Algoritmos glotones(greedy)

mat-151

Algoritmos glotones

Algoritmos glotones

- Algoritmos utilizados en problemas de optimización.

Algoritmos glotones

- Algoritmos utilizados en problemas de optimización.
- Estos algoritmos siguen típicamente una secuencia de pasos con un conjunto de opciones en cada paso.

Algoritmos glotones

- Algoritmos utilizados en problemas de optimización.
- Estos algoritmos siguen típicamente una secuencia de pasos con un conjunto de opciones en cada paso.
- Un algoritmo glotón siempre toma la decisión que parece mejor en ese instante.

Algoritmos glotones

- Algoritmos utilizados en problemas de optimización.
- Estos algoritmos siguen típicamente una secuencia de pasos con un conjunto de opciones en cada paso.
- Un algoritmo glotón siempre toma la decisión que parece mejor en ese instante.
- Esto es, toma la solución localmente óptima esperando que le lleve a la solución globalmente óptima.

Algoritmos glotones

- Algoritmos utilizados en problemas de optimización.
- Estos algoritmos siguen típicamente una secuencia de pasos con un conjunto de opciones en cada paso.
- Un algoritmo glotón siempre toma la decisión que parece mejor en ese instante.
- Esto es, toma la solución localmente óptima esperando que le lleve a la solución globalmente óptima.
- Los algoritmos glotones no siempre llevan a la solución óptima pero para muchos problemas la solución óptima se encuentra de manera más eficiente que con programación dinámica.

Algoritmos glotones: **problema de selección de tareas compatibles**

Algoritmos glotones: **problema de selección de tareas compatibles**

- Planificación de actividades que compiten por el mismo recurso.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Planificación de actividades que compiten por el mismo recurso.
- Meta: seleccionar el conjunto de mayor tamaño de tareas compatibles.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Planificación de actividades que compiten por el mismo recurso.
- Meta: seleccionar el conjunto de mayor tamaño de tareas compatibles.
- Conjunto de n actividades propuestas $S = \{a_1, a_2, \ldots a_n\}$ que quieren usar el mismo recurso.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Planificación de actividades que compiten por el mismo recurso.
- Meta: seleccionar el conjunto de mayor tamaño de tareas compatibles.
- Conjunto de n actividades propuestas $S = \{a_1, a_2, \ldots a_n\}$ que quieren usar el mismo recurso.
- Cada actividad $\mathbf{a_i}$ tiene un tiempo de inicio $\mathbf{s_i}$ y un tiempo final $\mathbf{f_i}$ donde:

Algoritmos glotones: **problema de selección de tareas compatibles**

- Planificación de actividades que compiten por el mismo recurso.
- Meta: seleccionar el conjunto de mayor tamaño de tareas compatibles.
- Conjunto de n actividades propuestas $S = \{a_1, a_2, \ldots a_n\}$ que quieren usar el mismo recurso.
- Cada actividad $\mathbf{a_i}$ tiene un tiempo de inicio $\mathbf{s_i}$ y un tiempo final $\mathbf{f_i}$ donde:

$$0 \leq s_i \leq f_i < \text{inf} .$$

Algoritmos glotones: **problema de selección de tareas compatibles**

- Planificación de actividades que compiten por el mismo recurso.
- Meta: seleccionar el conjunto de mayor tamaño de tareas compatibles.
- Conjunto de n actividades propuestas $S = \{a_1, a_2, \ldots a_n\}$ que quieren usar el mismo recurso.
- Cada actividad $\mathbf{a_i}$ tiene un tiempo de inicio $\mathbf{s_i}$ y un tiempo final $\mathbf{f_i}$ donde:

$$0 \leq s_i \leq f_i < \inf .$$

- Si es seleccionada, la actividad $\mathbf{a_i}$ tiene lugar en el intervalo semi-abierto $\mathbf{[s_i, f_i)}$.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Planificación de actividades que compiten por el mismo recurso.
- Meta: seleccionar el conjunto de mayor tamaño de tareas compatibles.
- Conjunto de n actividades propuestas $S = \{a_1, a_2, \ldots a_n\}$ que quieren usar el mismo recurso.
- Cada actividad $\mathbf{a_i}$ tiene un tiempo de inicio $\mathbf{s_i}$ y un tiempo final $\mathbf{f_i}$ donde:

$$0 \leq s_i \leq f_i < \inf .$$

- Si es seleccionada, la actividad $\mathbf{a_i}$ tiene lugar en el intervalo semi-abierto $\mathbf{[s_i, f_i)}$.
- Las actividades $\mathbf{a_i}$ y $\mathbf{a_j}$ son compatibles si los intervalos $[s_i, f_i)$ y $[s_j, f_j)$ no se translapan (si $s_i \geq f_j$ o si $s_j \geq f_i$).

Algoritmos glotones: **problema de selección de tareas compatibles**

Algoritmos glotones: **problema de selección de tareas compatibles**

- El problema de selección de actividades es el seleccionar el subconjunto de tamaño máximo de actividades mutuamente compatibles.

Algoritmos glotones: **problema de selección de tareas compatibles**

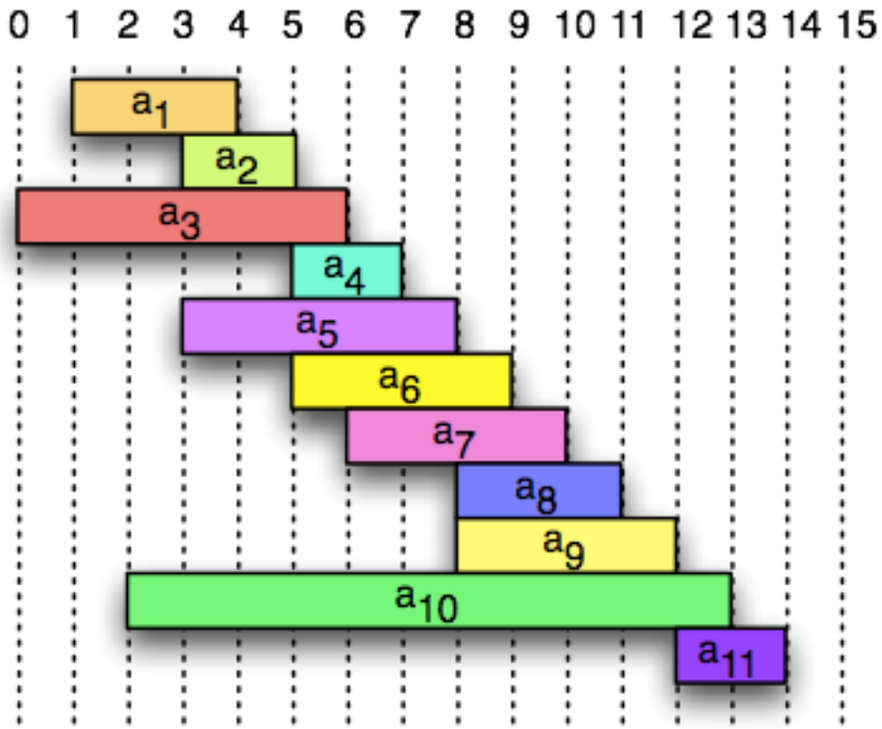
- El problema de selección de actividades es el seleccionar el subconjunto de tamaño máximo de actividades mutuamente compatibles.

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

Algoritmos glotones: **problema de selección de tareas compatibles**

- El problema de selección de actividades es el seleccionar el subconjunto de tamaño máximo de actividades mutuamente compatibles.

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

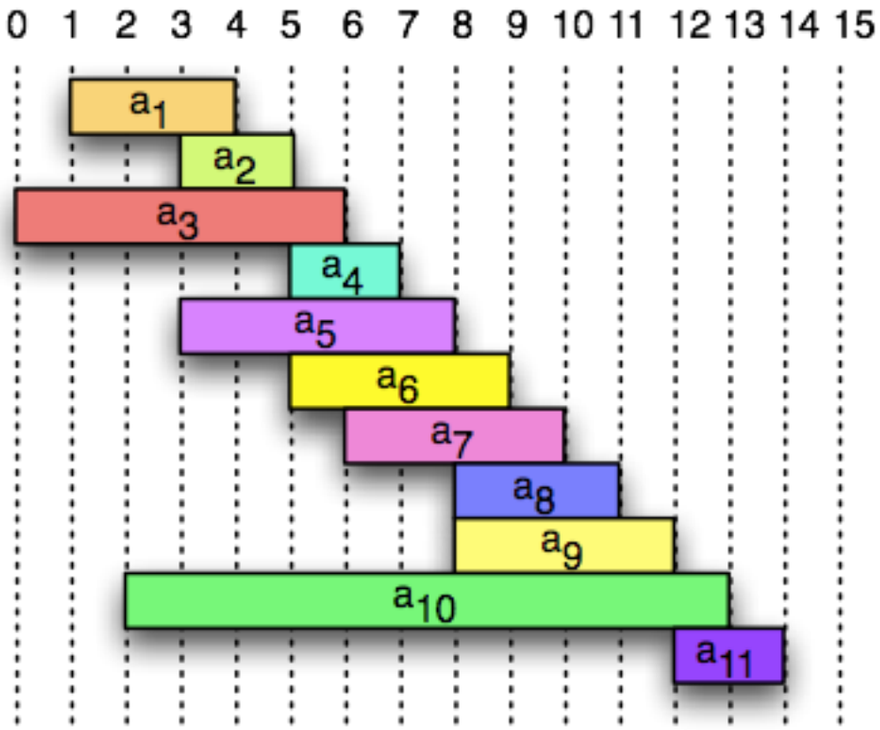


Algoritmos glotones: **problema de selección de tareas compatibles**

- El problema de selección de actividades es el seleccionar el subconjunto de tamaño máximo de actividades mutuamente compatibles.

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

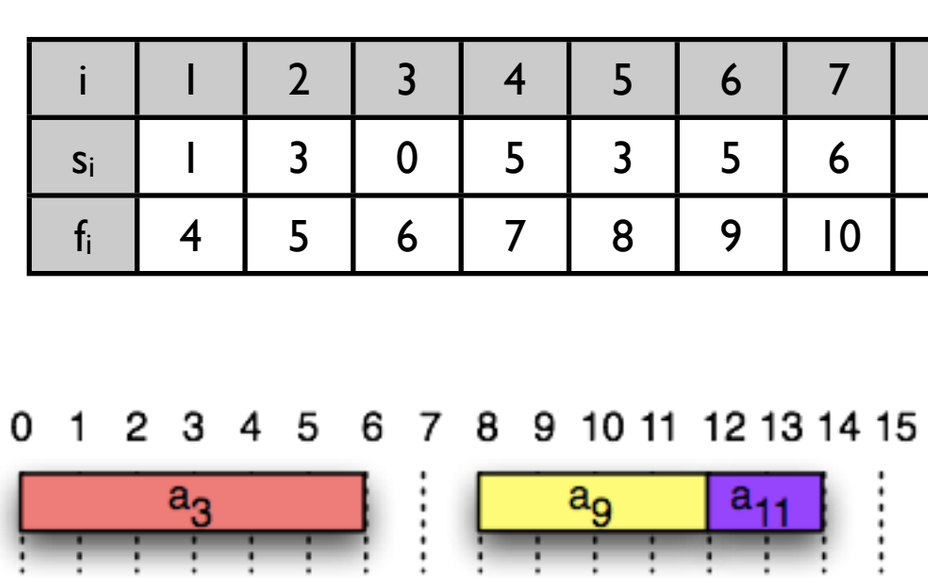
$$[a_3, a_9, a_{11}]$$



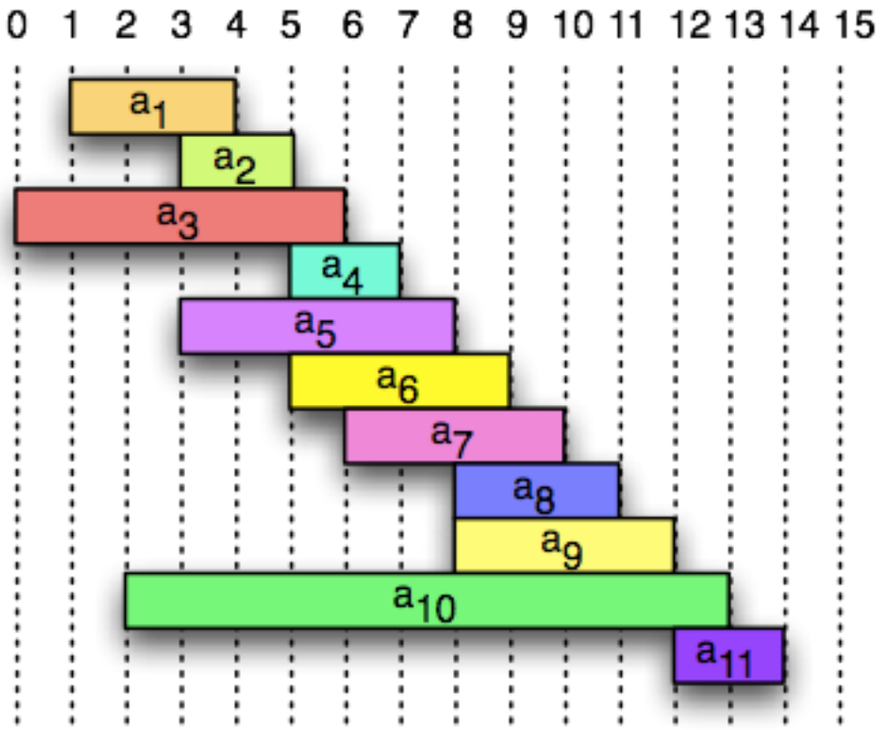
Algoritmos glotones: **problema de selección de tareas compatibles**

- El problema de selección de actividades es el seleccionar el subconjunto de tamaño máximo de actividades mutuamente compatibles.

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14



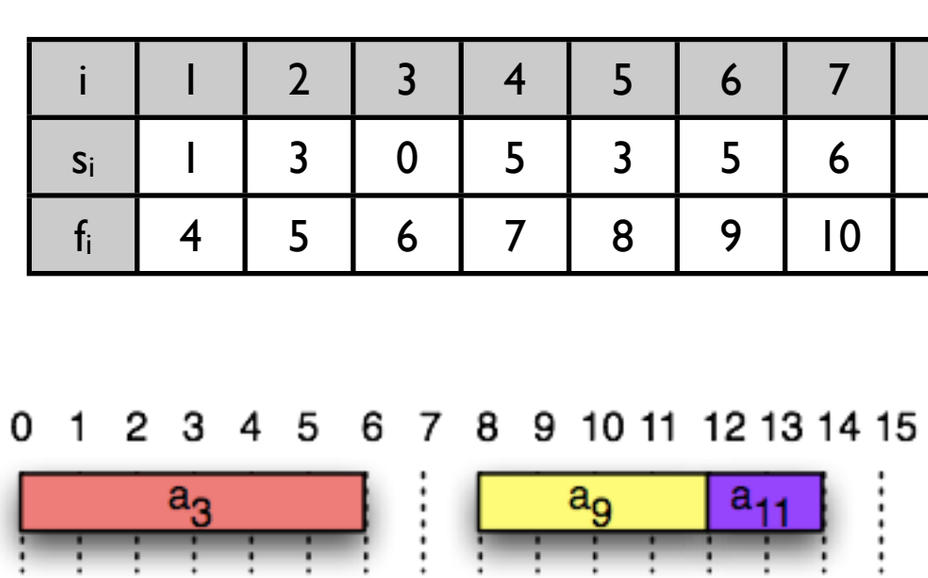
$[a_3, a_9, a_{11}]$



Algoritmos glotones: **problema de selección de tareas compatibles**

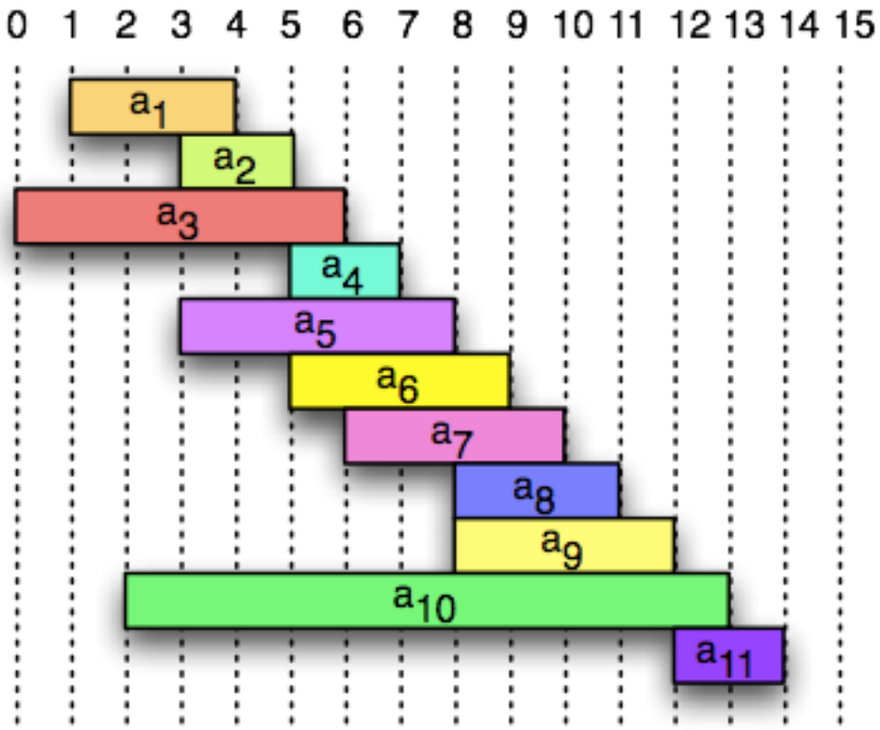
- El problema de selección de actividades es el seleccionar el subconjunto de tamaño máximo de actividades mutuamente compatibles.

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14



$[a_3, a_9, a_{11}]$

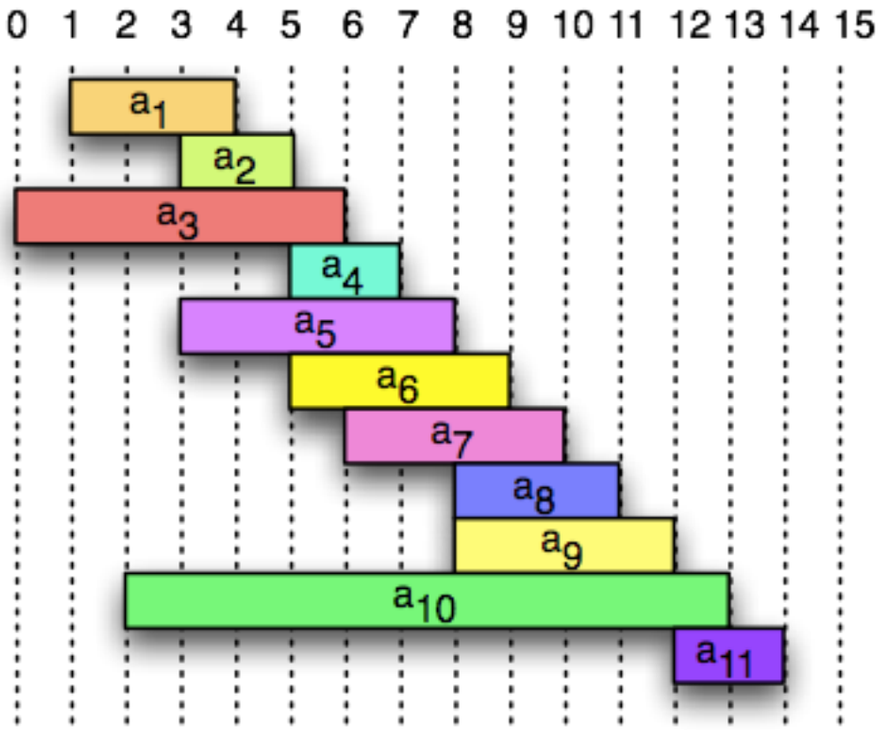
$[a_1, a_4, a_8, a_{11}]$

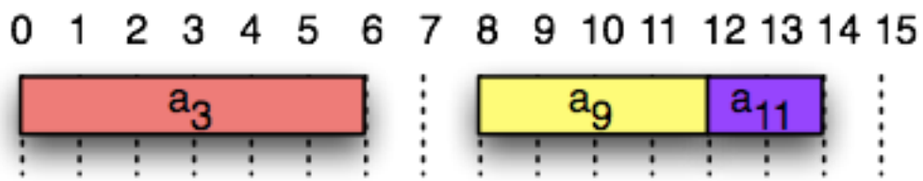


Algoritmos glotones: **problema de selección de tareas compatibles**

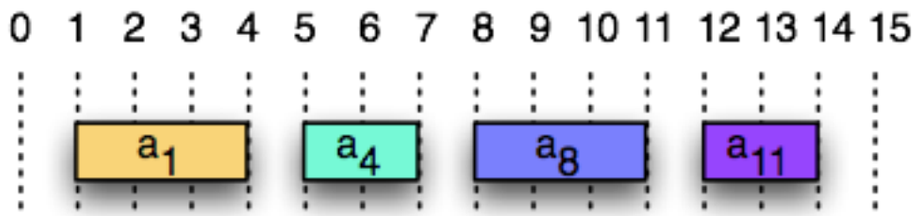
- El problema de selección de actividades es el seleccionar el subconjunto de tamaño máximo de actividades mutuamente compatibles.

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14





$[a_3, a_9, a_{11}]$

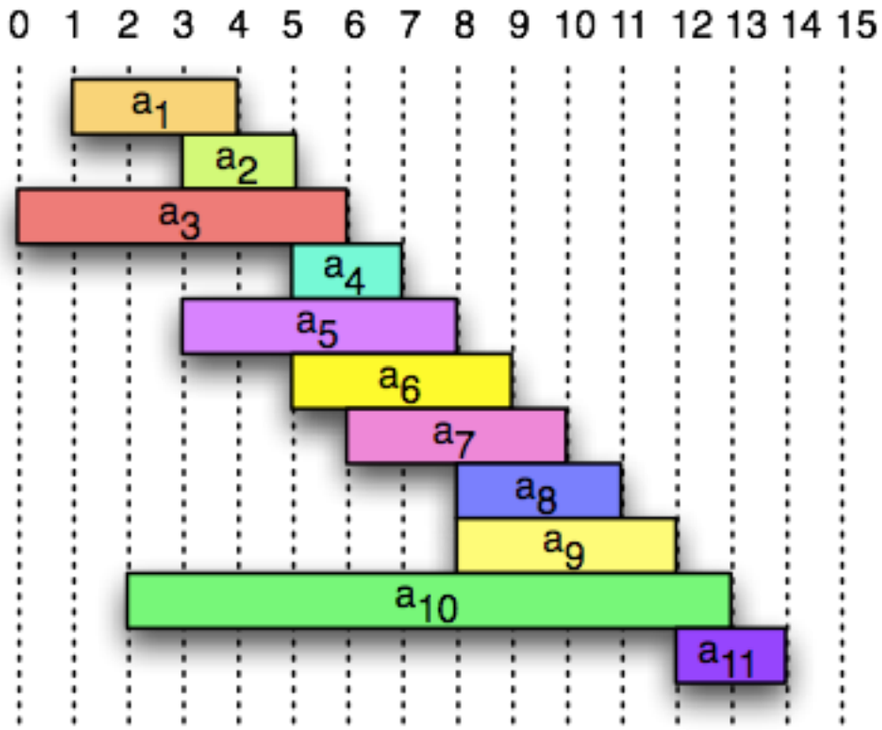


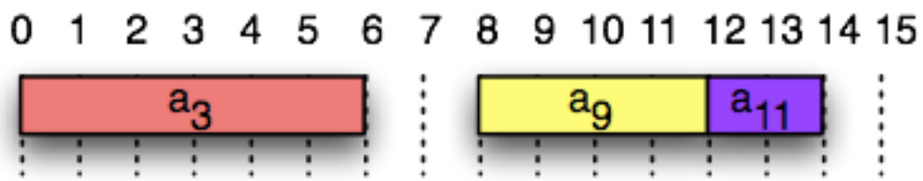
$[a_1, a_4, a_8, a_{11}]$

Algoritmos glotones: **problema de selección de tareas compatibles**

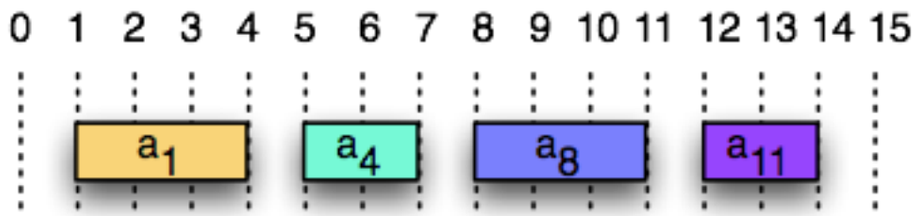
- El problema de selección de actividades es el seleccionar el subconjunto de tamaño máximo de actividades mutuamente compatibles.

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14





$[a_3, a_9, a_{11}]$



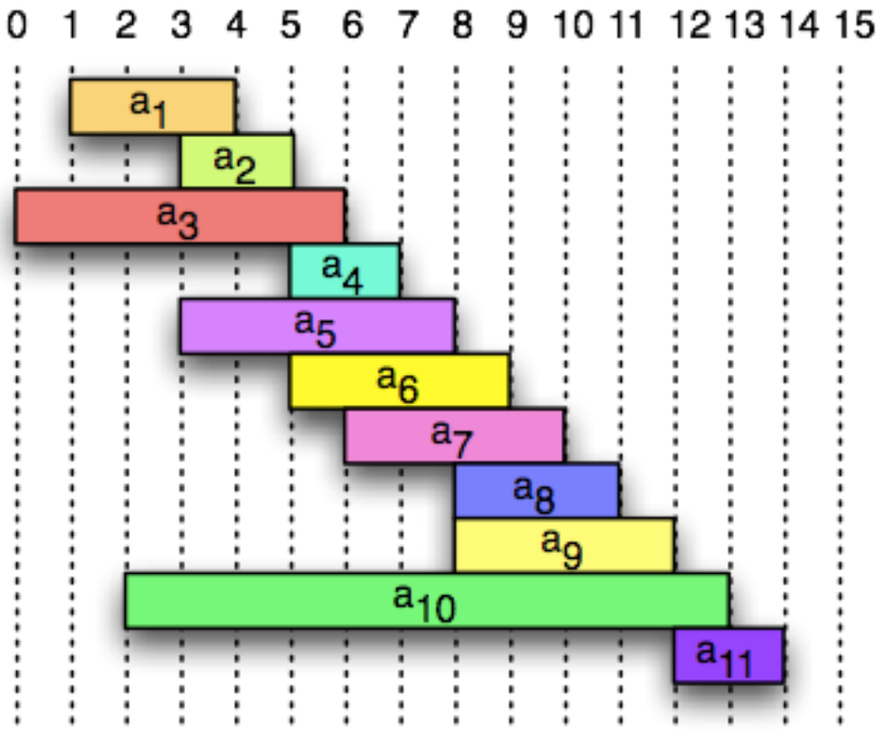
$[a_1, a_4, a_8, a_{11}]$

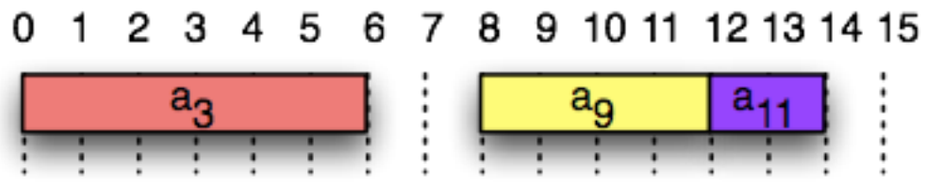
$[a_2, a_4, a_9, a_{11}]$

Algoritmos glotones: **problema de selección de tareas compatibles**

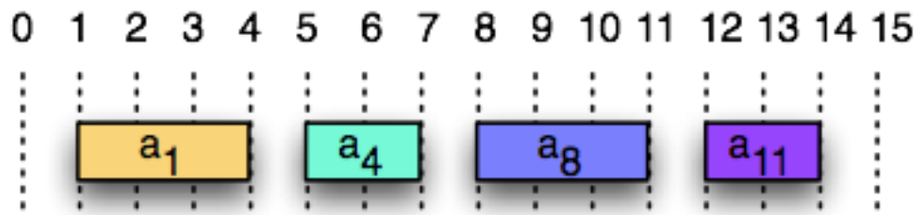
- El problema de selección de actividades es el seleccionar el subconjunto de tamaño máximo de actividades mutuamente compatibles.

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

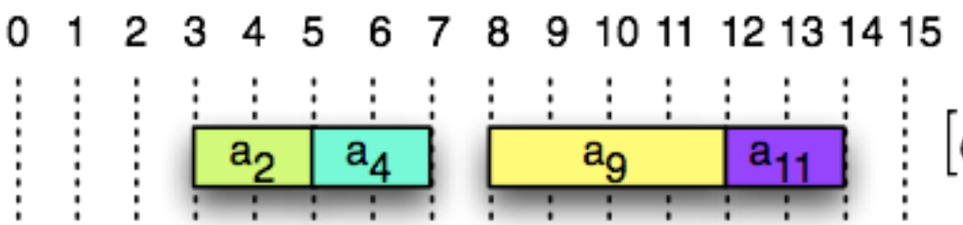




$[a_3, a_9, a_{11}]$



$[a_1, a_4, a_8, a_{11}]$



$[a_2, a_4, a_9, a_{11}]$

Algoritmos glotones: **problema de selección de tareas compatibles**

i

Algoritmos glotones: **problema de selección de tareas compatibles**

- Solución utilizando programación dinámica: combinar las soluciones óptimas a dos subproblemas para construir una solución óptima al problema original.

i

Algoritmos glotones: **problema de selección de tareas compatibles**

- Solución utilizando programación dinámica: combinar las soluciones óptimas a dos subproblemas para construir una solución óptima al problema original.
- Paso 1: subestructura óptima del problema de selección de actividades.

i

Algoritmos glotones: **problema de selección de tareas compatibles**

- Solución utilizando programación dinámica: combinar las soluciones óptimas a dos subproblemas para construir una solución óptima al problema original.
- Paso 1: subestructura óptima del problema de selección de actividades.

Definimos $S_{ij} = \{a_k \in S : f_i \leq s_k < f_k \leq s_j\}$ tal que S_{ij} es el subconjunto de actividades en S que puedan empezar después que la actividad a_i termine, y que terminen antes de que la actividad a_j empiece (actividades compatibles con a_i y a_j).

Algoritmos glotones: **problema de selección de tareas compatibles**

- Solución utilizando programación dinámica: combinar las soluciones óptimas a dos subproblemas para construir una solución óptima al problema original.
- Paso 1: subestructura óptima del problema de selección de actividades.

Definimos $S_{ij} = \{a_k \in S : f_i \leq s_k < f_k \leq s_j\}$ tal que S_{ij} es el subconjunto de actividades en S que puedan empezar después que la actividad a_i termine, y que terminen antes de que la actividad a_j empiece (actividades compatibles con a_i y a_j).

Agregamos las actividades a_0 y a_{n+1} con la convención que $f_0 = 0$ y $s_{n+1} = \infty$.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Solución utilizando programación dinámica: combinar las soluciones óptimas a dos subproblemas para construir una solución óptima al problema original.
- Paso 1: subestructura óptima del problema de selección de actividades.

Definimos $S_{ij} = \{a_k \in S : f_i \leq s_k < f_k \leq s_j\}$ tal que S_{ij} es el subconjunto de actividades en S que puedan empezar después que la actividad a_i termine, y que terminen antes de que la actividad a_j empiece (actividades compatibles con a_i y a_j).

Agregamos las actividades a_0 y a_{n+1} con la convención que $f_0 = 0$ y $s_{n+1} = \infty$.

Entonces $S = S_{0,n+1}$ y los rangos para i y j están dados por $0 \leq i, j \leq n + 1$.

Algoritmos glotones: **problema de selección de tareas compatibles**

Algoritmos glotones: **problema de selección de tareas compatibles**

- Si ordenamos las actividades en orden creciente respecto a sus tiempos de finalización, tenemos que:

Algoritmos glotones: **problema de selección de tareas compatibles**

- Si ordenamos las actividades en orden creciente respecto a sus tiempos de finalización, tenemos que:

$$f_0 \leq f_1 \leq f_2 \leq \ldots \leq f_n < f_{n+1}$$

Algoritmos glotones: **problema de selección de tareas compatibles**

- Si ordenamos las actividades en orden creciente respecto a sus tiempos de finalización, tenemos que:

$$f_0 \leq f_1 \leq f_2 \leq \ldots \leq f_n < f_{n+1}$$

y podemos decir que

Algoritmos glotones: **problema de selección de tareas compatibles**

- Si ordenamos las actividades en orden creciente respecto a sus tiempos de finalización, tenemos que:

$$f_0 \leq f_1 \leq f_2 \leq \ldots \leq f_n < f_{n+1}$$

y podemos decir que

$S_{ij} = \emptyset$ cuando $i \geq j$.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Si ordenamos las actividades en orden creciente respecto a sus tiempos de finalización, tenemos que:

$$f_0 \leq f_1 \leq f_2 \leq \ldots \leq f_n < f_{n+1}$$

y podemos decir que

$$S_{ij} = \emptyset \text{ cuando } i \geq j.$$

- Para encontrar la subestructura optima consideramos un subproblema no-vacío y suponemos que una solución a este subproblema incluye una actividad $\mathbf{a_k}$ tal que:

$$f_i \leq s_k < f_k \leq s_j$$

Algoritmos glotones: **problema de selección de tareas compatibles**

- Si ordenamos las actividades en orden creciente respecto a sus tiempos de finalización, tenemos que:

$$f_0 \leq f_1 \leq f_2 \leq \ldots \leq f_n < f_{n+1}$$

y podemos decir que

$$S_{ij} = \emptyset \text{ cuando } i \geq j.$$

- Para encontrar la subestructura optima consideramos un subproblema no-vacío y suponemos que una solución a este subproblema incluye una actividad $\mathbf{a_k}$ tal que:

$$f_i \leq s_k < f_k \leq s_j$$

- La actividad $\mathbf{a_k}$ nos genera dos subproblemas, $\mathbf{S_{ik}}$ y $\mathbf{S_{kj}}$, donde cada una consiste en un subconjunto de las actividades presentes en $\mathbf{S_{ij}}$.

Algoritmos glotones: **problema de selección de tareas compatibles**

Algoritmos glotones: **problema de selección de tareas compatibles**

- La estructura subóptima de este problema es la siguiente:

Algoritmos glotones: **problema de selección de tareas compatibles**

- La estructura subóptima de este problema es la siguiente:
  - supongamos que una solución óptima A_{ij} para S_{ij} incluye la actividad a_k.

Algoritmos glotones: **problema de selección de tareas compatibles**

- La estructura subóptima de este problema es la siguiente:
  - supongamos que una solución óptima A_{ij} para S_{ij} incluye la actividad a_k.
  - entonces las soluciones A_{ik} a S_{ik} y A_{kj} a S_{kj} utilizadas en esa solución óptima de S_{ij} son óptimas también.

Algoritmos glotones: **problema de selección de tareas compatibles**

- La estructura subóptima de este problema es la siguiente:
  - supongamos que una solución óptima A_{ij} para S_{ij} incluye la actividad a_k.
  - entonces las soluciones A_{ik} a S_{ik} y A_{kj} a S_{kj} utilizadas en esa solución óptima de S_{ij} son óptimas también.
  - se puede entonces encontrar un subconjunto de tamaño máximo de actividades compatibles en S_{ij} separando el problema en dos subproblemas y uniendo sus soluciones:

Algoritmos glotones: **problema de selección de tareas compatibles**

- La estructura subóptima de este problema es la siguiente:
  - supongamos que una solución óptima A_{ij} para S_{ij} incluye la actividad a_k.
  - entonces las soluciones A_{ik} a S_{ik} y A_{kj} a S_{kj} utilizadas en esa solución óptima de S_{ij} son óptimas también.
  - se puede entonces encontrar un subconjunto de tamaño máximo de actividades compatibles en S_{ij} separando el problema en dos subproblemas y uniendo sus soluciones:

$$A_{ij} = A_{ik} \cup \{a_k\} \cup A_{kj}$$

Algoritmos glotones: **problema de selección de tareas compatibles**

- La estructura subóptima de este problema es la siguiente:
  - supongamos que una solución óptima $\mathbf{A_{ij}}$ para $\mathbf{S_{ij}}$ incluye la actividad $\mathbf{a_k}$.
  - entonces las soluciones $\mathbf{A_{ik}}$ a $\mathbf{S_{ik}}$ y $\mathbf{A_{kj}}$ a $\mathbf{S_{kj}}$ utilizadas en esa solución óptima de $\mathbf{S_{ij}}$ son óptimas también.
  - se puede entonces encontrar un subconjunto de tamaño máximo de actividades compatibles en $\mathbf{S_{ij}}$ separando el problema en dos subproblemas y uniendo sus soluciones:

$$A_{ij} = A_{ik} \cup \{a_k\} \cup A_{kj}$$

Definida para todo el problema $S_{0,n+1}$.

Algoritmos glotones: **problema de selección de tareas compatibles**

Algoritmos glotones: **problema de selección de tareas compatibles**

- Paso 2: solución recursiva

Algoritmos glotones: **problema de selección de tareas compatibles**

- Paso 2: solución recursiva

- Sea **c[i,j]** el número de actividades en un subconjunto de actividades compatibles de tamaño máximo en $\mathbf{S_{ij}}$.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Paso 2: solución recursiva

- Sea **c[i,j]** el número de actividades en un subconjunto de actividades compatibles de tamaño máximo en S_{ij}.

  - tenemos que c[i,j]=0 cuando $S_{ij}=\varnothing$; en particular c[i,j]=0 para $i \geq j$.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Paso 2: solución recursiva
- Sea **c[i,j]** el número de actividades en un subconjunto de actividades compatibles de tamaño máximo en $\mathbf{S_{ij}}$.
  - tenemos que $c[i,j]=0$ cuando $S_{ij}=\varnothing$; en particular $c[i,j]=0$ para $i \geq j$.
  - en un $S_{ij} \neq \varnothing$ utilizamos los subproblemas S_{ik} y S_{kj} obteniendo la recurrencia:

Algoritmos glotones: **problema de selección de tareas compatibles**

- Paso 2: solución recursiva
- Sea **c[i,j]** el número de actividades en un subconjunto de actividades compatibles de tamaño máximo en $\mathbf{S_{ij}}$.
  - tenemos que $c[i,j]=0$ cuando $S_{ij}=\varnothing$; en particular $c[i,j]=0$ para $i \geq j$.
  - en un $S_{ij} \neq \varnothing$ utilizamos los subproblemas S_{ik} y S_{kj} obteniendo la recurrencia:
  - $$c[i,j] = c[i,k] + c[k,j] + 1$$

Algoritmos glotones: **problema de selección de tareas compatibles**

- Paso 2: solución recursiva
- Sea **c[i,j]** el número de actividades en un subconjunto de actividades compatibles de tamaño máximo en S_{ij}.
  - tenemos que $c[i,j]=0$ cuando $S_{ij}=\varnothing$; en particular $c[i,j]=0$ para $i \geq j$.
  - en un $S_{ij} \neq \varnothing$ utilizamos los subproblemas S_{ik} y S_{kj} obteniendo la recurrencia:
  - $$c[i,j] = c[i,k] + c[k,j] + 1$$
  - pero no conocemos el valor de k y tenemos hasta $j-i-1$ valores posibles, $k=i+1,\ldots,j-1$. Hay que recorrerlos para encontrar el mejor.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Paso 2: solución recursiva
- Sea **c[i,j]** el número de actividades en un subconjunto de actividades compatibles de tamaño máximo en S_{ij}.
  - tenemos que c[i,j]=0 cuando $S_{ij}=\varnothing$; en particular c[i,j]=0 para $i \geq j$.
  - en un $S_{ij} \neq \varnothing$ utilizamos los subproblemas S_{ik} y S_{kj} obteniendo la recurrencia:
  - $$c[i,j] = c[i,k] + c[k,j] + 1$$
  - pero no conocemos el valor de k y tenemos hasta j-i-1 valores posibles, k=i+1,...,j-1. Hay que recorrerlos para encontrar el mejor.

$$c[i,j] = \begin{cases} 0 & \text{si } S_{ij} = \emptyset, \\ \max\limits_{\substack{i<k<j \\ a_k \in S_{ij}}} \{c[i,k] + c[k,j] + 1\} & \text{si } S_{ij} \neq \emptyset, \end{cases}$$

Algoritmos glotones: **problema de selección de tareas compatibles**

$$f_m = \min\{f_k : a_k \in S_{ij}\}$$

Algoritmos glotones: **problema de selección de tareas compatibles**

- Para convertir el algoritmo de programación dinámica en un algoritmo glotón nos basamos en dos observaciones principales:

$$f_m = \min\{f_k : a_k \in S_{ij}\}$$

Algoritmos glotones: **problema de selección de tareas compatibles**

- Para convertir el algoritmo de programación dinámica en un algoritmo glotón nos basamos en dos observaciones principales:
  - Considerando cualquier subproblema no-vacío $\mathbf{S_{ij}}$, y sea $\mathbf{a_m}$ la actividad en $\mathbf{S_{ij}}$ que termina más rápido:

$$f_m = \min\{f_k : a_k \in S_{ij}\}$$

Algoritmos glotones: **problema de selección de tareas compatibles**

- Para convertir el algoritmo de programación dinámica en un algoritmo glotón nos basamos en dos observaciones principales:
  - Considerando cualquier subproblema no-vacío S_{ij}, y sea a_m la actividad en S_{ij} que termina más rápido:

Entonces

$$f_m = \min\{f_k : a_k \in S_{ij}\}$$

1. La actividad a_m es usada en un subconjunto máximo de actividades compatibles en S_{ij}.(Prueba: formar otro conjunto maximo sin a_m y substituir la primera actividad por a_m, y tiene el mismo número de elementos)

2. El subproblema S_{im} está vacío, de tal manera que eligiendo a_m deja al subproblema S_{mj} como el único no-vacío. (Prueba: esto implicaría que existe una a_k tal que $f_i <= s_k < f_k <= s_m < f_m$ y entonces $f_k < f_m$ lo cual es una contradicción)

Algoritmos glotones: **problema de selección de tareas compatibles**

Algoritmos glotones: **problema de selección de tareas compatibles**

- Según vimos en las clases de programación dinámica la estructura sub-óptima varía de acuerdo a:

Algoritmos glotones: **problema de selección de tareas compatibles**

- Según vimos en las clases de programación dinámica la estructura sub-óptima varía de acuerdo a:
  - cuántos sub-problemas se utilizan en la solución óptima.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Según vimos en las clases de programación dinámica la estructura sub-óptima varía de acuerdo a:
  - cuántos sub-problemas se utilizan en la solución óptima.
  - el número de elecciones se tienen para determinar cada sub-problema.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Según vimos en las clases de programación dinámica la estructura sub-óptima varía de acuerdo a:
  - cuántos sub-problemas se utilizan en la solución óptima.
  - el número de elecciones se tienen para determinar cada sub-problema.
- La solución con programación dinámica propone 2 subproblemas y j-i-1 elecciones por sub-problema.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Según vimos en las clases de programación dinámica la estructura sub-óptima varía de acuerdo a:
  - cuántos sub-problemas se utilizan en la solución óptima.
  - el número de elecciones se tienen para determinar cada sub-problema.
- La solución con programación dinámica propone 2 subproblemas y j-i-1 elecciones por sub-problema.
- La solución glotona disminuye el número de sub-problemas (1 subproblema) y este subproblema tiene solo 1 elección: calcular la actividad con el tiempo de finalización más rápido.

Algoritmos glotones: **problema de selección de tareas compatibles**

Algoritmos glotones: **problema de selección de tareas compatibles**

- Para resolver el subproblema S_{ij}:

Algoritmos glotones: **problema de selección de tareas compatibles**

- Para resolver el subproblema S_{ij}:
  - elegimos la actividad a_m en S_{ij} que termina más rápido y

Algoritmos glotones: **problema de selección de tareas compatibles**

- Para resolver el subproblema S_{ij}:
  - elegimos la actividad a_m en S_{ij} que termina más rápido y
  - sumamos esta solución al conjunto de actividades usando la solución del problema $S_{m,j}$.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Para resolver el subproblema S_{ij}:
  - elegimos la actividad a_m en S_{ij} que termina más rápido y
  - sumamos esta solución al conjunto de actividades usando la solución del problema $S_{m,j}$.
- Se observa un patrón en los problemas resueltos: el problema original es $S_{0,n+1}$.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Para resolver el subproblema S_{ij}:
  - elegimos la actividad a_m en S_{ij} que termina más rápido y
  - sumamos esta solución al conjunto de actividades usando la solución del problema $S_{m,j}$.
- Se observa un patrón en los problemas resueltos: el problema original es $S_{0,n+1}$.
- Elegimos la actividad a_{m1} como actividad en $S_{0,n+1}$ que se termina más rápido.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Para resolver el subproblema S_{ij}:
  - elegimos la actividad a_m en S_{ij} que termina más rápido y
  - sumamos esta solución al conjunto de actividades usando la solución del problema $S_{m,j}$.
- Se observa un patrón en los problemas resueltos: el problema original es $S_{0,n+1}$.
- Elegimos la actividad a_{m1} como actividad en $S_{0,n+1}$ que se termina más rápido.
- El siguiente subproblema es $S_{m1,n+1}$. Elegimos a_{m2} como actividad,

Algoritmos glotones: **problema de selección de tareas compatibles**

- Para resolver el subproblema S_{ij}:
  - elegimos la actividad a_m en S_{ij} que termina más rápido y
  - sumamos esta solución al conjunto de actividades usando la solución del problema $S_{m,j}$.
- Se observa un patrón en los problemas resueltos: el problema original es $S_{0,n+1}$.
- Elegimos la actividad a_{m1} como actividad en $S_{0,n+1}$ que se termina más rápido.
- El siguiente subproblema es $S_{m1,n+1}$. Elegimos a_{m2} como actividad,
- Notamos que cada subproblema es de la forma $S_{mi,n+1}$ cuando una actividad m_i es seleccionada.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Para resolver el subproblema S_{ij}:
  - elegimos la actividad a_m en S_{ij} que termina más rápido y
  - sumamos esta solución al conjunto de actividades usando la solución del problema $S_{m,j}$.
- Se observa un patrón en los problemas resueltos: el problema original es $S_{0,n+1}$.
- Elegimos la actividad a_{m1} como actividad en $S_{0,n+1}$ que se termina más rápido.
- El siguiente subproblema es $S_{m1,n+1}$. Elegimos a_{m2} como actividad,
- Notamos que cada subproblema es de la forma $S_{mi,n+1}$ cuando una actividad m_i es seleccionada.
- El algoritmo glotón deja la mayor oportunidad para programar las actividades que quedan: maximiza la cantidad de tiempo restante sin programar.

Algoritmos glotones: **problema de selección de tareas compatibles**

Algoritmos glotones: **problema de selección de tareas compatibles**

- Algoritmo glotón recursivo (top-down)

Algoritmos glotones: **problema de selección de tareas compatibles**

- Algoritmo glotón recursivo (top-down)
  - Entradas:

Algoritmos glotones: **problema de selección de tareas compatibles**

- Algoritmo glotón recursivo (top-down)
  - Entradas:
    - conjuntos de tiempos de inicio **s** y fin **f** (ordenadas incrementalmente).

Algoritmos glotones: **problema de selección de tareas compatibles**

- Algoritmo glotón recursivo (top-down)
  - Entradas:
    - conjuntos de tiempos de inicio **s** y fin **f** (ordenadas incrementalmente).
    - **i** y **n** que definen el subproblema $S_{i,n+1}$ a resolver.

Algoritmos glotones: **problema de selección de tareas compatibles**

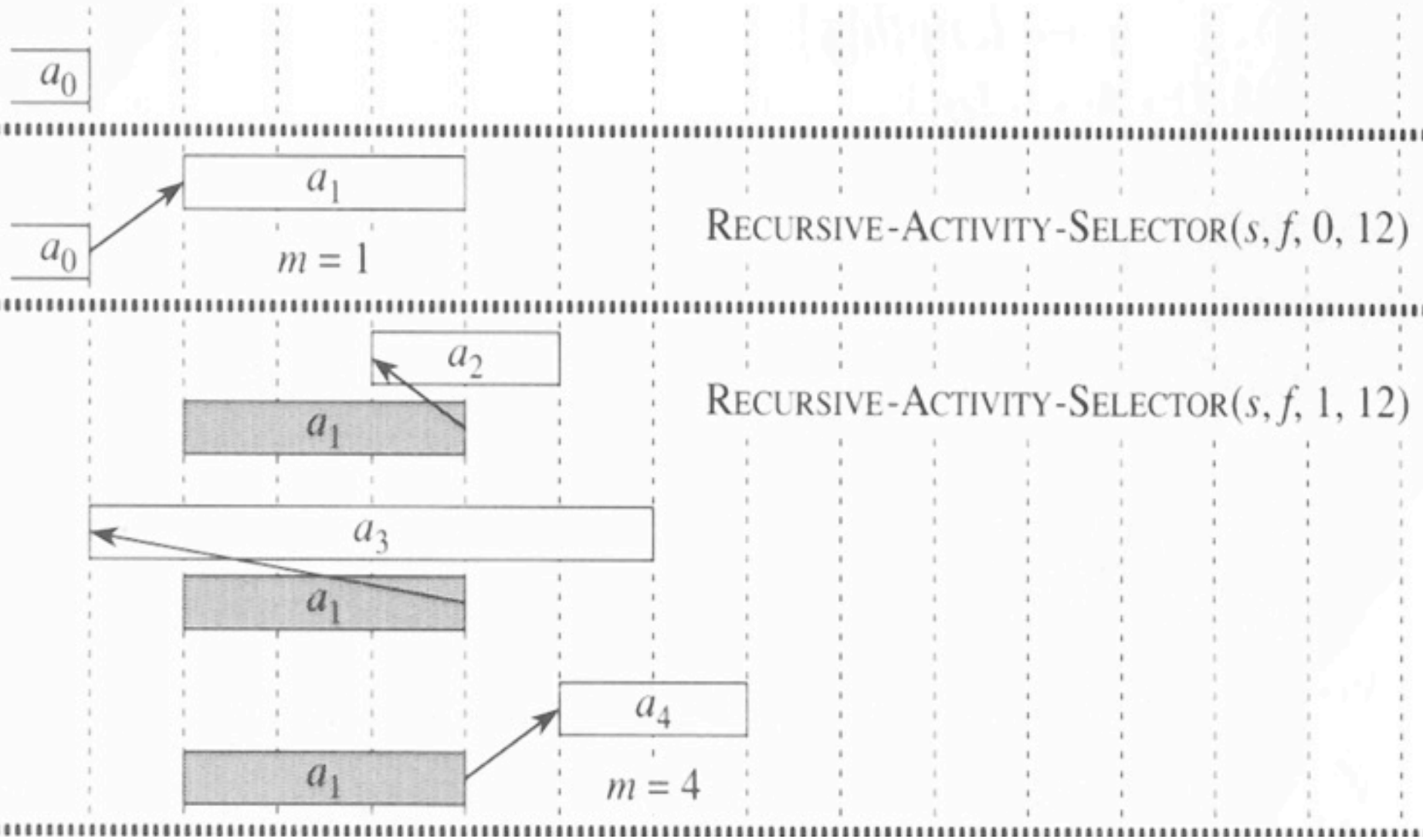
- Algoritmo glotón recursivo (top-down)
  - Entradas:
    - conjuntos de tiempos de inicio **s** y fin **f** (ordenadas incrementalmente).
    - **i** y **n** que definen el subproblema $S_{i,n+1}$ a resolver.
  - Salida:

Algoritmos glotones: **problema de selección de tareas compatibles**

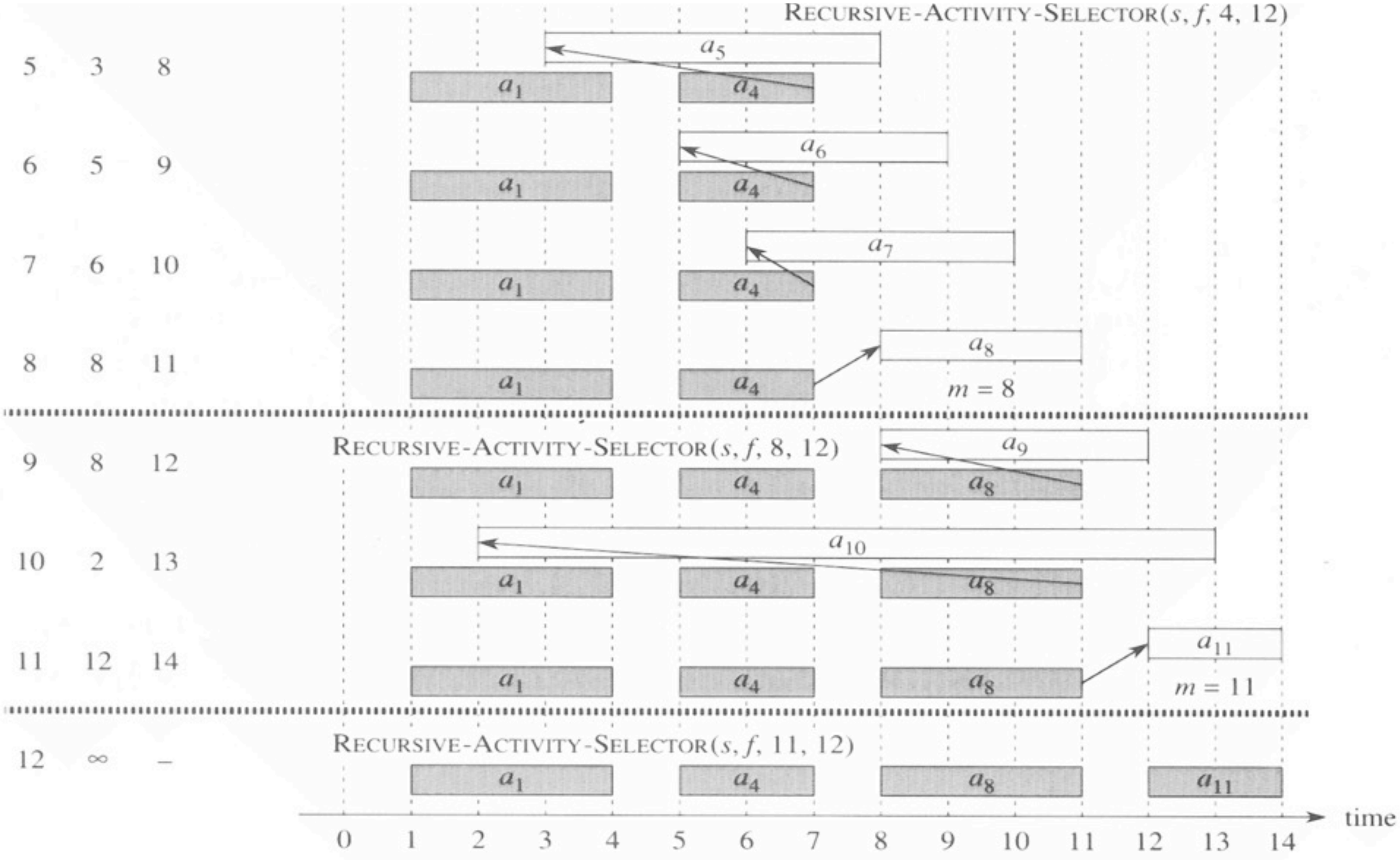
- Algoritmo glotón recursivo (top-down)
  - Entradas:
    - conjuntos de tiempos de inicio **s** y fin **f** (ordenadas incrementalmente).
    - **i** y **n** que definen el subproblema $S_{i,n+1}$ a resolver.
  - Salida:
    - conjunto más grande de actividades compatibles en $S_{i,n+1}$.

Algoritmos glotones: **problema de selección de tareas compatibles**

k	s_k	f_k
0	–	0
1	1	4
2	3	5
3	0	6
4	5	7



Algoritmos glotones: **problema de selección de tareas compatibles**



Algoritmos glotones: **problema de selección de tareas compatibles**

Algoritmos glotones: problema de selección de tareas compatibles

- **RECURSIVE-ACTIVITY-SELECTOR(s,f,i,n)**

Algoritmos glotones: problema de selección de tareas compatibles

- **RECURSIVE-ACTIVITY-SELECTOR(s,f,i,n)**

1. m ← i+1

Algoritmos glotones: problema de selección de tareas compatibles

- **RECURSIVE-ACTIVITY-SELECTOR(s,f,i,n)**

1. $m \leftarrow i+1$
2. **while** $m \leq n$ y $s_m < f_i$

Algoritmos glotones: problema de selección de tareas compatibles

- **RECURSIVE-ACTIVITY-SELECTOR(s,f,i,n)**

1. m ← i+1
2. **while** m ≤ n y s_m < f_i
3. **do** m ← m+1

Algoritmos glotones: problema de selección de tareas compatibles

```
•   RECURSIVE-ACTIVITY-SELECTOR(s,f,i,n)
1.  m ← i+1
2.  while m ≤ n y s_m < f_i
3.      do m ← m+1
4.  if  m ≤ n
```

Algoritmos glotones: **problema de selección de tareas compatibles**

- **RECURSIVE-ACTIVITY-SELECTOR(s,f,i,n)**

1. m ← i+1
2. **while** m ≤ n y s_m < f_i
3. **do** m ← m+1
4. **if** m ≤ n
5. **then return** $\{a_m\}$ U RECURSIVE-ACTIVITY-SELECTOR(s,f,m,n)

Algoritmos glotones: **problema de selección de tareas compatibles**

- **RECURSIVE-ACTIVITY-SELECTOR(s,f,i,n)**

1. m ← i+1
2. **while** m ≤ n y s_m < f_i
3. **do** m ← m+1
4. **if** m ≤ n
5. **then return** {a_m} U RECURSIVE-ACTIVITY-SELECTOR(s,f,m,n)
6. **else return** NULL.

Algoritmos glotones: **problema de selección de tareas compatibles**

Algoritmos glotones: **problema de selección de tareas compatibles**

- Suponiendo que las actividades han sido ordenadas respecto a su tiempo final, el tiempo de ejecución de la llamada RECURSIVE-ACTIVITY-SELECTOR(s,f,0,n) es de $\Theta(n)$.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Suponiendo que las actividades han sido ordenadas respecto a su tiempo final, el tiempo de ejecución de la llamada RECURSIVE-ACTIVITY-SELECTOR(s,f,0,n) es de $\Theta(n)$.

- En las llamadas recursivas cada actividad a_k se examina una sola vez en el ciclo **while** de la linea 2.

Algoritmos glotones: **problema de selección de tareas compatibles**

- Suponiendo que las actividades han sido ordenadas respecto a su tiempo final, el tiempo de ejecución de la llamada RECURSIVE-ACTIVITY-SELECTOR(s,f,0,n) es de $\Theta(n)$.

- En las llamadas recursivas cada actividad a_k se examina una sola vez en el ciclo **while** de la linea 2.

- Se puede escribir la versión iterativa (hacerlo de tarea)

Algoritmos glotones: **elementos**

Algoritmos glotones: **elementos**

- Los algoritmos glotones obtienen una solución óptima realizando una secuencia de elecciones.

Algoritmos glotones: **elementos**

- Los algoritmos glotones obtienen una solución óptima realizando una secuencia de elecciones.
- En cada punto del algoritmo se toma la mejor solución en ese momento.

Algoritmos glotones: **elementos**

- Los algoritmos glotones obtienen una solución óptima realizando una secuencia de elecciones.
- En cada punto del algoritmo se toma la mejor solución en ese momento.
- Para el ejemplo anterior seguimos los pasos:

Algoritmos glotones: **elementos**

- Los algoritmos glotones obtienen una solución óptima realizando una secuencia de elecciones.
- En cada punto del algoritmo se toma la mejor solución en ese momento.
- Para el ejemplo anterior seguimos los pasos:
  - Determinar la estructura subóptima del problema.

Algoritmos glotones: **elementos**

- Los algoritmos glotones obtienen una solución óptima realizando una secuencia de elecciones.
- En cada punto del algoritmo se toma la mejor solución en ese momento.
- Para el ejemplo anterior seguimos los pasos:
  - Determinar la estructura subóptima del problema.
  - Desarrollar una solución recursiva.

Algoritmos glotones: **elementos**

- Los algoritmos glotones obtienen una solución óptima realizando una secuencia de elecciones.
- En cada punto del algoritmo se toma la mejor solución en ese momento.
- Para el ejemplo anterior seguimos los pasos:
  - Determinar la estructura subóptima del problema.
  - Desarrollar una solución recursiva.
  - Probar que, en cualquier punto de la recursión, una de las elecciones óptimas es la opción glotona.

Algoritmos glotones: **elementos**

- Los algoritmos glotones obtienen una solución óptima realizando una secuencia de elecciones.
- En cada punto del algoritmo se toma la mejor solución en ese momento.
- Para el ejemplo anterior seguimos los pasos:
  - Determinar la estructura subóptima del problema.
  - Desarrollar una solución recursiva.
  - Probar que, en cualquier punto de la recursión, una de las elecciones óptimas es la opción glotona.
  - Mostrar que todos excepto uno de los subproblemas inducidos por hacer la elección glotona están vacíos.

Algoritmos glotones: **elementos**

Algoritmos glotones: **elementos**

- Desarrollar un algoritmo recursivo que implemente la estrategia glotona.

Algoritmos glotones: **elementos**

- Desarrollar un algoritmo recursivo que implemente la estrategia glotona.
- Convertir el algoritmo recursivo en uno iterativo.

Algoritmos glotones: **elementos**

Algoritmos glotones: **elementos**

- Generalmente se diseña un algoritmo glotón con la opción glotona en mente siguiendo los pasos:

Algoritmos glotones: **elementos**

- Generalmente se diseña un algoritmo glotón con la opción glotona en mente siguiendo los pasos:
  - Especificar el problema de manera que al hacer una elección quede solo un subproblema por resolver.

Algoritmos glotones: **elementos**

- Generalmente se diseña un algoritmo glotón con la opción glotona en mente siguiendo los pasos:
  - Especificar el problema de manera que al hacer una elección quede solo un subproblema por resolver.
  - Probar que siempre hay una solución optima al problema original tomando la opción glotona.

Algoritmos glotones: **elementos**

- Generalmente se diseña un algoritmo glotón con la opción glotona en mente siguiendo los pasos:
  - Especificar el problema de manera que al hacer una elección quede solo un subproblema por resolver.
  - Probar que siempre hay una solución optima al problema original tomando la opción glotona.
  - Demostrar que, una vez tomada la opción glotona, lo que resulta es un subproblema con la propiedad que si se combinan las solución óptima del subproblema con la elección glotona se puede llegar a la solución óptima del problema original.

Algoritmos glotones: **elementos**

- Generalmente se diseña un algoritmo glotón con la opción glotona en mente siguiendo los pasos:
  - Especificar el problema de manera que al hacer una elección quede solo un subproblema por resolver.
  - Probar que siempre hay una solución optima al problema original tomando la opción glotona.
  - Demostrar que, una vez tomada la opción glotona, lo que resulta es un subproblema con la propiedad que si se combinan las solución óptima del subproblema con la elección glotona se puede llegar a la solución óptima del problema original.
- La elección glotona no depende de la resolución del subproblema.

Algoritmos glotones: **elementos**

Algoritmos glotones: **elementos**

- No hay una forma general para decir si la solución glotona es optima pero los ingredientes principales son:

Algoritmos glotones: **elementos**

- No hay una forma general para decir si la solución glotona es optima pero los ingredientes principales son:
  - Propiedad de la opción glotona: Una solución optima globalmente se puede alcanzar realizando la elección optima localmente.

Algoritmos glotones: **elementos**

- No hay una forma general para decir si la solución glotona es optima pero los ingredientes principales son:
  - Propiedad de la opción glotona: Una solución optima globalmente se puede alcanzar realizando la elección optima localmente.
  - Exhibe Subestructura óptima