

Análise de Algoritmos

Slides de Paulo Feofiloff

[com erros do coelho e agora também da cris]

Algoritmos gulosos (*greedy*)

CLRS 16.1–16.3

Algoritmos gulosos

“A *greedy algorithm* starts with a solution to a very small subproblem and augments it successively to a solution for the big problem. The augmentation is done in a “greedy” fashion, that is, paying attention to short-term or local gain, without regard to whether it will lead to a good long-term or global solution. As in real life, greedy algorithms sometimes lead to the best solution, sometimes lead to pretty good solutions, and sometimes lead to lousy solutions. The trick is to determine when to be greedy.”

“One thing you will notice about greedy algorithms is that they are usually easy to design, easy to implement, easy to analyse, and they are very fast, but they are *almost always difficult to prove correct*.”

I. Parberry, *Problems on Algorithms*, Prentice Hall, 1995.

Problema fracionário da mochila

Problema: Dados (w, v, n, W) , encontrar uma **mochila ótima**.

Exemplo: $W = 50, n = 4$

	1	2	3	4
w	40	30	20	10
v	840	600	400	100
x	1	0	0	0
x	1	0	0	1
x	0	1	1	0
x	1	1/3	0	0

valor = 840

valor = 940

valor = 1000

valor = 1040

A propósito ...

O problema fracionário da mochila é um problema de **programação linear (PL)**: encontrar um vetor x que

$$\begin{aligned} & \text{maximize} && x \cdot v \\ & \text{sob as restrições} && x \cdot w \leq W \\ & && x[i] \geq 0 \quad \text{para } i = 1, \dots, n \\ & && x[i] \leq 1 \quad \text{para } i = 1, \dots, n \end{aligned}$$

PL's podem ser resolvidos por

SIMPLEX: no **pior caso** consome tempo **exponencial**
na prática é **muito rápido**

ELIPSÓIDES: consome tempo **polinomial**
na prática é **lento**

PONTOS-INTERIORES: consome tempo **polinomial**
na prática é **rápido**

Subestrutura ótima

Suponha que $x[1..n]$ é **mochila ótima** para o problema (w, v, n, W) .

Se $x[n] = \delta$

então $x[1..n-1]$ é **mochila ótima** para

$$(w, v, n - 1, W - \delta w[n])$$

NOTA. Não há nada de especial acerca do índice n . Uma afirmação semelhante vale para qualquer índice i .

Escolha gulosa

Suponha $w[i] \neq 0$ para todo i .

Se $v[n]/w[n] \geq v[i]/w[i]$ para todo i

então **EXISTE** uma mochila ótima $x[1..n]$ tal que

$$x[n] = \min \left\{ 1, \frac{W}{w[n]} \right\}$$

Algoritmo guloso

Esta **propriedade da escolha gulosa** sugere um algoritmo que atribui os valores de $x[1..n]$ supondo que os dados estejam em ordem decrescente de “valor específico” :

$$\frac{v[1]}{w[1]} \leq \frac{v[2]}{w[2]} \leq \dots \leq \frac{v[n]}{w[n]}$$

É nessa ordem “**mágica**” que está o **segredo do funcionamento** do algoritmo.

Algoritmo guloso

Devolve uma **mochila ótima** para (w, v, n, W) .

MOCHILA-FRACIONÁRIA (w, v, n, W)

```
0  ordene  $w$  e  $v$  de tal forma que  
    $v[1]/w[1] \leq v[2]/w[2] \leq \dots \leq v[n]/w[n]$   
1  para  $i \leftarrow n$  decrecendo até 1 faça  
2      se  $w[i] \leq W$   
3          então  $x[i] \leftarrow 1$   
4               $W \leftarrow W - w[i]$   
5          senão  $x[i] \leftarrow W/w[i]$   
6               $W \leftarrow 0$   
7  devolva  $x$ 
```

Consumo de tempo da linha 0 é $\Theta(n \lg n)$.

Consumo de tempo das linhas 1–7 é $\Theta(n)$.

Invariante

Seja W_0 o valor original de W .

No início de cada execução da linha 1 vale que

(i0) $x' = x[i+1..n]$ é **mochila ótima** para

$$(w', v', n', W_0)$$

onde

$$w' = w[i+1..n]$$

$$v' = v[i+1..n]$$

$$n' = n - i$$

Na última iteração $i = 0$ e portanto $x[1..n]$ é **mochila ótima** para (w, v, n, W_0) .

Conclusão

O consumo de tempo do algoritmo
MOCHILA-FRACIONÁRIA é $\Theta(n \lg n)$.

Escolha gulosa

Precisamos mostrar que se $x[1..n]$ é uma **mochila ótima**, então podemos supor que

$$x[n] = \alpha := \min \left\{ 1, \frac{W}{w[n]} \right\}$$

Depois de mostrar isto, indução faz o resto do serviço.

Técnica: transformar uma **solução ótima** em uma **solução ótima 'gulosa'**.

Esta transformação é semelhante ao processo de pivotação feito pelo algoritmo **SIMPLEX** para programação linear.

Algoritmos gulosos

Algoritmo guloso

- procura ótimo local e acaba obtendo ótimo global
- costuma ser
- muito simples e intuitivo
 - muito eficiente
 - difícil provar que está correto

Problema precisa ter

- subestrutura ótima (como na programação dinâmica)
- propriedade da escolha gulosa (*greedy-choice property*)

Exercício: O problema da mochila booleana pode ser resolvido por um algoritmo guloso?

Problema dos intervalos disjuntos

Problema: Dados intervalos $[s[1], f[1]), \dots, [s[n], f[n])$, encontrar uma **coleção máxima de intervalos** disjuntos dois a dois.

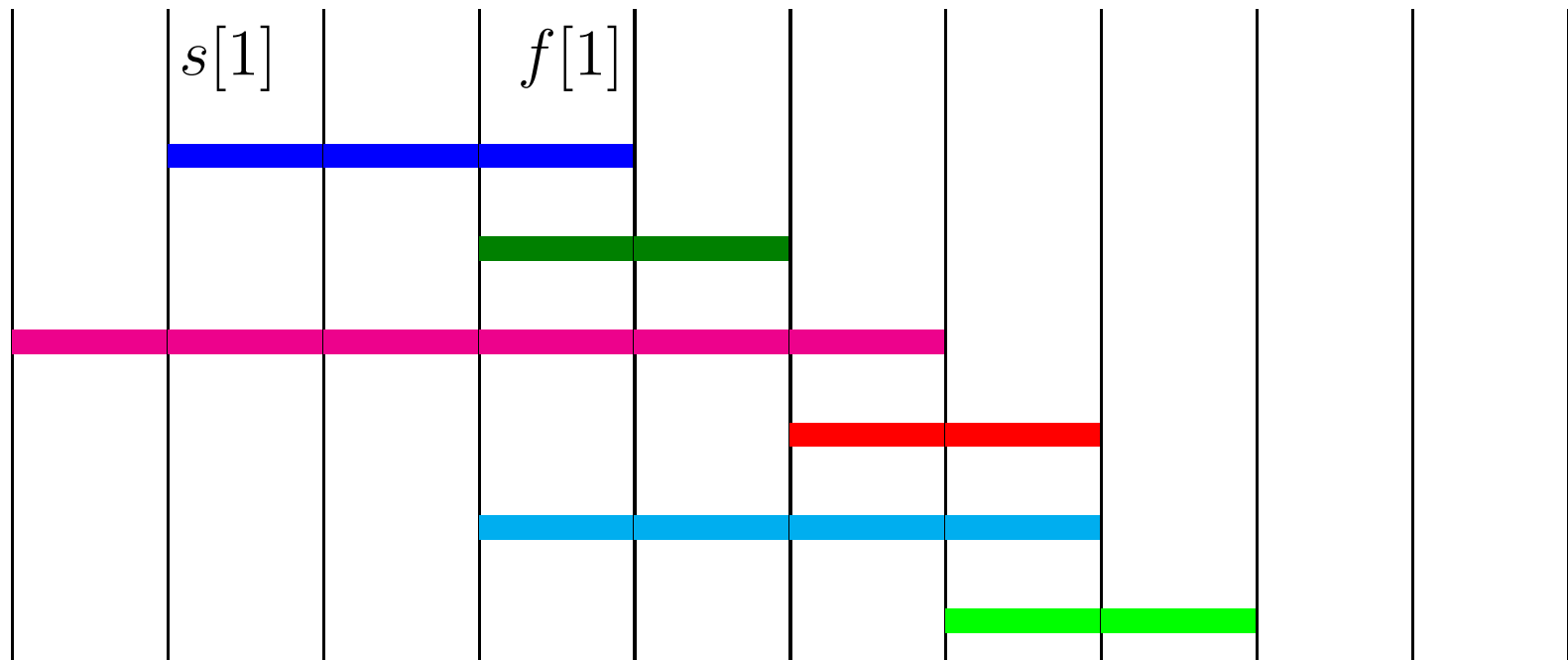
Solução é um subconjunto A de $\{1, \dots, n\}$.

Problema dos intervalos disjuntos

Problema: Dados intervalos $[s[1], f[1]), \dots, [s[n], f[n])$, encontrar uma **coleção máxima de intervalos** disjuntos dois a dois.

Solução é um subconjunto A de $\{1, \dots, n\}$.

Exemplo:

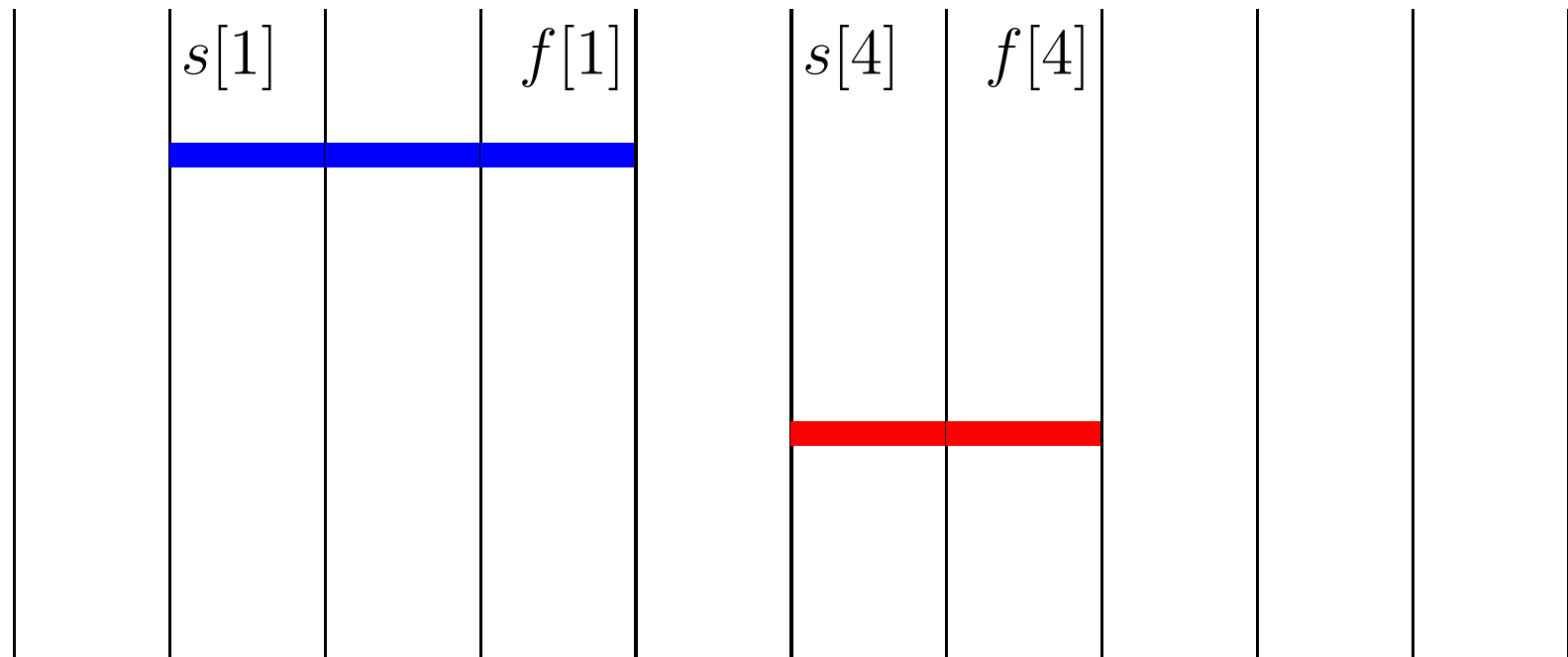


Problema dos intervalos disjuntos

Problema: Dados intervalos $[s[1], f[1]), \dots, [s[n], f[n])$, encontrar uma **coleção máxima de intervalos** disjuntos dois a dois.

Solução é um subconjunto A de $\{1, \dots, n\}$.

Solução:

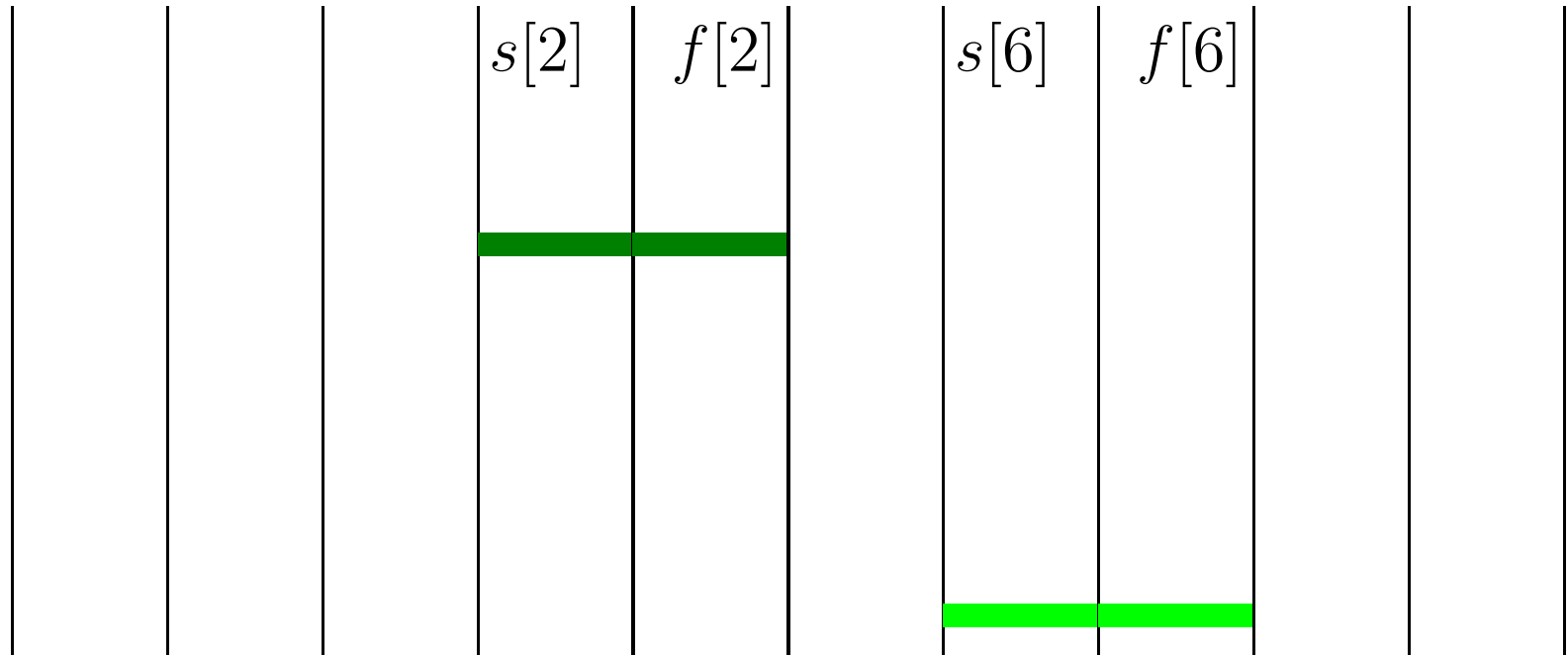


Problema dos intervalos disjuntos

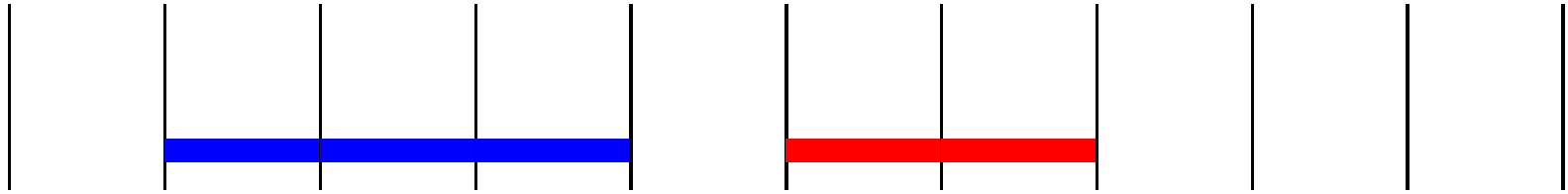
Problema: Dados intervalos $[s[1], f[1]), \dots, [s[n], f[n])$, encontrar uma **coleção máxima de intervalos** disjuntos dois a dois.

Solução é um subconjunto A de $\{1, \dots, n\}$.

Solução:



Motivação



Se cada intervalo é uma “**atividade**”, queremos coleção disjunta máxima de atividades compatíveis (*i* e *j* são compatíveis se $f[i] \leq s[j]$)

Nome no CLRS: **Activity Selection Problem**

Subestrutura ótima

Intervalos $S := \{1, \dots, n\}$

Suponha que A é **coleção máxima** de intervalos de S disjuntos dois a dois.

Se $i \in A$

então $A - \{i\}$ é **coleção máxima** de intervalos disjuntos de $S - \{k : [s[k], f[k]) \cap [s[i], f[i]) \neq \emptyset\}$.

senão A é **coleção máxima** de intervalos disjuntos de $S - \{i\}$.

Demonstre a propriedade.

Subestrutura ótima II

Intervalos $S := \{1, \dots, n\}$

Suponha que A é **coleção máxima** de intervalos de S disjuntos dois a dois.

Se $i \in A$ é tal que $f[i]$ é **mínimo**

então $A - \{i\}$ é **coleção máxima** de intervalos disjuntos de $\{k : s[k] \geq f[i]\}$.

$\{k : s[k] \geq f[i]\} =$ todos intervalos “à direita” de “ i ”.

Demonstre a propriedade.

Algoritmo de programação dinâmica

Suponha $f[1] \leq f[2] \leq \dots \leq f[n]$

$t[i]$ = tamanho de uma subcoleção
disjunta máxima de $\{i, \dots, n\}$

Exercício:

- Ache uma recorrência para $t[i]$;
- Escreva um algoritmo de PD para calcular $t[1]$.