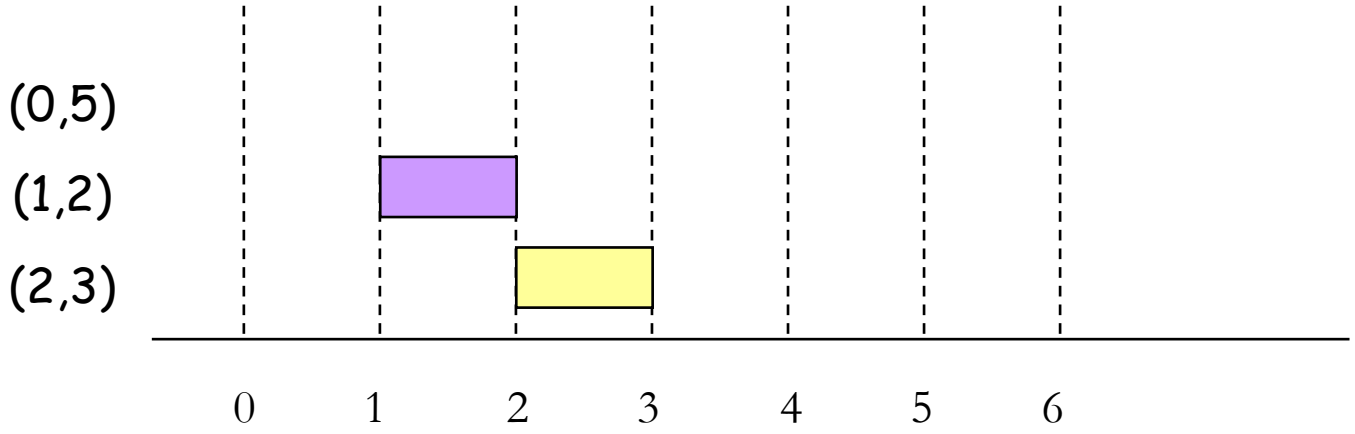
Entrada: $S=\{a_1, \ldots, a_n\}$

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



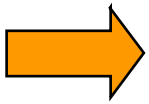
A={2,3} es la solución óptima

Programación voraz

Solución:

- Ordenar las actividades ascendentemente según los tiempos de finalización f_i

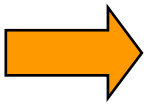
S={1,2,3}
(0,5), (1,2), (2,3)



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

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

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



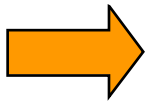
A={2}

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)

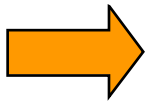


Programación voraz

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)



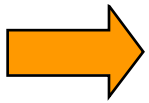
A={2,3}

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

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



A={2,3}

¿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*

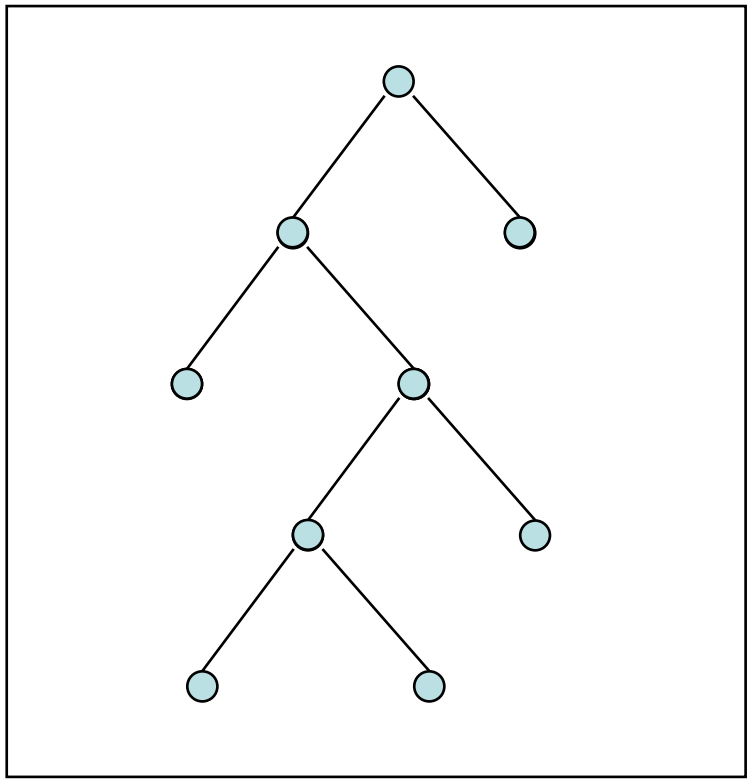
Programación voraz

¿Cuándo utilizar una estrategia voraz?

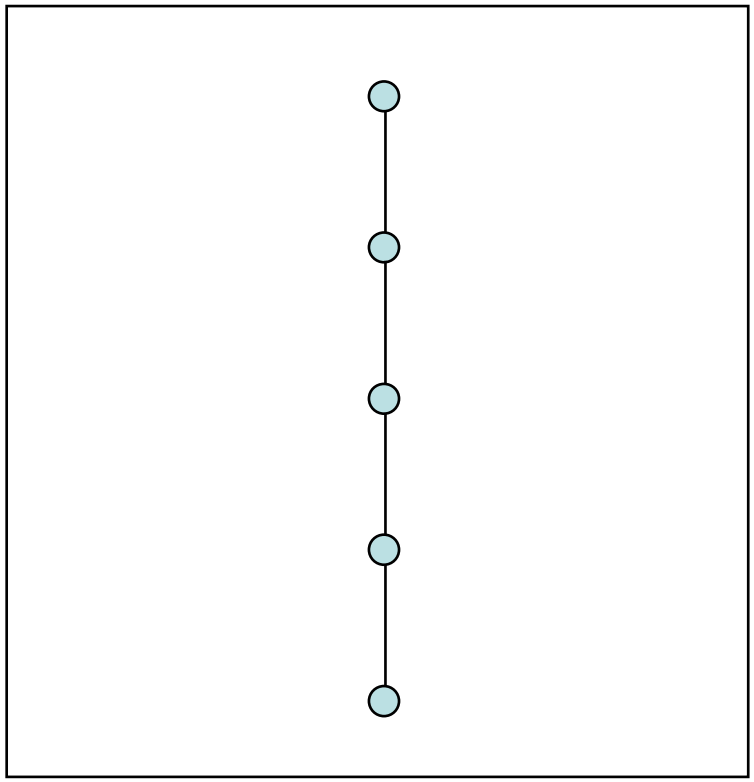
Cuando el problema exhiba:

- *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. Ademas, 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 $\langle x_1, x_2, \ldots, x_n \rangle$ tal que:

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

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

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

Programación voraz

N=3, M=9, b=<10,6,8>, w=<3,4,5>

<1,0,1> es una solución que indica colocar en la mochila los objetos 1 y 3, esto implica un beneficio de 18

<1,1,0> es una solución que indica colocar en la mochila los objetos 1 y 2, esto implica un beneficio de 16

<0,1,1> 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=<10,6,8>, w=<3,4,5>

Beneficio/peso=<10/3, 6/4, 8/5> = <3.3 , 1.5, 1.6 >

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

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=<10,6,8>, w=<3,4,5>

Beneficio/peso=<10/3, 6/4, 8/5> = <3.3 , 1.5, 1.6 >

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

Solución: <1,0,1>

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=<60,100,120>, w=<10,20,30>

Beneficio/peso=<60/10, 100/20, 120/30> = <6, 5, 4 >

Seleccionar el item1, luego el 2

Solución: <1,1,0>

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=<60,100,120>, w=<10,20,30>

Beneficio/peso=<60/10, 100/20, 120/30> = <6, 5, 4 >

La solución óptima es: <0,1,1>

Beneficio=100+120=220

1. Resuelve aplicando programación dinámica el problema siguiente: Se trata de asignar días de estudio para preparar los exámenes de cuatro asignaturas. Se dispone de 10 días para todas ellas, y estos días han de repartirse de manera que se optimice la mejora prevista en las calificaciones totales de las mismas.

   Se ha estimado que para un cierto número de días asignado a cada asignatura se pueden conseguir las mejoras en las notas que se indican en la tabla siguiente:

	Asignatura			
Días	1	2	3	4
1	1	3	1	2
2	3	4	2	4
3	4	4	4	5
4	5	5	4	5

   A ninguna asignatura se le asignarán más de cuatro días, y a cada una de ellas se le asignará al menos un día.

   Sugerencia: Define como etapas la asignación de días de estudio a cada una de las asignaturas.

Solución voraz para cada subproblema: Propongo solución 10 / #Asignaturas: 2.5

Asigno a tarea 4,3,2 Asigno dos días, cuando me queda, entonces asigno 4 días a la tarea 1

Solución:

Ganancia 15.