MC458 — Projeto e Análise de Algoritmos I

C.C. de Souza C.N. da Silva O. Lee

Antes de mais nada...

- Uma versão anterior deste conjunto de slides foi preparada por Cid Carvalho de Souza e Cândida Nunes da Silva para uma instância anterior desta disciplina.
- O que vocês tem em mãos é uma versão modificada preparada para atender a meus gostos.
- Nunca é demais enfatizar que o material é apenas um guia e não deve ser usado como única fonte de estudo. Para isso consultem a bibliografia (em especial o CLR ou CLRS).

Orlando Lee

Agradecimentos (Cid e Cândida)

- Várias pessoas contribuíram direta ou indiretamente com a preparação deste material.
- Algumas destas pessoas cederam gentilmente seus arquivos digitais enquanto outras cederam gentilmente o seu tempo fazendo correções e dando sugestões.
- Uma lista destes "colaboradores" (em ordem alfabética) é dada abaixo:
  - Célia Picinin de Mello
  - José Coelho de Pina
  - Orlando Lee
  - Paulo Feofiloff
  - Pedro Rezende
  - Ricardo Dahab
  - Zanoni Dias

Algoritmos gulosos

Algoritmos Gulosos

Algoritmos Gulosos

- Tipicamente algoritmos gulosos são utilizados para resolver problemas de **otimização**.

Algoritmos Gulosos

- Tipicamente algoritmos gulosos são utilizados para resolver problemas de **otimização**.
- Uma característica comum dos problemas onde se aplicam algoritmos gulosos é a existência de **subestrutura ótima**, semelhante à programação dinâmica.

Algoritmos Gulosos

- Tipicamente algoritmos gulosos são utilizados para resolver problemas de **otimização**.
- Uma característica comum dos problemas onde se aplicam algoritmos gulosos é a existência de **subestrutura ótima**, semelhante à programação dinâmica.
- Programação dinâmica: os subproblemas são resolvidos **antes** de se proceder à escolha de um elemento que irá compor a solução ótima.

Algoritmos Gulosos

- Tipicamente algoritmos gulosos são utilizados para resolver problemas de **otimização**.
- Uma característica comum dos problemas onde se aplicam algoritmos gulosos é a existência de **subestrutura ótima**, semelhante à programação dinâmica.
- Programação dinâmica: os subproblemas são resolvidos **antes** de se proceder à escolha de um elemento que irá compor a solução ótima.
- Algoritmo Guloso: escolhe-se um elemento que irá compor a solução ótima e **depois** um subproblema é resolvido.

Algoritmos Gulosos

Algoritmos Gulosos

- Um algoritmo guloso sempre faz a escolha que parece ser a *"melhor"* a cada iteração usando um **critério guloso**.

Algoritmos Gulosos

- Um algoritmo guloso sempre faz a escolha que parece ser a *"melhor"* a cada iteração usando um **critério guloso**.
- Propriedade da **escolha gulosa**: garante que a cada iteração a escolha feita leva a uma solução ótima.

Algoritmos Gulosos

- Um algoritmo guloso sempre faz a escolha que parece ser a *"melhor"* a cada iteração usando um **critério guloso**.
- Propriedade da **escolha gulosa**: garante que a cada iteração a escolha feita leva a uma solução ótima.
- Em geral é fácil projetar ou descrever um algoritmo guloso.

Algoritmos Gulosos

- Um algoritmo guloso sempre faz a escolha que parece ser a *"melhor"* a cada iteração usando um **critério guloso**.
- Propriedade da **escolha gulosa**: garante que a cada iteração a escolha feita leva a uma solução ótima.
- Em geral é fácil projetar ou descrever um algoritmo guloso.
- O **difícil** é provar que ele funciona!

Seleção de Atividades

Seleção de Atividades

- $S = \{a_1, \ldots, a_n\}$: conjunto de n atividades.

Seleção de Atividades

- $S = \{a_1, \ldots, a_n\}$: conjunto de n atividades.
- Para todo $i = 1, \ldots, n$, a atividade a_i começa no instante s_i e termina no instante f_i, com $0 \leq s_i < f_i < \infty$.

Seleção de Atividades

- $S = \{a_1, \ldots, a_n\}$: conjunto de n atividades.
- Para todo $i = 1, \ldots, n$, a atividade a_i começa no instante s_i e termina no instante f_i, com $0 \leq s_i < f_i < \infty$.
  Ou seja, supõe-se que a atividade a_i será executada no intervalo de tempo (semi-aberto) $[s_i, f_i)$.

Seleção de Atividades

- $S = \{a_1, \ldots, a_n\}$: conjunto de n atividades.
- Para todo $i = 1, \ldots, n$, a atividade a_i começa no instante s_i e termina no instante f_i, com $0 \leq s_i < f_i < \infty$.
  Ou seja, supõe-se que a atividade a_i será executada no intervalo de tempo (semi-aberto) $[s_i, f_i)$.

Definição

As atividades a_i e a_j são ditas **compatíveis** se os intervalos $[s_i, f_i)$ e $[s_j, f_j)$ são disjuntos.

Seleção de Atividades

- $S = \{a_1, \ldots, a_n\}$: conjunto de n atividades.
- Para todo $i = 1, \ldots, n$, a atividade a_i começa no instante s_i e termina no instante f_i, com $0 \leq s_i < f_i < \infty$.
  Ou seja, supõe-se que a atividade a_i será executada no intervalo de tempo (semi-aberto) $[s_i, f_i)$.

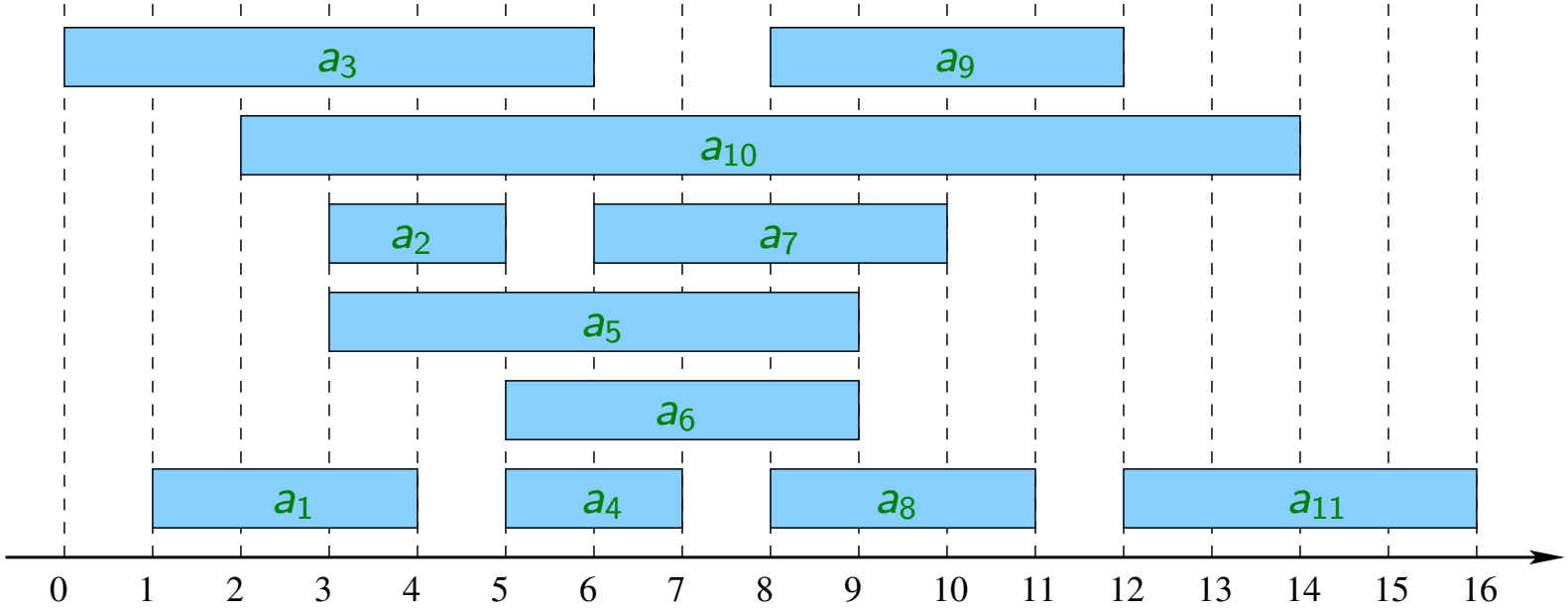
Definição

As atividades a_i e a_j são ditas **compatíveis** se os intervalos $[s_i, f_i)$ e $[s_j, f_j)$ são disjuntos.

Problema de Seleção de Atividades

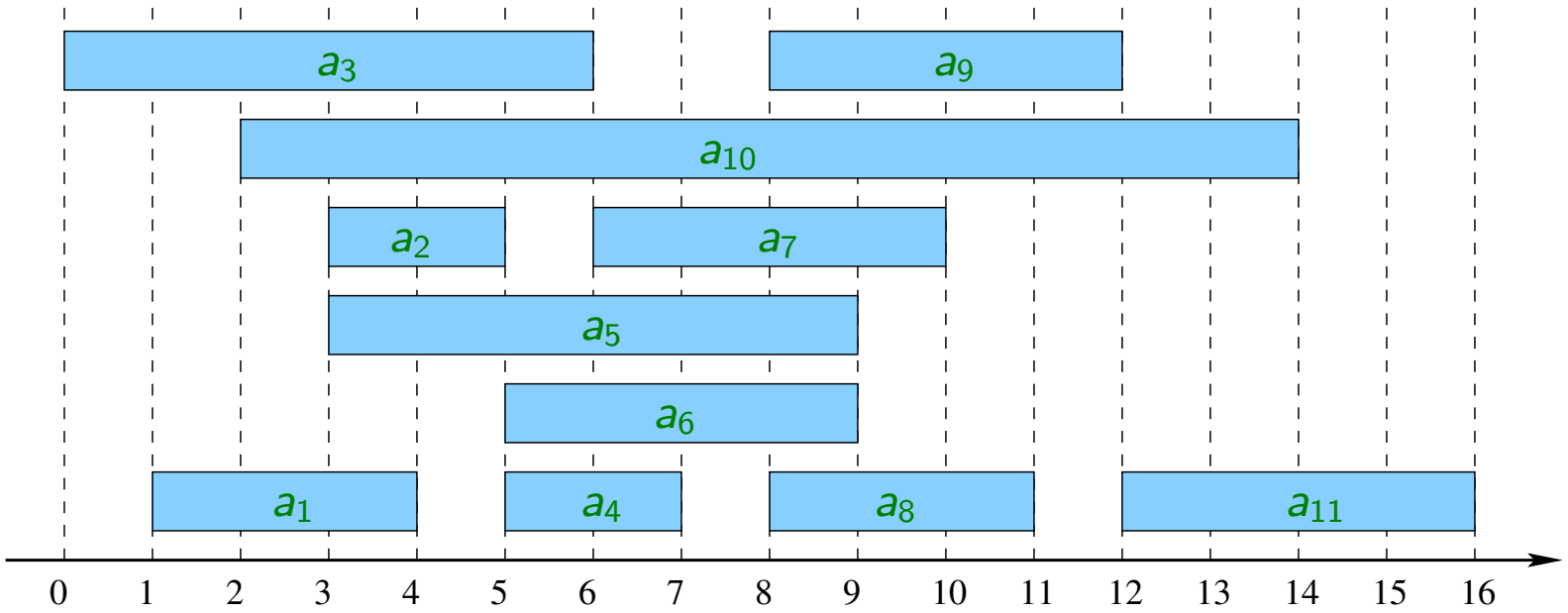
Encontre em S um subconjunto de atividades mutuamente compatíveis que tenha tamanho **máximo**.

Seleção de Atividades



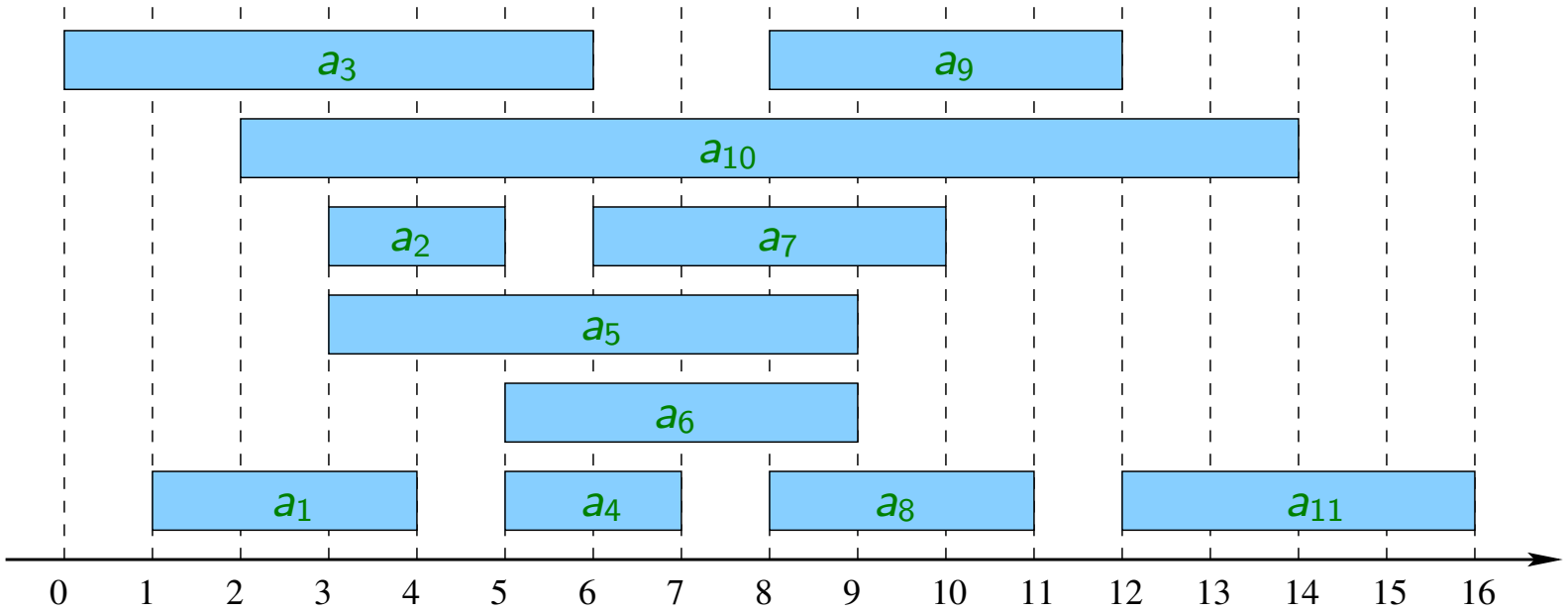
i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	4	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

Seleção de Atividades



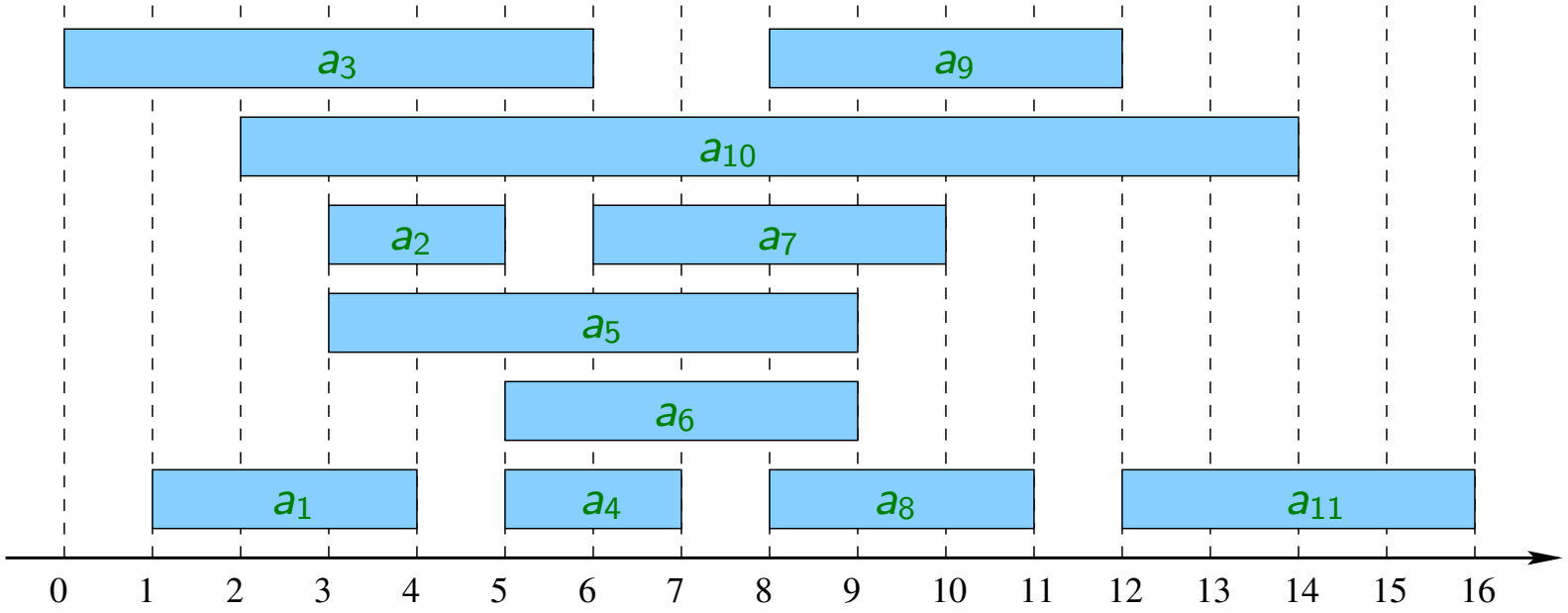
- Pares de atividades incompatíveis: (a_1, a_2), (a_1, a_3), (a_5, a_9)
  Pares de atividades compatíveis: (a_1, a_9), (a_4, a_8), (a_2, a_6)

Seleção de Atividades



- Conjunto **maximal** de atividades compatíveis: (a_3, a_9, a_{11}).
- Conjunto **máximo** de atividades compatíveis: (a_1, a_4, a_8, a_{11}).

Seleção de Atividades



Observação:

As atividades estão ordenadas em **ordem crescente** de instantes de término. Isso será importante mais adiante.

Seleção de Atividades

Seleção de Atividades

- Tanto os algoritmos gulosos quanto os de programação dinâmica valem-se da propriedade de subestrutura ótima.

Seleção de Atividades

- Tanto os algoritmos gulosos quanto os de programação dinâmica valem-se da propriedade de subestrutura ótima.
- Primeiro mostraremos que o problema da seleção de atividades tem esta propriedade e, então, projetaremos um algoritmo por programação dinâmica.

Seleção de Atividades

- Tanto os algoritmos gulosos quanto os de programação dinâmica valem-se da propriedade de subestrutura ótima.
- Primeiro mostraremos que o problema da seleção de atividades tem esta propriedade e, então, projetaremos um algoritmo por programação dinâmica.
- Em seguida, mostraremos que há uma forma de resolver o problema usando um algoritmo guloso que examina um número muito menor de subproblemas.

Seleção de Atividades

- Tanto os algoritmos gulosos quanto os de programação dinâmica valem-se da propriedade de subestrutura ótima.
- Primeiro mostraremos que o problema da seleção de atividades tem esta propriedade e, então, projetaremos um algoritmo por programação dinâmica.
- Em seguida, mostraremos que há uma forma de resolver o problema usando um algoritmo guloso que examina um número muito menor de subproblemas.
- Este processo auxiliará no entendimento da diferença entre estas duas **técnicas de projeto de algoritmos**.

Seleção de Atividades

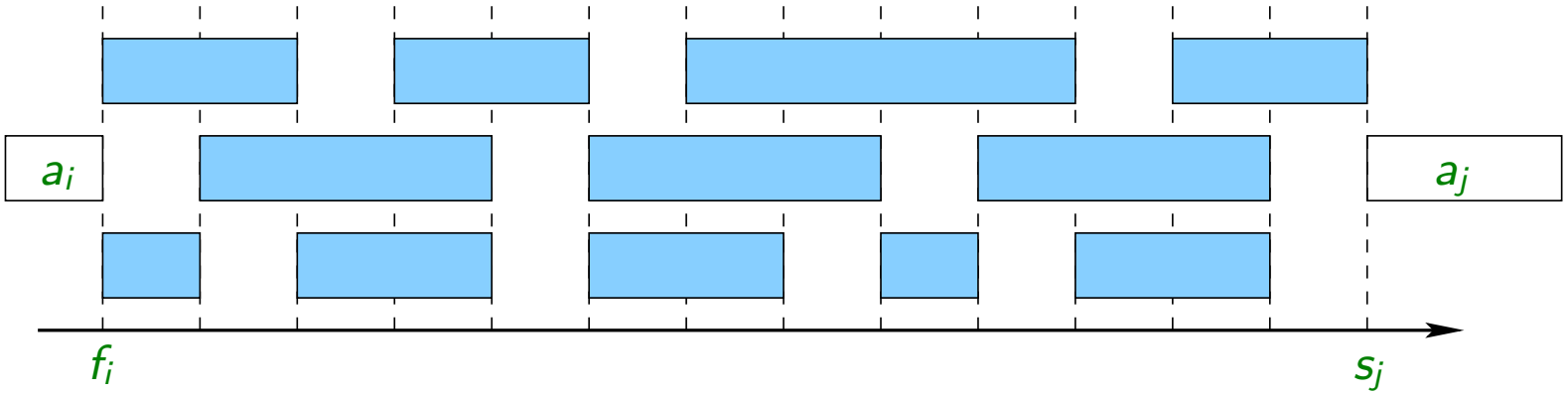
Suponha que $f_1 \leq f_2 \leq \ldots \leq f_n$, ou seja, as atividades estão ordenadas em ordem crescente de instantes de término.

Seleção de Atividades

Suponha que $f_1 \leq f_2 \leq \ldots \leq f_n$, ou seja, as atividades estão ordenadas em ordem crescente de instantes de término.

Definição

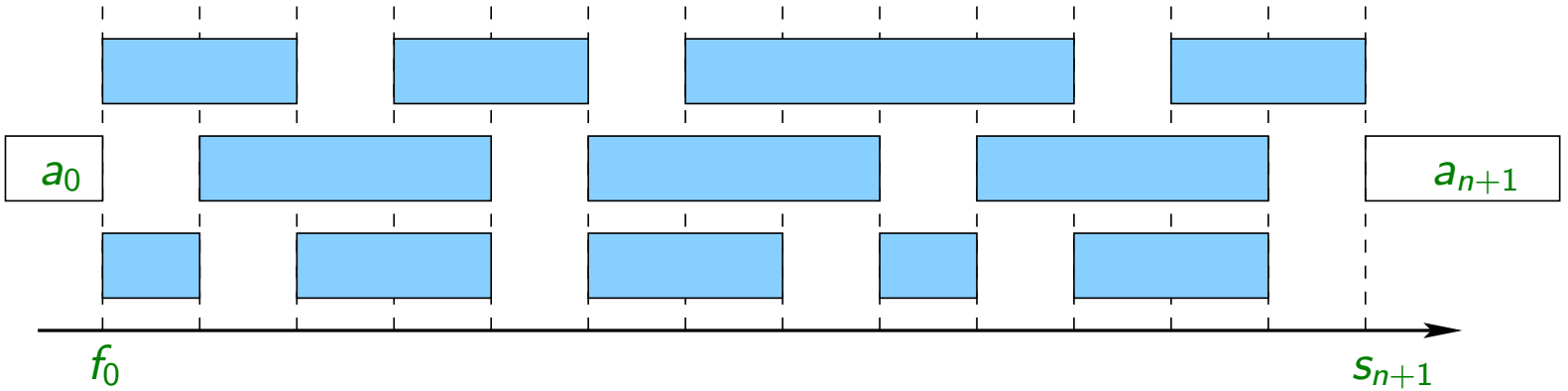
Seja $S_{i,j} = \{a_k \in S : f_i \leq s_k < f_k \leq s_j\}$, i.e o conjunto de atividades que começam depois do término de a_i e terminam antes do início de a_j.



Seleção de Atividades

Definição

Seja $S_{i,j} = \{a_k \in S : f_i \leq s_k < f_k \leq s_j\}$, i.e o conjunto de atividades que começam depois do término de a_i e terminam antes do início de a_j.

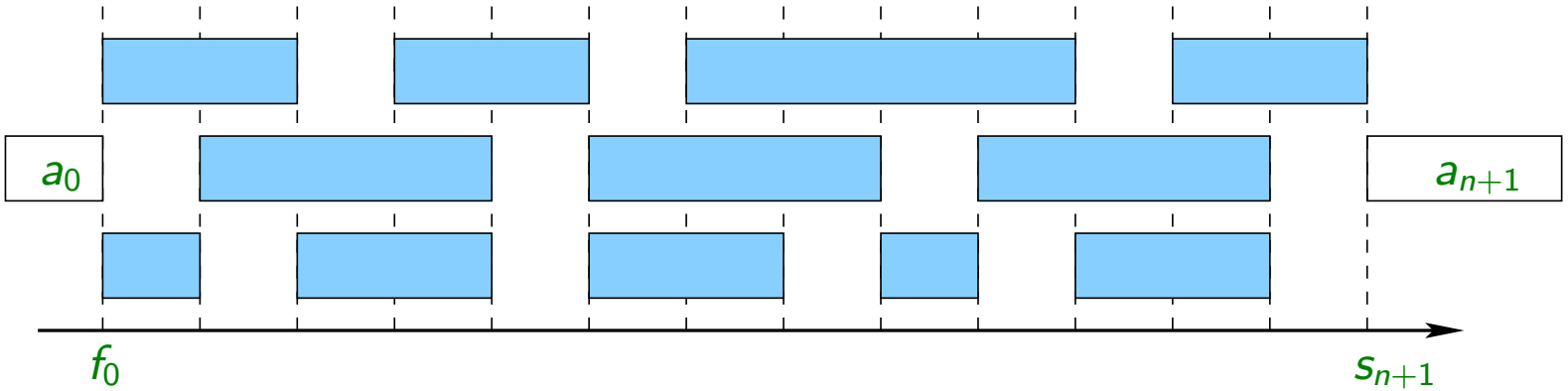


- Atividades artificiais: a_0 com $f_0 = 0$ e a_{n+1} com $s_{n+1} = f_n$.

Seleção de Atividades

Definição

Seja $S_{i,j} = \{a_k \in S : f_i \leq s_k < f_k \leq s_j\}$, i.e o conjunto de atividades que começam depois do término de a_i e terminam antes do início de a_j.

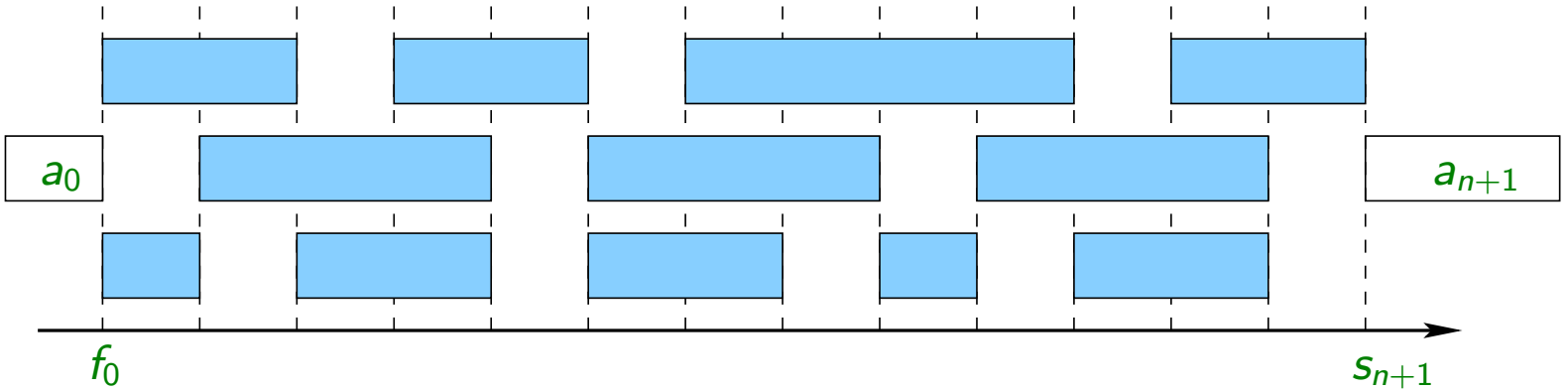


- Atividades artificiais: a_0 com $f_0 = 0$ e a_{n+1} com $s_{n+1} = f_n$.
- Temos que $S = S_{0,n+1}$ e, com isso, $S_{i,j}$ está bem definido para qualquer par (i, j) tal que $0 \leq i, j \leq n + 1$.

Seleção de Atividades

Definição

Seja $S_{i,j} = \{a_k \in S : f_i \leq s_k < f_k \leq s_j\}$, i.e o conjunto de atividades que começam depois do término de a_i e terminam antes do início de a_j.



- Atividades artificiais: a_0 com $f_0 = 0$ e a_{n+1} com $s_{n+1} = f_n$.
- Temos que $S = S_{0,n+1}$ e, com isso, $S_{i,j}$ está bem definido para qualquer par (i, j) tal que $0 \leq i, j \leq n + 1$.
- Note que $S_{i,j} = \emptyset$ para todo $i \geq j$. (**Por quê?**)

Seleção de Atividades - Subestrutura Ótima

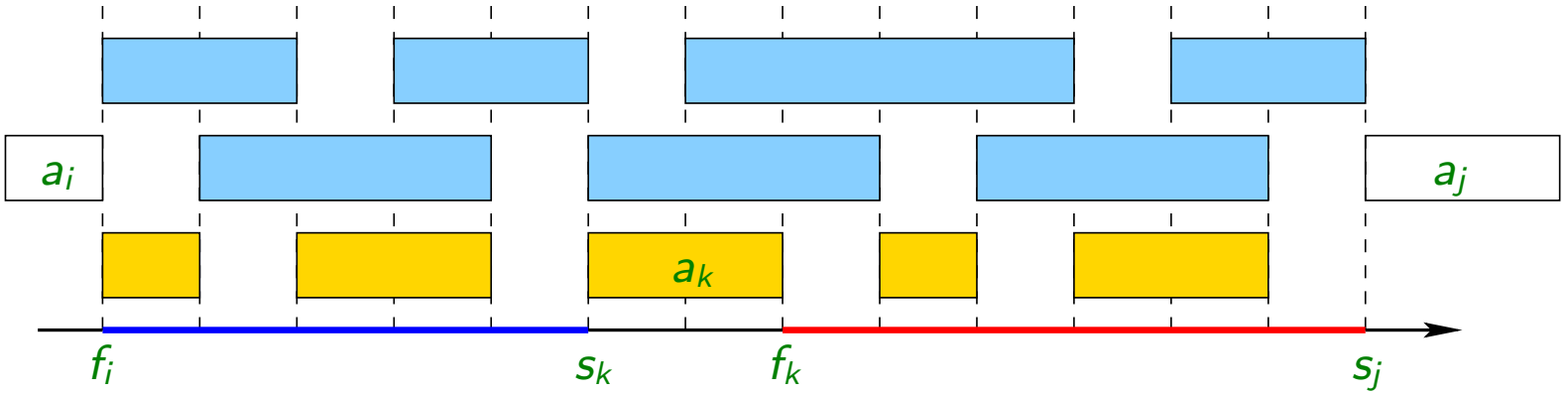
Seleção de Atividades - Subestrutura Ótima

- Considere o subproblema da seleção de atividades para $S_{i,j}$. Suponha que a_k pertence a uma solução ótima de $S_{i,j}$.

Seleção de Atividades - Subestrutura Ótima

- Considere o subproblema da seleção de atividades para $S_{i,j}$. Suponha que a_k pertence a uma solução ótima de $S_{i,j}$.

  Como $f_i \leq s_k < f_k \leq s_j$, uma solução ótima para $S_{i,j}$ que contém a_k é composta pelas atividades de uma solução ótima de S_{ik}, pelas atividades de uma solução ótima de S_{kj} e por a_k. **Por quê?**



Seleção de Atividades

- **Definição**: para todo $0 \leq i, j \leq n+1$, seja $c[i,j]$ o valor ótimo do problema de seleção de atividades de $S_{i,j}$.

Seleção de Atividades

- **Definição**: para todo $0 \leq i, j \leq n+1$, seja $c[i, j]$ o valor ótimo do problema de seleção de atividades de $S_{i,j}$.
- Deste modo, o valor ótimo do problema de seleção de atividades para instância $S = S_{0,n+1}$ é $c[0, n+1]$.

Seleção de Atividades

- **Definição**: para todo $0 \leq i, j \leq n+1$, seja $c[i,j]$ o valor ótimo do problema de seleção de atividades de $S_{i,j}$.
- Deste modo, o valor ótimo do problema de seleção de atividades para instância $S = S_{0,n+1}$ é $c[0, n+1]$.
- **Fórmula de recorrência**:

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

Seleção de Atividades

- **Definição**: para todo $0 \leq i, j \leq n+1$, seja $c[i,j]$ o valor ótimo do problema de seleção de atividades de $S_{i,j}$.
- Deste modo, o valor ótimo do problema de seleção de atividades para instância $S = S_{0,n+1}$ é $c[0, n+1]$.
- **Fórmula de recorrência**:

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

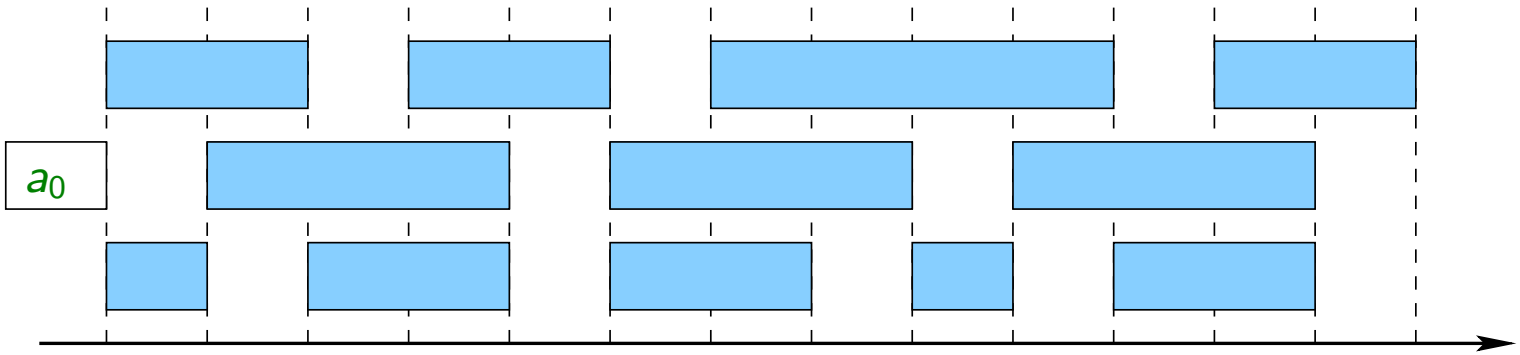
Agora é fácil escrever o algoritmo de programação dinâmica. (Exercício.)

Seleção de Atividades – Escolha Gulosa

- No algoritmo de programação dinâmica (ou na recorrência), precisamos primeiro resolver os subproblemas e depois fazer a escolha da atividade a acrescentar à nossa solução.

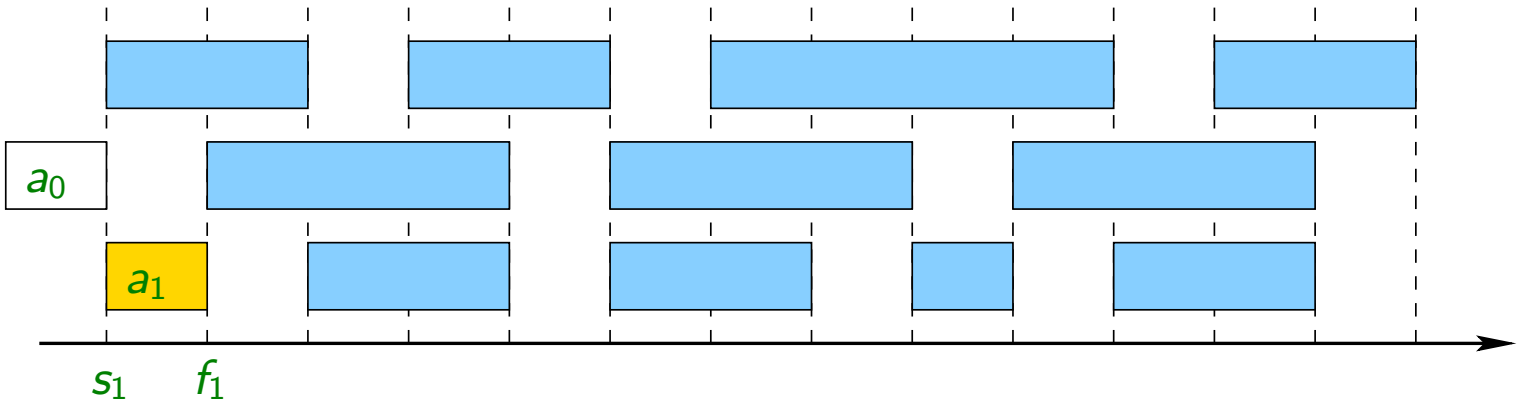
Seleção de Atividades – Escolha Gulosa

- No algoritmo de programação dinâmica (ou na recorrência), precisamos primeiro resolver os subproblemas e depois fazer a escolha da atividade a acrescentar à nossa solução.
- Seria possível **escolher** uma atividade que garantidamente pertence a alguma solução ótima? **Alguma sugestão?**

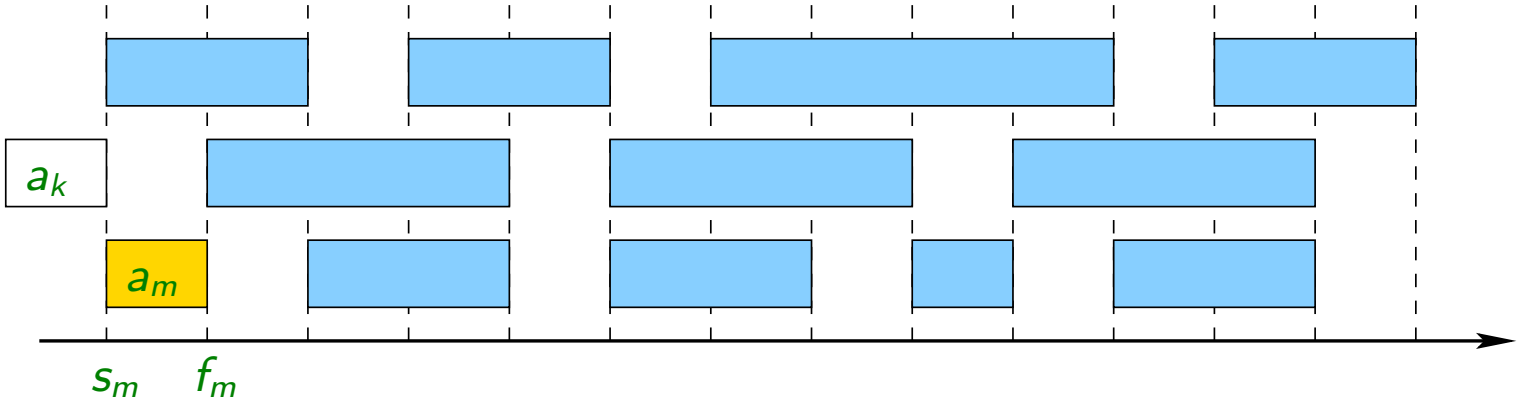


Seleção de Atividades – Escolha Gulosa

- No algoritmo de programação dinâmica (ou na recorrência), precisamos primeiro resolver os subproblemas e depois fazer a escolha da atividade a acrescentar à nossa solução.
- Seria possível **escolher** uma atividade que garantidamente pertence a alguma solução ótima? **Alguma sugestão?**
- Intuitivamente, uma escolha gulosa óbvia para o problema S é escolher uma atividade m com menor instante de término f_m, ou seja, a_1.

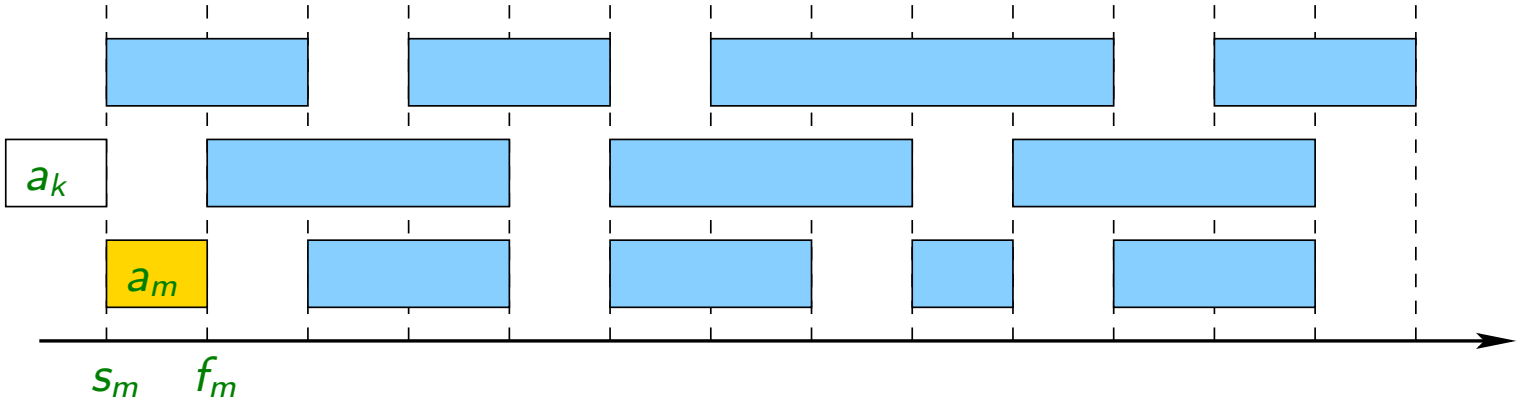


Seleção de Atividades – Escolha Gulosa



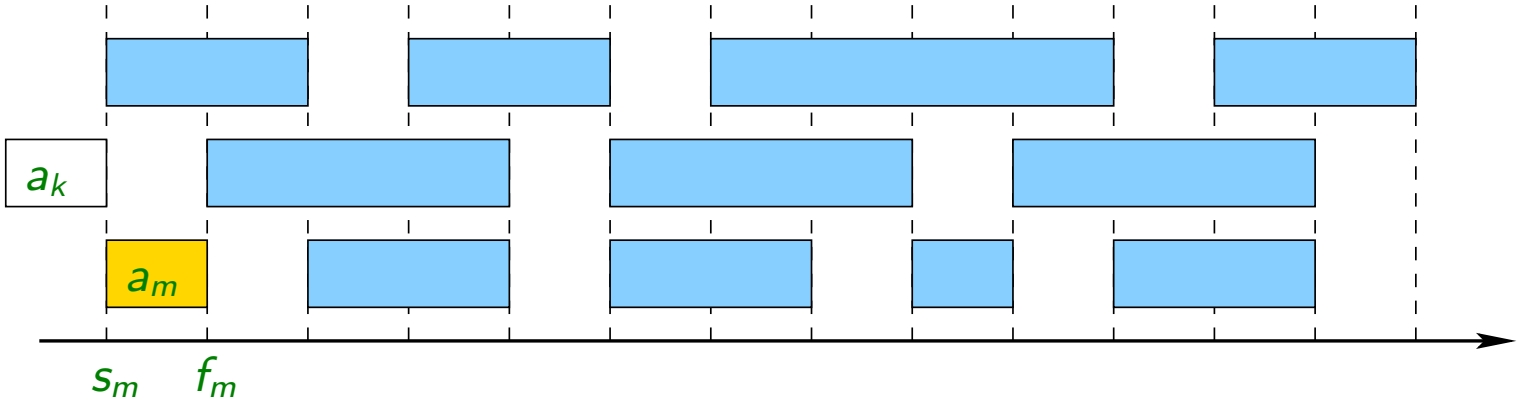
- Seja $S_k = \{a_i \in S : s_i \geq f_k\}$. Ou seja, S_k é o conjunto de atividades de S que começam depois de f_k, o instante em que a_k termina.

Seleção de Atividades – Escolha Gulosa



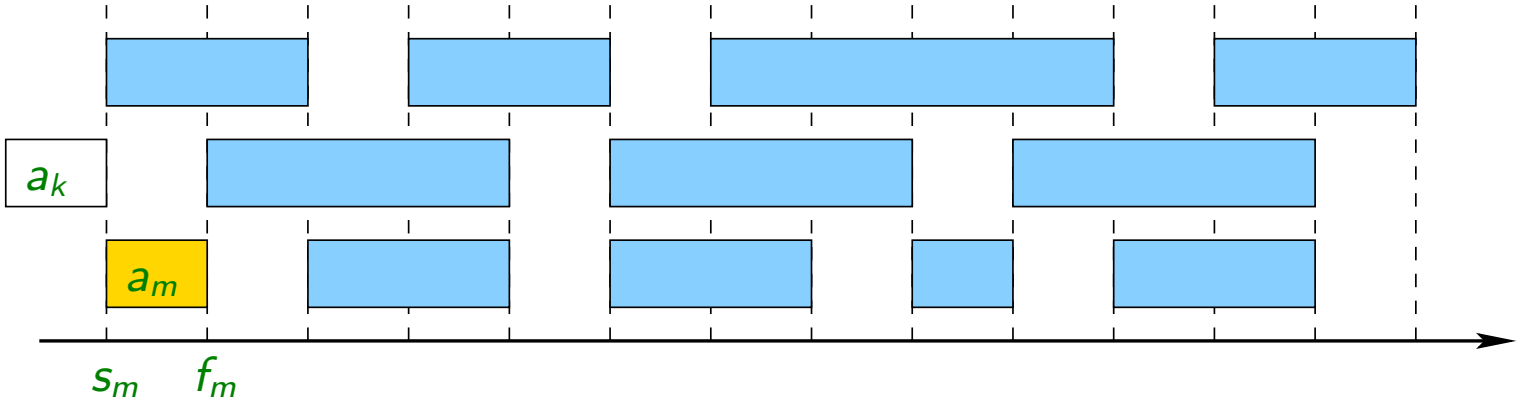
- Seja $S_k = \{a_i \in S : s_i \geq f_k\}$. Ou seja, S_k é o conjunto de atividades de S que começam depois de f_k, o instante em que a_k termina.
- Seja a_m a atividade com menor instante de término de S_k.

Seleção de Atividades – Escolha Gulosa



- Seja $S_k = \{a_i \in S : s_i \geq f_k\}$. Ou seja, S_k é o conjunto de atividades de S que começam depois de f_k, o instante em que a_k termina.
- Seja a_m a atividade com menor instante de término de S_k.
- A subestrutura ótima diz que se a_m pertence a uma solução ótima A de S_k, então $A - \{a_m\}$ é uma solução ótima de S_m.

Seleção de Atividades – Escolha Gulosa



- Seja $S_k = \{a_i \in S : s_i \geq f_k\}$. Ou seja, S_k é o conjunto de atividades de S que começam depois de f_k, o instante em que a_k termina.
- Seja a_m a atividade com menor instante de término de S_k.
- A subestrutura ótima diz que se a_m pertence a uma solução ótima A de S_k, então $A - \{a_m\}$ é uma solução ótima de S_m.
- Nossa intuição nos diz que a_m pertence a alguma solução ótima. Como podemos provar isto?

Seleção de Atividades – Escolha Gulosa

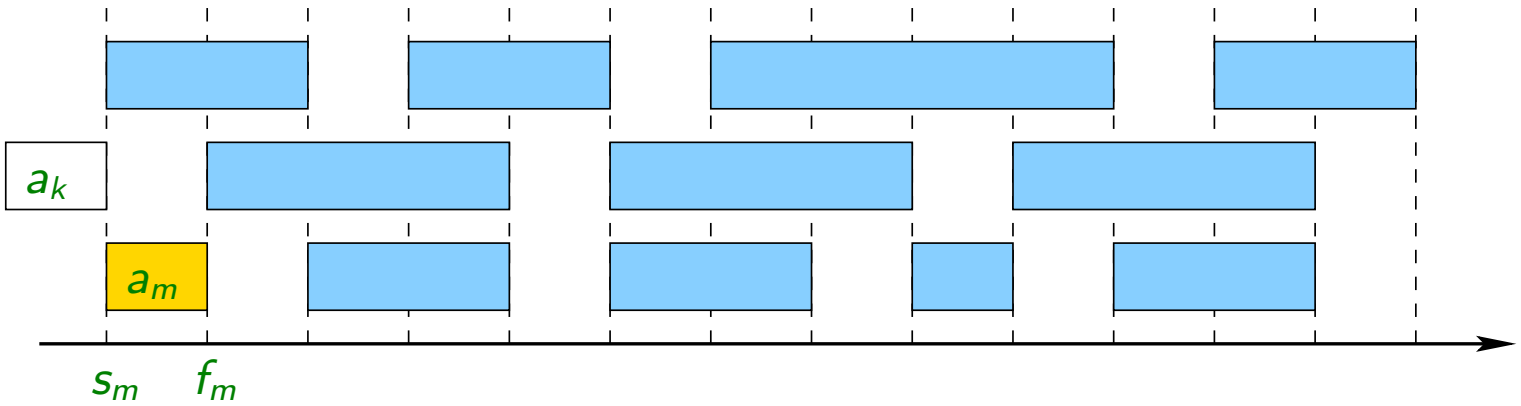
Seleção de Atividades – Escolha Gulosa

Teorema (Escolha Gulosa).

Considere um subproblema não-vazio S_k, e seja a_m uma atividade de S_k com o menor instante de término, i.e:

$$f_m = \min\{f_j : a_j \in S_k\}.$$

Então existe um subconjunto máximo de atividades mutuamente compatíveis de S_k que contém a_m.



Seleção de Atividades – Escolha Gulosa

Mostraremos que existe uma solução ótima de S_k que contém a_m.

Seleção de Atividades – Escolha Gulosa

Mostraremos que existe uma solução ótima de S_k que contém a_m.

Seja A um conjunto de atividades mutuamente compatíveis de tamanho máximo em S_k. Se $a_m \in A$ então nada há a fazer. Suponha então que $a_m \notin A$.

Seleção de Atividades – Escolha Gulosa

Mostraremos que existe uma solução ótima de S_k que contém a_m.

Seja A um conjunto de atividades mutuamente compatíveis de tamanho máximo em S_k. Se $a_m \in A$ então nada há a fazer. Suponha então que $a_m \notin A$.

Seja $a_j \in A$ com menor f_j.

Seleção de Atividades – Escolha Gulosa

Mostraremos que existe uma solução ótima de S_k que contém a_m.

Seja A um conjunto de atividades mutuamente compatíveis de tamanho máximo em S_k. Se $a_m \in A$ então nada há a fazer. Suponha então que $a_m \notin A$.

Seja $a_j \in A$ com menor f_j. Seja $A' = A - \{a_j\} \cup \{a_m\}$.

Seleção de Atividades – Escolha Gulosa

Mostraremos que existe uma solução ótima de S_k que contém a_m.

Seja A um conjunto de atividades mutuamente compatíveis de tamanho máximo em S_k. Se $a_m \in A$ então nada há a fazer. Suponha então que $a_m \notin A$.

Seja $a_j \in A$ com menor f_j. Seja $A' = A - \{a_j\} \cup \{a_m\}$.
Então A' também é um conjunto de atividades mutuamente compatíveis de tamanho máximo. (Por quê?)

Seleção de Atividades – Escolha Gulosa

Mostraremos que existe uma solução ótima de S_k que contém a_m.

Seja A um conjunto de atividades mutuamente compatíveis de tamanho máximo em S_k. Se $a_m \in A$ então nada há a fazer. Suponha então que $a_m \notin A$.

Seja $a_j \in A$ com menor f_j. Seja $A' = A - \{a_j\} \cup \{a_m\}$.
Então A' também é um conjunto de atividades mutuamente compatíveis de tamanho máximo. (Por quê?)

Técnica muito importante!

Modificar uma solução ótima *genérica* para obter uma solução ótima com a escolha gulosa. **Tenho que me lembrar disso!**

Seleção de Atividades – Algoritmo Guloso

Suponha que estamos tentando resolver o problema para S_k.

Seleção de Atividades – Algoritmo Guloso

Suponha que estamos tentando resolver o problema para S_k.

- Pelo Teorema, existe um conjunto máximo de atividades mutuamente compatíveis que contém a_m, o intervalo com menor instante de término de S_k.

Seleção de Atividades – Algoritmo Guloso

Suponha que estamos tentando resolver o problema para S_k.

- Pelo Teorema, existe um conjunto máximo de atividades mutuamente compatíveis que contém a_m, o intervalo com menor instante de término de S_k.
- Pela subestrutura ótima do problema sabemos que uma tal solução ótima é composta por a_m e uma solução ótima para S_m, o conjunto de intervalos compatíveis com a_m.

Seleção de Atividades – Algoritmo Guloso

Suponha que estamos tentando resolver o problema para S_k.

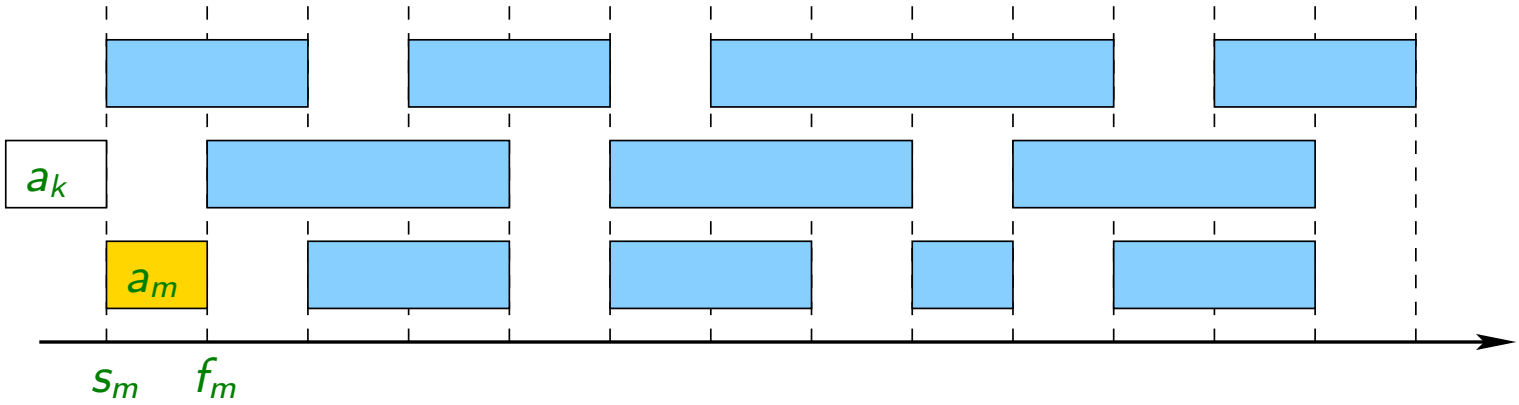
- Pelo Teorema, existe um conjunto máximo de atividades mutuamente compatíveis que contém a_m, o intervalo com menor instante de término de S_k.
- Pela subestrutura ótima do problema sabemos que uma tal solução ótima é composta por a_m e uma solução ótima para S_m, o conjunto de intervalos compatíveis com a_m.
- Assim, basta encontrar uma coleção máxima de atividades mutuamente compatíveis de S_m e juntar a_m a ela para obter uma solução ótima para S_k.

Seleção de Atividades – Algoritmo Guloso

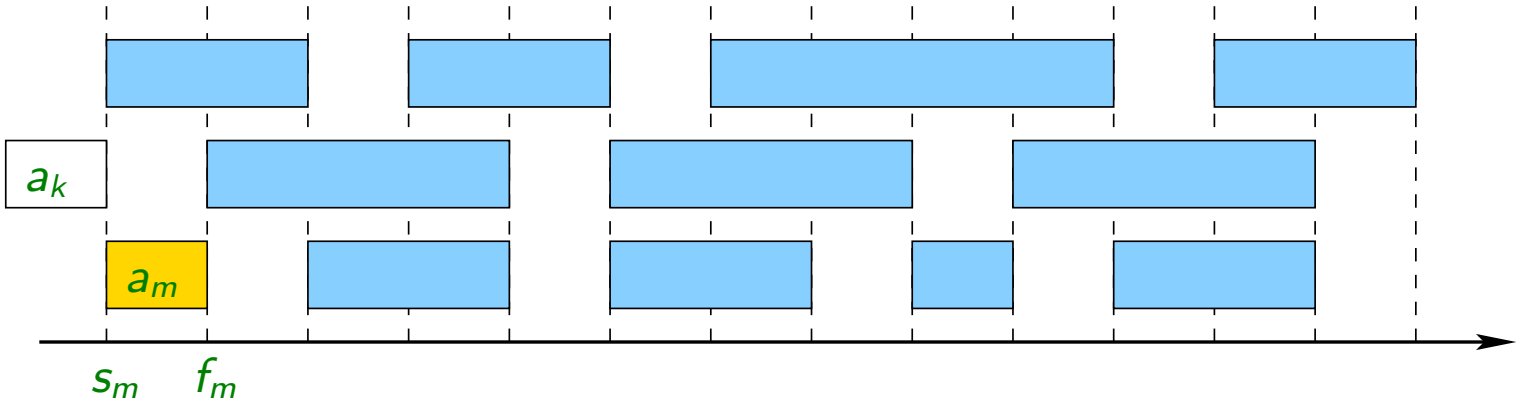
Suponha que estamos tentando resolver o problema para S_k.

- Pelo Teorema, existe um conjunto máximo de atividades mutuamente compatíveis que contém a_m, o intervalo com menor instante de término de S_k.
- Pela subestrutura ótima do problema sabemos que uma tal solução ótima é composta por a_m e uma solução ótima para S_m, o conjunto de intervalos compatíveis com a_m.
- Assim, basta encontrar uma coleção máxima de atividades mutuamente compatíveis de S_m e juntar a_m a ela para obter uma solução ótima para S_k.
- Note que S_m é uma instância (menor) do mesmo tipo de S_k. Logo, podemos usar a mesma escolha gulosa para S_m e repetir o processo (recursivamente ou iterativamente).

Seleção de Atividades - Algoritmo

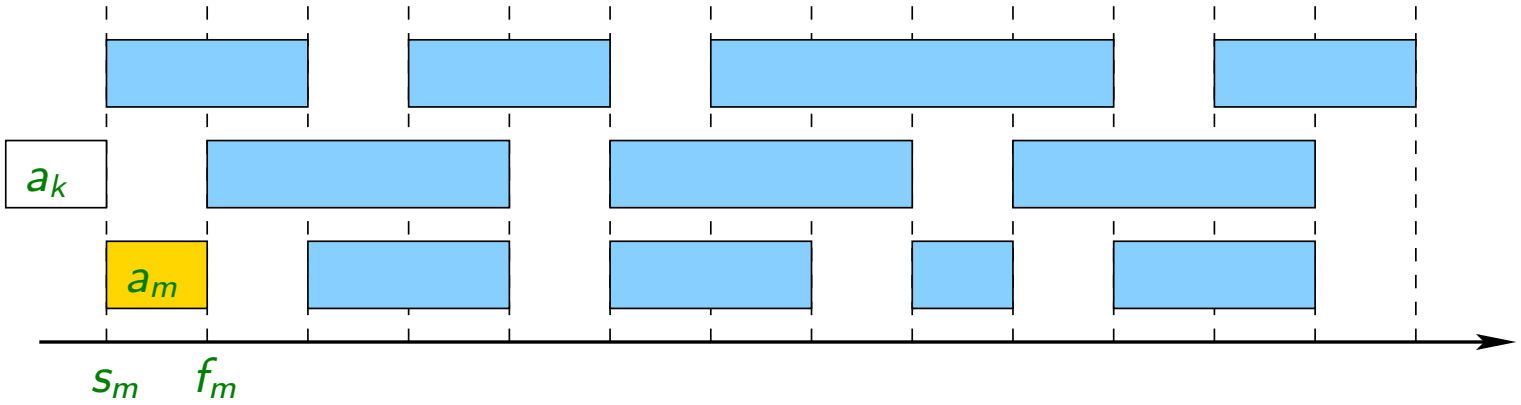


Seleção de Atividades - Algoritmo



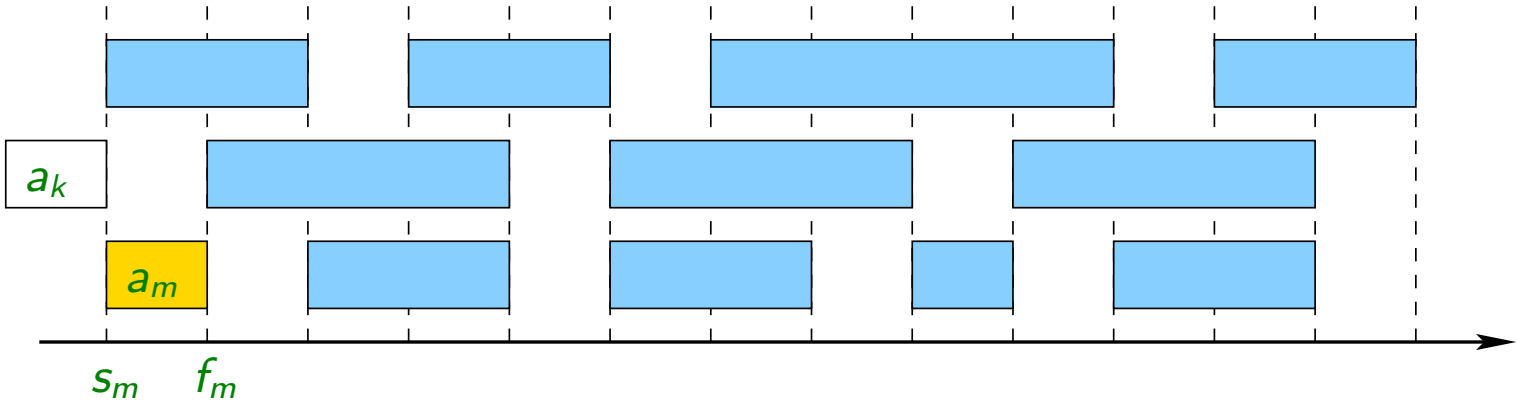
- Suponha que estamos tentando resolver S_k.

Seleção de Atividades - Algoritmo



- Suponha que estamos tentando resolver S_k.
- Determine a atividade a_m com menor instante de término em S_k.

Seleção de Atividades - Algoritmo



- Suponha que estamos tentando resolver S_k.
- Determine a atividade a_m com menor instante de término em S_k.
- Resolva o subproblema S_m e junte a_m à solução obtida na recursão. Devolva este conjunto de atividades.

Seleção de Atividades

SELEÇÃO-ATIVIDADES-REC(s, f, k, n)

1. $m \leftarrow k + 1$;
2. **enquanto** $m < n$ **e** $s_m < f_k$ **faça** $\triangleright$ acha a_m com menor f_m
3. $\quad m \leftarrow m + 1$;
4. **se** $m \leq n$ **então**
5. $\quad$ **devolva** $\{a_m\} \cup$ SELEÇÃO-ATIVIDADES-REC(s, f, m, n)
6. **devolva** $\emptyset$

Seleção de Atividades

SELEÇÃO-ATIVIDADES-REC(s, f, k, n)

1. $m \leftarrow k + 1$;
2. **enquanto** $m < n$ **e** $s_m < f_k$ **faça** $\triangleright$ acha a_m com menor f_m
3. $\quad m \leftarrow m + 1$;
4. **se** $m \leq n$ **então**
5. $\quad$ **devolva** $\{a_m\} \cup$ SELEÇÃO-ATIVIDADES-REC(s, f, m, n)
6. **devolva** $\emptyset$

SELEÇÃO-ATIVIDADES-REC(s, f, k, n) devolve uma solução ótima de S_k.

Seleção de Atividades

SELEÇÃO-ATIVIDADES-REC(s, f, k, n)

1. $m \leftarrow k + 1$;
2. **enquanto** $m < n$ **e** $s_m < f_k$ **faça** ▷ acha a_m com menor f_m
3. $\quad m \leftarrow m + 1$;
4. **se** $m \leq n$ **então**
5. $\quad$ **devolva** $\{a_m\} \cup$ SELEÇÃO-ATIVIDADES-REC(s, f, m, n)
6. **devolva** $\emptyset$

SELEÇÃO-ATIVIDADES-REC(s, f, k, n) devolve uma solução ótima de S_k.

Note que a_k foi a atividade previamente escolhida para fazer parte da solução.

Assim, as linhas 2–3 encontram a atividade a_m com menor f_m que seja compatível com a_k (ou seja, está em S_k).

Seleção de Atividades - Recursivo

Seleção de Atividades - Recursivo

- A chamada inicial é SELEÇÃO-ATIVIDADES-REC$(s, f, 0, n)$.

Seleção de Atividades - Recursivo

- A chamada inicial é SELEÇÃO-ATIVIDADES-REC$(s, f, 0, n)$.
- **Complexidade**: $\Theta(n)$.
  Ao longo de todas as chamadas recursivas,cada atividade é examinada exatamente uma vez no laço da linha 2.

Seleção de Atividades - Recursivo

- A chamada inicial é SELEÇÃO-ATIVIDADES-REC$(s, f, 0, n)$.
- **Complexidade**: $\Theta(n)$.
  Ao longo de todas as chamadas recursivas,cada atividade é examinada exatamente uma vez no laço da linha 2.
- Como SELEÇÃO-ATIVIDADES-REC usa recursão caudal, é trivial escrever uma versão iterativa do mesmo.

Seleção de Atividades - Iterativo

SELEÇÃO-ATIVIDADES-GULOSO(s, f, n)

1. $A \leftarrow \{a_1\}$;
2. $k \leftarrow 1$;
3. **para** $m \leftarrow 2$ **até** n **faça**
4. **se** $s_m \geq f_k$ **então**
5. $A \leftarrow A \cup \{a_m\}$;
6. $k \leftarrow m$;
7. **devolva** A.

Seleção de Atividades - Iterativo

SELEÇÃO-ATIVIDADES-GULOSO(s, f, n)

1. $A \leftarrow \{a_1\}$;
2. $k \leftarrow 1$;
3. **para** $m \leftarrow 2$ **até** n **faça**
4. **se** $s_m \geq f_k$ **então**
5. $A \leftarrow A \cup \{a_m\}$;
6. $k \leftarrow m$;
7. **devolva** A.

Complexidade: $\Theta(n)$.

Seleção de Atividades – Corretude

Seleção de Atividades – Corretude

- Observe que no início da linha 3, a_k é a última atividade que foi colocada em A. Como as atividades estão em ordenadas pelo instante de término, temos que:

$$f_k = \max\{f_j : a_j \in A\},$$

ou seja, f_k é o maior instante de término de uma atividade em A.

Seleção de Atividades – Corretude

- Observe que no início da linha 3, a_k é a última atividade que foi colocada em A. Como as atividades estão em ordenadas pelo instante de término, temos que:

$$f_k = \max\{f_j : a_j \in A\},$$

  ou seja, f_k é o maior instante de término de uma atividade em A.
- Nas linhas 4–6 procura-se a próxima atividade a_m (menor f_m) que seja compatível com a_k e toma-se esta como a nova atividade a_k.

Seleção de Atividades – Corretude

- Observe que no início da linha 3, a_k é a última atividade que foi colocada em A. Como as atividades estão em ordenadas pelo instante de término, temos que:

$$f_k = \max\{f_j : a_j \in A\},$$

  ou seja, f_k é o maior instante de término de uma atividade em A.
- Nas linhas 4–6 procura-se a próxima atividade a_m (menor f_m) que seja compatível com a_k e toma-se esta como a nova atividade a_k.
- Assim, SELEÇÃO-ATIVIDADES-GULOSO faz as mesmas escolhas de SELEÇÃO-ATIVIDADES-REC. Portanto, está correto.

Método de prova de algoritmos gulosos

Método de prova de algoritmos gulosos

1. Mostre que o problema tem subestrutura ótima.

Método de prova de algoritmos gulosos

1. Mostre que o problema tem subestrutura ótima.
2. Mostre que se a foi a primeira escolha do algoritmo, então **existe** alguma solução ótima que contém a.

Segue então por indução e pela subestrutura ótima que o algoritmo sempre faz escolhas corretas. (**Por quê?**)

Método de prova de algoritmos gulosos

1. Mostre que o problema tem subestrutura ótima.
2. Mostre que se a foi a primeira escolha do algoritmo, então **existe** alguma solução ótima que contém a.

Segue então por indução e pela subestrutura ótima que o algoritmo sempre faz escolhas corretas. (**Por quê?**)

Rascunho:

Método de prova de algoritmos gulosos

1. Mostre que o problema tem subestrutura ótima.
2. Mostre que se a foi a primeira escolha do algoritmo, então **existe** alguma solução ótima que contém a.

Segue então por indução e pela subestrutura ótima que o algoritmo sempre faz escolhas corretas. (**Por quê?**)

Rascunho:

Por (2) segue que a_1 pertence a alguma solução ótima A.

Método de prova de algoritmos gulosos

1. Mostre que o problema tem subestrutura ótima.
2. Mostre que se a foi a primeira escolha do algoritmo, então **existe** alguma solução ótima que contém a.

Segue então por indução e pela subestrutura ótima que o algoritmo sempre faz escolhas corretas. (**Por quê?**)

Rascunho:

Por (2) segue que a_1 pertence a alguma solução ótima A.

Por (1) segue que $A' := A - \{a_1\}$ é uma solução ótima de S_1 e portanto, A' pode ser encontrada recursivamente/indutivamente.

Método de prova de algoritmos gulosos

Método de prova de algoritmos gulosos

- Para alguns problemas, a definição de subproblema é um pouco mais sutil; não basta eliminar "itens" que não podem ser usados. Veremos um exemplo disto no problema do Código de Huffman.

Método de prova de algoritmos gulosos

- Para alguns problemas, a definição de subproblema é um pouco mais sutil; não basta eliminar "itens" que não podem ser usados. Veremos um exemplo disto no problema do Código de Huffman.
- Resumo da ópera. Para mostrar que um algoritmo guloso está correto, você deve mostrar que
  1. o problema tem subestrutura ótima, e
  2. a escolha gulosa do algoritmo de fato pertence a alguma solução ótima.

Método de prova de algoritmos gulosos

- Para alguns problemas, a definição de subproblema é um pouco mais sutil; não basta eliminar "itens" que não podem ser usados. Veremos um exemplo disto no problema do Código de Huffman.
- Resumo da ópera. Para mostrar que um algoritmo guloso está correto, você deve mostrar que
  1. o problema tem subestrutura ótima, e
  2. a escolha gulosa do algoritmo de fato pertence a alguma solução ótima.
- Obviamente, esses dois passos dependem do problema.

Exercícios

Exercício. Você trabalha para a IMF (Impossible Mission Force) e acabou de invadir uma fortaleza super-protegida sem ser detectado! Você está diante do computador central da instalação e precisa copiar os arquivos que estão no HD. Você tem à sua disposição um super-pendrive de capacidade W, mas ao contrário dos filmes, ele não é grande o suficiente para copiar todos os arquivos. Suponha que os n arquivos tenham tamanho $w_1, \ldots, w_n$. Você deve tentar copiar o maior número possível de arquivos para seu pendrive e *get the hell out of there*!.

Formalmente, sejam $n+1$ inteiros positivos $w_1, \ldots, w_n$ e W. Projete um algoritmo de complexidade $o(n^2)$ para encontrar o maior subconjunto $I \subseteq \{1, \ldots, n\}$ tal que $\sum_{i \in I} w_i \leq W$ e $|I|$ é máximo. Justifique a complexidade e a corretude do seu algoritmo.

Exercícios

Exercício. Suponha agora que em vez de maximizar o número de arquivos copiados, você quer copiar arquivos de modo a maximizar o espaço ocupado do seu pendrive. Formalmente, queremos encontrar um subconjunto $I \subseteq \{1, \ldots, n\}$ tal que $\sum_{i \in I} w_i \leq W$ e $\sum_{i \in I} w_i$ é máximo.

Isto pode ser visto como o problema da mochila em que o valor de cada item é igual ao seu peso. Proponha alguns algoritmos gulosos para o problema. Para cada um deles encontre uma instância para o qual ele falha!