Paradigmas de Projeto de Algoritmos

Algoritmos gulosos

Professora:

Fátima L. S. Nunes

Algoritmos gulosos

- O que é guloso?

Algoritmos gulosos

- O que é guloso?

adj. e s.m. Que ou quem come muito; comilão; glutão

Algoritmos gulosos

- Tipicamente usados para resolver problemas de otimização.
- Exemplo: o problema do GPS
  - queremos encontrar o melhor caminho entre dois locais

Algoritmos gulosos

- Tipicamente usados para resolver problemas de otimização.
- Exemplo: o problema do GPS
  - queremos encontrar o melhor caminho entre dois locais:
    - mais curto;
    - mais barato (menos pedágio);
    - mais rápido;
    - mais bonito.

Algoritmos gulosos

- Tipicamente usados para resolver problemas de otimização.
- Exemplo: o problema do GPS
  - queremos encontrar o melhor caminho entre dois locais:
    - mais curto;
    - mais barato (menos pedágio);
    - mais rápido;
    - mais bonito.

Como podemos representar as características dos locais com o que sabemos até agora (2º período do curso ?)

Algoritmos gulosos

- Tipicamente usados para resolver problemas de otimização.
- Exemplo: considerando que cada célula da matriz é a distância entre duas cidades vizinhas, qual é o caminho mais curto entre as cidades A e B?

A	4	11	1	2	9	8
9	3	20	14	5	7	7
10	2	5	6	6	6	6
15	10	12	3	8	5	3
2	6	22	2	9	5	B
5	2	11	9	3	7	8

Algoritmos gulosos

- Tipicamente usados para resolver problemas de otimização.
- Exemplo: considerando que cada célula da matriz é a distância entre duas cidades vizinhas, qual é o caminho mais curto entre as cidades A e B?

A	4	11	1	2	9	8
9	3	20	14	5	7	7
10	2	5	6	6	6	6
15	10	12	3	8	5	3
2	6	22	2	9	5	B
5	2	11	9	3	7	8

Algoritmos gulosos

- Tipicamente usados para resolver problemas de otimização.
- Exemplo: considerando que cada célula da matriz é a distância entre duas cidades vizinhas, qual é o caminho mais curto entre as cidades A e B?

A	4	11	1	2	9	8
9	3	20	14	5	7	7
10	2	5	6	6	6	6
15	10	12	3	8	5	3
2	6	22	2	9	5	B
5	2	11	9	3	7	8

Algoritmos gulosos

- Tipicamente usados para resolver problemas de otimização.
- Exemplo: considerando que cada célula da matriz é a distância entre duas cidades vizinhas, qual é o caminho mais curto entre as cidades A e B?

A	4	11	1	2	9	8
9	3	20	14	5	7	7
10	2	5	6	6	6	6
15	10	12	3	8	5	3
2	6	22	2	9	5	B
5	2	11	9	3	7	8

Algoritmos gulosos

- Tipicamente usados para resolver problemas de otimização.
- Exemplo: considerando que cada célula da matriz é a distância entre duas cidades vizinhas, qual é o caminho mais curto entre as cidades A e B?

A	4	11	1	2	9	8
9	3	20	14	5	7	7
10	2	5	6	6	6	6
15	10	12	3	8	5	3
2	6	22	2	9	5	B
5	2	11	9	3	7	8

Algoritmos gulosos

- Tipicamente usados para resolver problemas de otimização.
- Exemplo: considerando que cada célula da matriz é a distância entre duas cidades vizinhas, qual é o caminho mais curto entre as cidades A e B?

A	4	11	1	2	9	8
9	3	20	14	5	7	7
10	2	5	6	6	6	6
15	10	12	3	8	5	3
2	6	22	2	9	5	B
5	2	11	9	3	7	8

Algoritmos gulosos

- Tipicamente usados para resolver problemas de otimização.
- Exemplo: considerando que cada célula da matriz é a distância entre duas cidades vizinhas, qual é o caminho mais curto entre as cidades A e B?

A	4	11	1	2	9	8
9	3	20	14	5	7	7
10	2	5	6	6	6	6
15	10	12	3	8	5	3
2	6	22	2	9	5	B
5	2	11	9	3	7	8

Algoritmos gulosos

- Tipicamente usados para resolver problemas de otimização.
- Exemplo: considerando que cada célula da matriz é a distância entre duas cidades vizinhas, qual é o caminho mais curto entre as cidades A e B?

A	4	11	1	2	9	8
9	3	20	14	5	7	7
10	2	5	6	6	6	6
15	10	12	3	8	5	3
2	6	22	2	9	5	B
5	2	11	9	3	7	8

Algoritmos gulosos

- Tipicamente usados para resolver problemas de otimização.
- Exemplo:
  - encontrar o menor caminho entre duas cidades A e B;
  - digamos que pode-se escolher qualquer direção;
  - cada célula contém a distância da cidade atual até a próxima cidade.

A	4	11	1	2	9	8
9	3	20	14	5	7	7
10	2	5	6	6	6	6
15	10	12	3	8	5	3
2	6	22	2	9	5	B
5	2	11	9	3	7	8

Algoritmos gulosos

- Tipicamente usados para resolver problemas de otimização.
- Em cada passo:
  - escolhe a decisão ótima em cada passo, na esperança de obter solução ótima global *(mas nem sempre consegue!)*;
  - nunca reconsidera a decisão tomada em um momento anterior;

A	4	11	1	2	9	8
9	3	20	14	5	7	7
10	2	5	6	6	6	6
15	10	12	3	8	5	3
2	6	22	2	9	5	B
5	2	11	9	3	7	8

Algoritmos gulosos

- Tipicamente usados para resolver problemas de otimização.
- Em cada passo:
  - escolhe a decisão ótima em cada passo, na esperança de obter solução ótima global *(mas nem sempre consegue!)*;
  - nunca reconsidera a decisão tomada em um momento anterior;

- uma vez que o candidato é adicionado à solução, permanecerá na solução para sempre;
- uma vez que o candidato é rejeitado, nunca mais será considerado.

A	4	11	1	2	9	8
9	3	20	14	5	7	7
10	2	5	6	6	6	6
15	10	12	3	8	5	3
2	6	22	2	9	5	B
5	2	11	9	3	7	8

Algoritmos gulosos

- Algoritmo guloso é eficiente para uma grande variedade de problemas.
- Vamos analisar dois problemas:
  - locação de atividades em uma sala;
  - escolher produtos que caibam em uma mochila (famoso ‘problema da mochila’).

Algoritmos gulosos

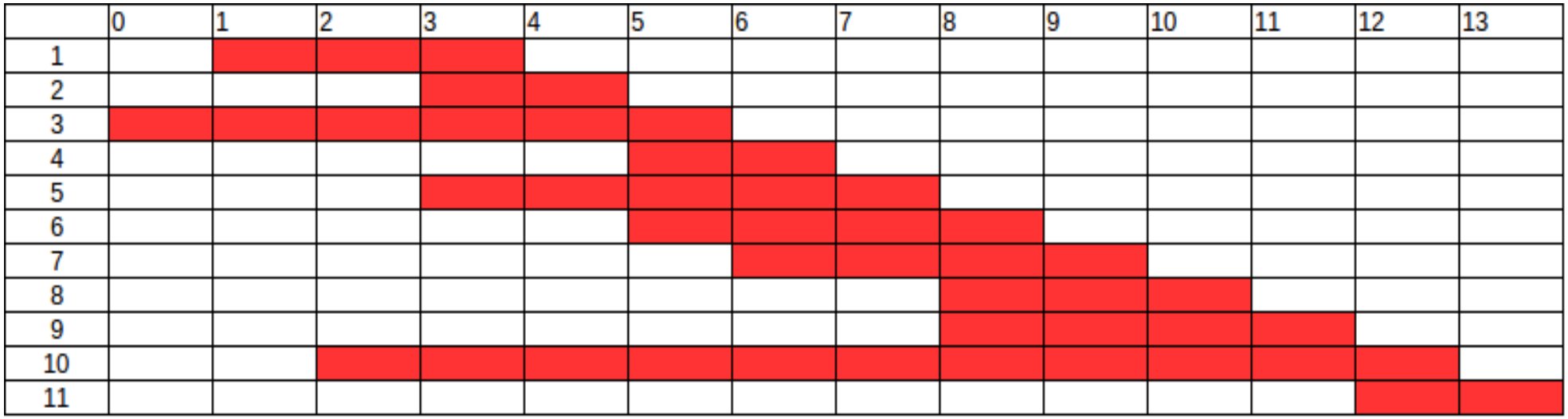
- Exemplo 1: locação de atividades em uma sala
  - existem diversas atividades (por exemplo aulas) que querem usar uma mesma sala;
  - cada atividade tem um horário de início e um horário de fim;
  - só existe uma sala disponível;
  - duas aulas não podem ser ministradas na mesma sala ao mesmo tempo.

Algoritmos gulosos

- Exemplo 1: locação de atividades em uma sala

  - 11 atividades a serem distribuídas em 14 unidades de tempo;
  - queremos selecionar um conjunto máximo de atividades que não têm sobreposição de tempo.

  - Sugestões ???

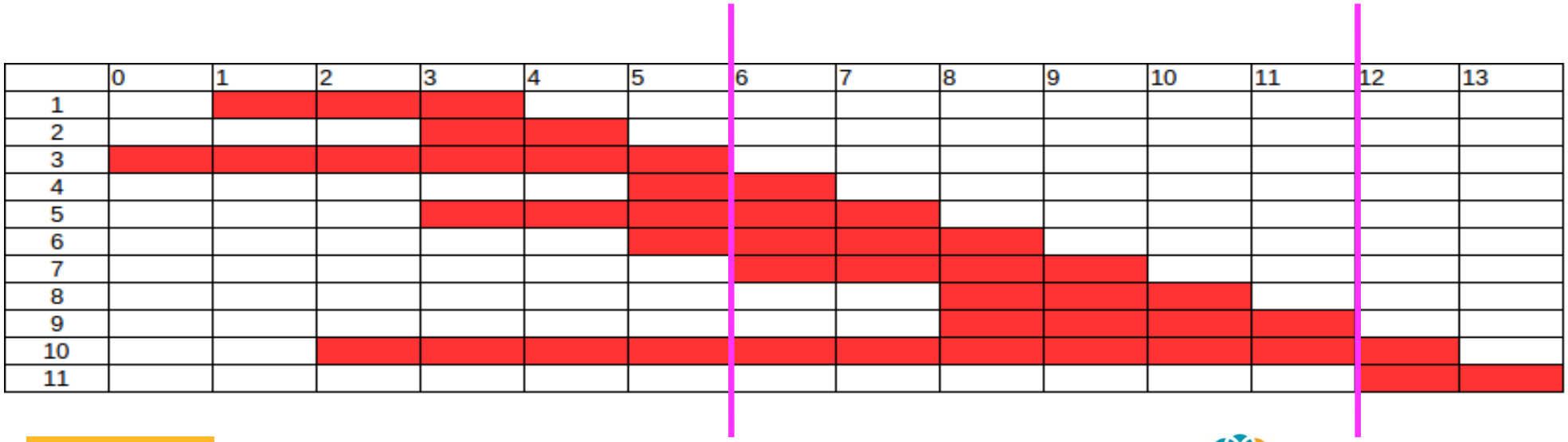


Alocação de atividades

- Opa, mas já resolvemos esse problema!
- Vamos rever a solução ótima que adotamos para nossa programação dinâmica
- A cada intervalo de atividades, temos algumas opções para subdividir o nosso intervalo i...j

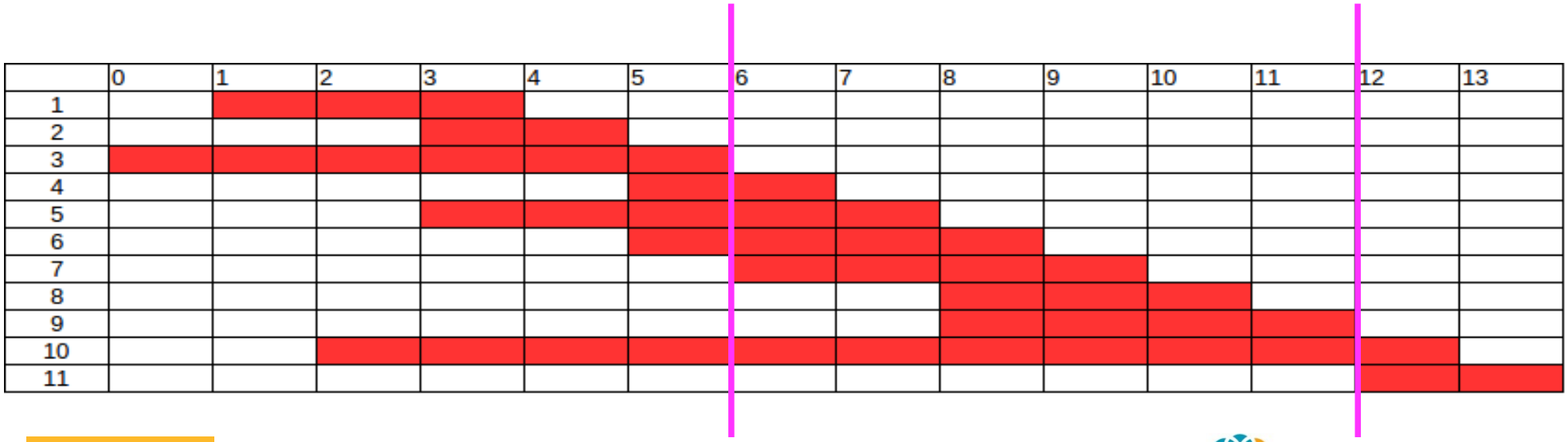
Alocação de atividades

- Vejamos por exemplo o intervalo entre as atividades 3 e 11
- Quais os valores possíveis de k?



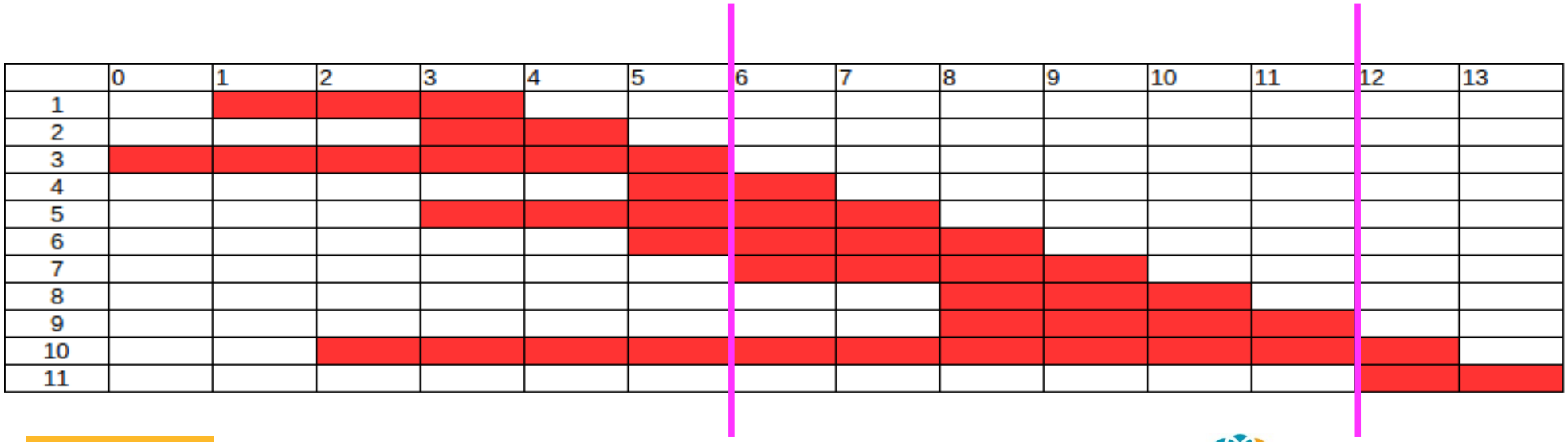
Alocação de atividades

- Vejamos por exemplo o intervalo entre as atividades 3 e 11, ou seja $S_{3\,11}$
- Quais os valores possíveis de k?
- 7 8 9



Alocação de atividades

- Vejamos por exemplo o intervalo entre as atividades 3 e 11, ou seja $S_{3\,11}$
- Quais os valores possíveis de k?
- 7 8 9 – o que acontece se eu escolher a 7?



Alocação de atividades

- Se escolhemos o menor valor de k, então sobra-nos apenas um subproblema a resolver
- Teríamos que resolver S_{ik} e S_{kj}
- O problema S_{ik} é vazio pois se não fosse, o k escolhido não seria o menor possível

Alocação de atividades

- Se escolhemos o menor valor de k, então sobra-nos apenas um subproblema a resolver
- Teríamos que resolver S_{ik} e S_{kj}
- O problema S_{ik} é vazio pois se não fosse, o k escolhido não seria o menor possível
- Posso sempre pegar o menor k????

Alocação de atividades

- Se escolhemos o menor valor de k, então sobra-nos apenas um subproblema a resolver
- Teríamos que resolver S_{ik} e S_{kj}
- O problema S_{ik} é vazio pois se não fosse, o k escolhido
- Posso sem

Somente se ele for o que produzir o maior ganho!!!!!!!!!!

Alocação de atividades

- Mas (não por acaso) isso sempre acontece

0 4
    1(1) 2(1)
0 6
    1(1) 2(1)
5 11
    8(1) 9(1)
0 7
    1(1) 2(1) 3(1)
4 11
    8(1) 9(1)
5 12
    8(2) 9(2) 11(2)
0 8
    1(2) 2(2) 3(1) 4(2) 5(1)
3 11
    7(1) 8(1) 9(1)
4 12
    8(2) 9(2) 11(2)

0 9
    1(2) 2(2) 3(1) 4(2) 5(1)
2 11
    4(2) 6(1) 7(1) 8(2) 9(2)
3 12
    7(2) 8(2) 9(2) 11(2)
1 11
    4(2) 6(1) 7(1) 8(2) 9(2)
2 12
    4(3) 6(2) 7(2) 8(3) 9(3) 11(3)
0 11
    1(3) 2(3) 3(2) 4(3) 5(2) 6(2) 7(2) 8(3) 9(3)
1 12
    4(3) 6(2) 7(2) 8(3) 9(3) 11(3)
0 12
    1(4) 2(4) 3(3) 4(4) 5(3) 6(3) 7(3) 8(4) 9(4) 10(1) 11(4)

Resumindo

- Podemos sempre escolher o menor valor de k para subdividir nosso problema
- Com isso, alcançamos sempre o maior ganho possível
- E podemos ignorar o subproblema que fica “abaixo” do valor de k escolhido

Voltando ao algoritmo guloso

A solução para a PD:

0 12

1(4) 2(4) 3(3) 4(4) 5(3) 6(3) 7(3) 8(4) 9(4) 10(1) 11(4)

Voltando ao algoritmo guloso

A solução para a PD:

0 12

1(4) 2(4) 3(3) 4(4) 5(3) 6(3) 7(3) 8(4) 9(4) 10(1) 11(4)

1 12

4(3) 6(2) 7(2) 8(3) 9(3) 11(3)

Voltando ao algoritmo guloso

A solução para a PD:

0 12

1(4) 2(4) 3(3) 4(4) 5(3) 6(3) 7(3) 8(4) 9(4) 10(1) 11(4)

1 12

4(3) 6(2) 7(2) 8(3) 9(3) 11(3)

4 12

8(2) 9(2) 11(2)

Voltando ao algoritmo guloso

A solução para a PD:

0 12

1(4) 2(4) 3(3) 4(4) 5(3) 6(3) 7(3) 8(4) 9(4) 10(1) 11(4)

1 12

4(3) 6(2) 7(2) 8(3) 9(3) 11(3)

4 12

8(2) 9(2) 11(2)

8 12

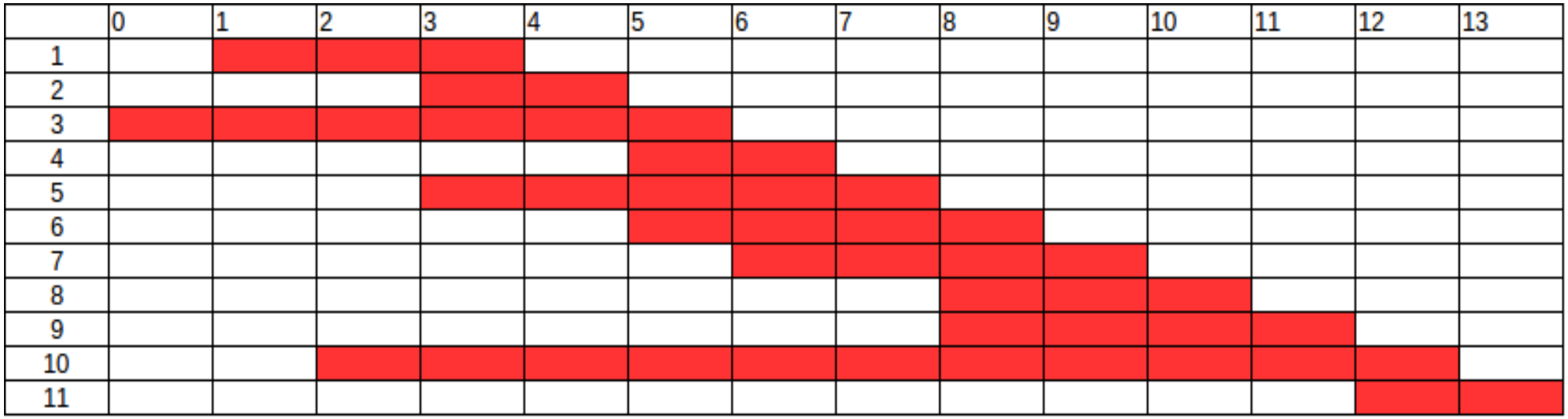
11(1)

Voltando ao algoritmo guloso

- Então, o nosso algoritmo pode ser simplifica a um algoritmo guloso, cuja meta local é sempre pegar a próxima atividade com menor tempo de termino

Voltando ao algoritmo guloso

- Então, o nosso algoritmo pode ser simplificado a um algoritmo guloso, cuja meta local é sempre pegar a próxima atividade com menor tempo de termino



Algoritmos gulosos

- Exemplo 1: locação de atividades em uma sala

  - Algoritmo para o problema de locação de atividades

    - recebe a lista de atividades ordenadas pelo horário de término;
    - a cada iteração verifica se a atividade atual pode ser incluída na lista de atividades;
    - atividade atual é a que termina primeiro, dentre as atividades restantes.

Algoritmos gulosos

- Exemplo 1: locação de atividades em uma sala

```
aloca_guloso(s,f)
    n = tamanho de s
    A = {1}   // adicion aatividade 1
    corrente = 1
    para m de 2 até n
          se s[m] >= f[corrente] // achou uma atividade compativel
                A = A U {m}
                corrente = m
    retorna A
```

Algoritmos gulosos

- Exemplo 1: locação de atividades em uma sala

...

Implemente em C a solução que acabamos de ver.

Algoritmos gulosos

- Exemplo 2: problema da mochila

- há uma mochila que admite um peso máximo
- há um conjunto de objetos, cada um com um valor e um peso;
- devemos selecionar o conjunto de objetos que caibam dentro da mochila de forma a maximizar o valor total dentro dela.

Algoritmos gulosos

- Exemplo 2: problema da mochila

- Dois subproblemas distintos:
  - Objetos podem ser particionados (e o valor será proporcional à fração do objeto), ou seja, você pode colocar um pedaço do objeto dentro da mochila;
    - Ex: ouro em pó
  - Os objetos não podem ser particionados (ou estarão dentro da mochila ou fora). Problema conhecido como “Problema da Mochila Binária ou 0-1”
    - Ex: ouro em barras

Algoritmos gulosos

- Exemplo 2: problema da mochila

- Problema da mochila fracionada:
  - Qual seria a melhor ordenação da entrada?

Algoritmos gulosos

- Exemplo 2: problema da mochila

- Problema da mochila fracionada:
  - Qual seria a melhor ordenação da entrada?
    - ordenar pelo valor/peso
  - A solução gulosa será ótima?
    - Sim (é demonstrável)
  - Algoritmo:

Algoritmos gulosos

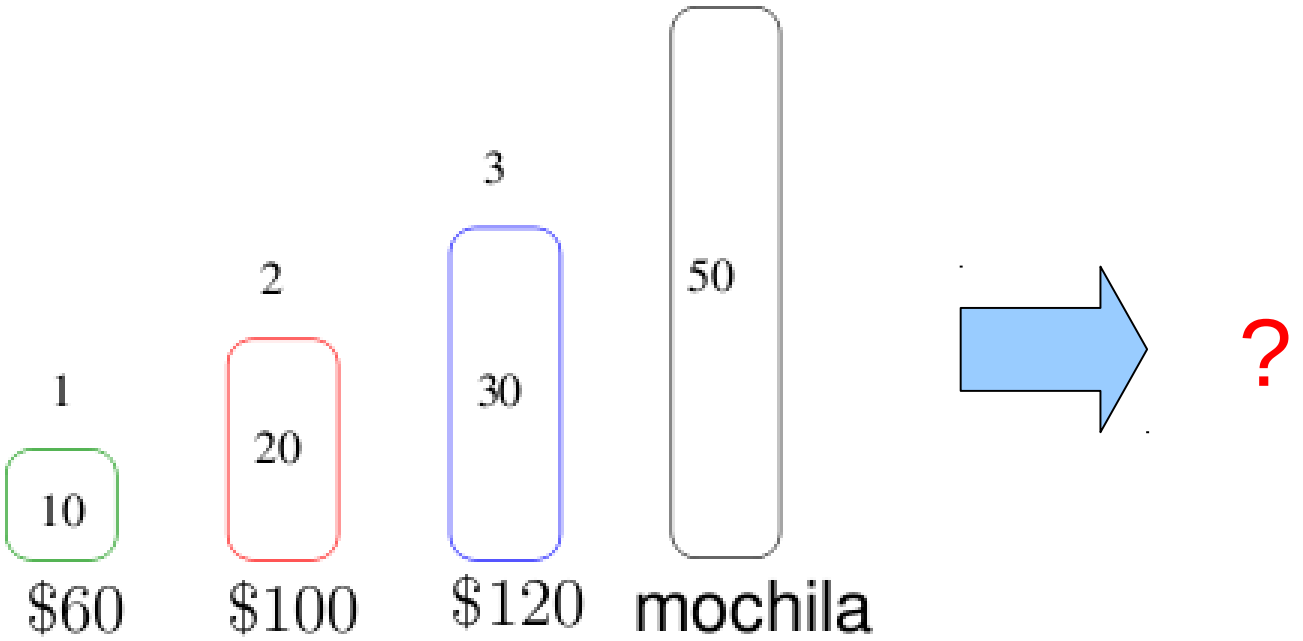
- Exemplo 2: problema da mochila

- Problema da mochila fracionada:
  - Qual seria a melhor ordenação da entrada?
    - ordenar pelo valor/peso
  - A solução gulosa será ótima?
    - Sim (é demonstrável)
  - Algoritmo:
    - ordenar os itens por valor/peso decrescentemente;
    - colocar na mochila o máximo do item *i* que estiver disponível e for possível;
    - passar para o próximo item.

Algoritmos gulosos

- Exemplo 2: problema da mochila

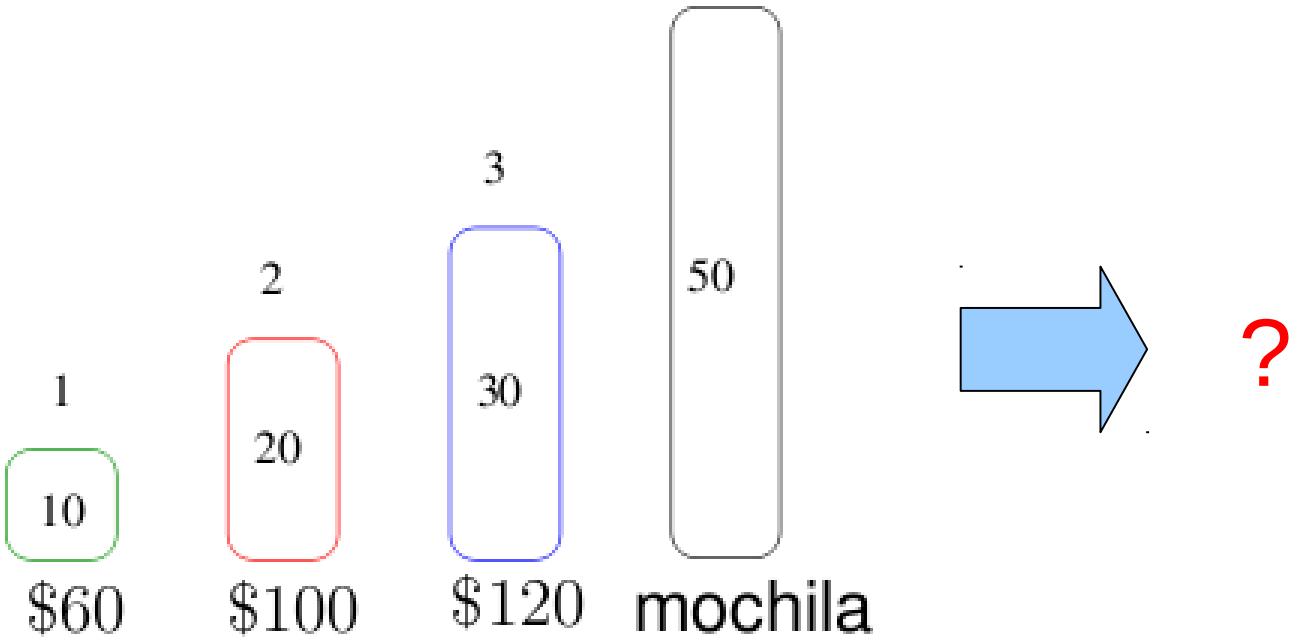
- Problema da mochila fracionada:



Algoritmos gulosos

- Exemplo 2: problema da mochila

- Problema da mochila fracionada:

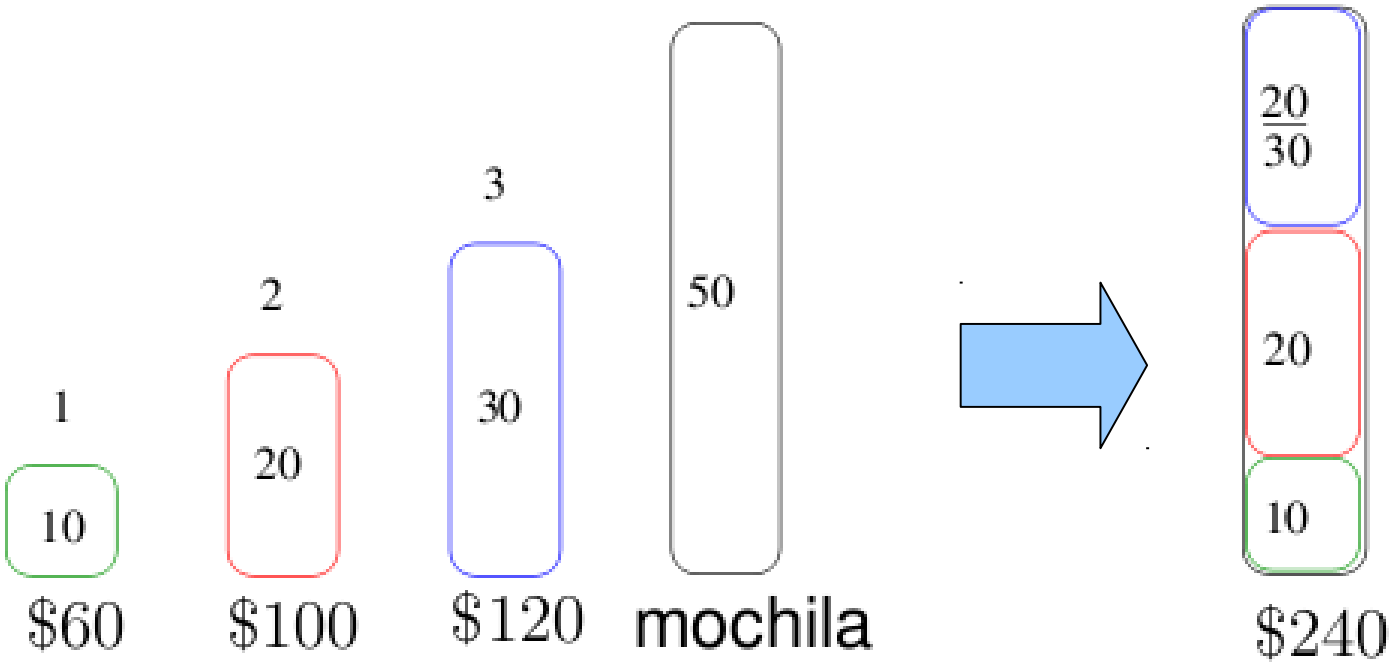


valor/peso = 6 5 4

Algoritmos gulosos

- Exemplo 2: problema da mochila

- Problema da mochila fracionada:



valor/peso = 6 5 4

Algoritmos gulosos

- Exemplo 2: problema da mochila

- Problema da mochila fracionada:

```
// W = capacidade máxima da mochila
load = 0 // carga na mochila
i = 1
while (load < W) and (i <= n) do
{
   if (wi <= (W – load)
      Pegue todo o item i
   else
      Pegue (W – load)/wi do item i
   Adicione a load o peso que foi pego
   i++
}
```

Algoritmos gulosos

- Exemplo 2: problema da mochila

- Problema da mochila binária:
  - Qual seria a melhor ordenação da entrada?

Algoritmos gulosos

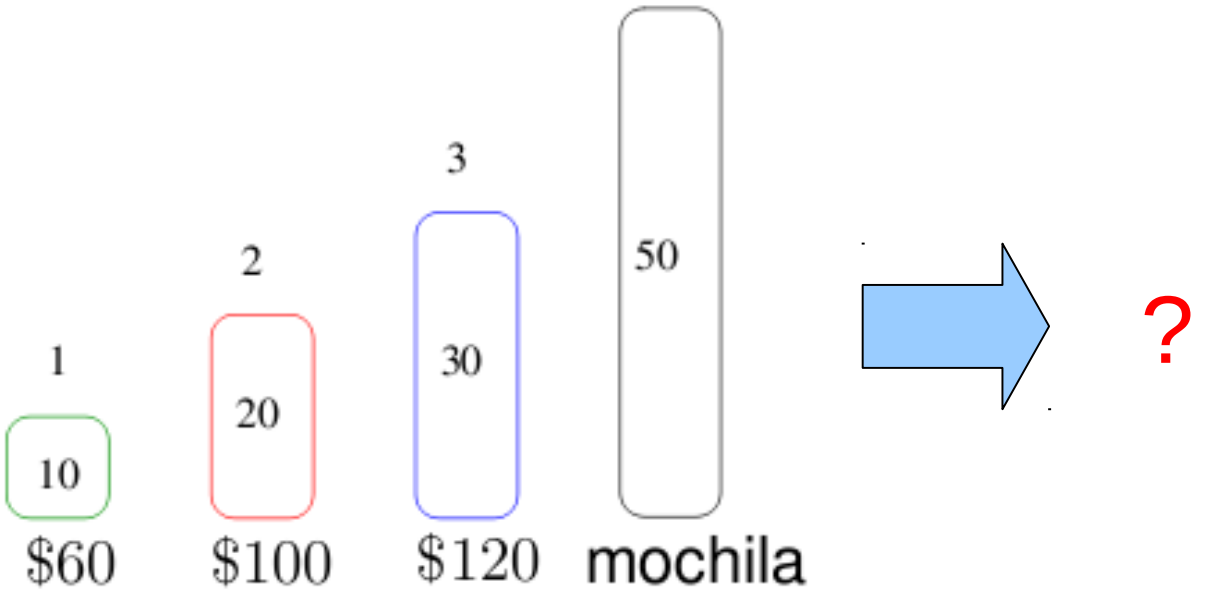
- Exemplo 2: problema da mochila

- Problema da mochila binária:
  - Qual seria a melhor ordenação da entrada?
    - ordenar pelo valor/peso
  - Algoritmo:
    - ordenar os itens por valor/peso decrescentemente;
    - colocar na mochila o item *i* se for possível;
    - passar para o próximo item.
  - A solução gulosa será ótima?

Algoritmos gulosos

- Exemplo 2: problema da mochila

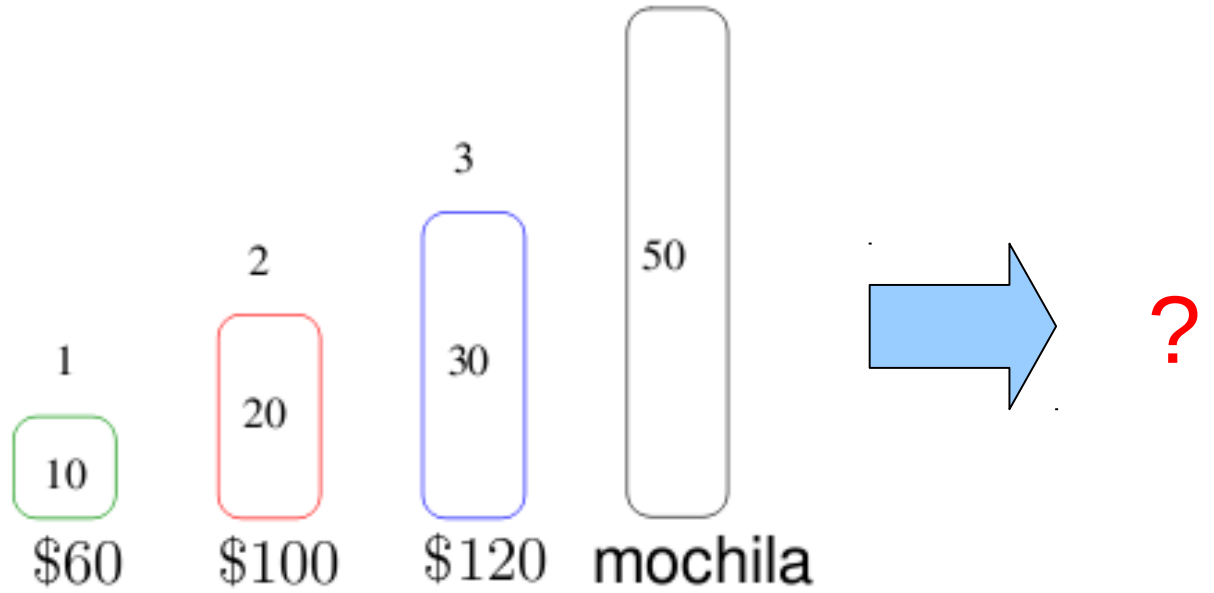
- Problema da mochila binária:



Algoritmos gulosos

- Exemplo 2: problema da mochila

- Problema da mochila binária:



valor/peso = 6 5 4

Algoritmos gulosos

- Exemplo 2: problema da mochila

- Problema da mochila binária:

Algoritmos gulosos

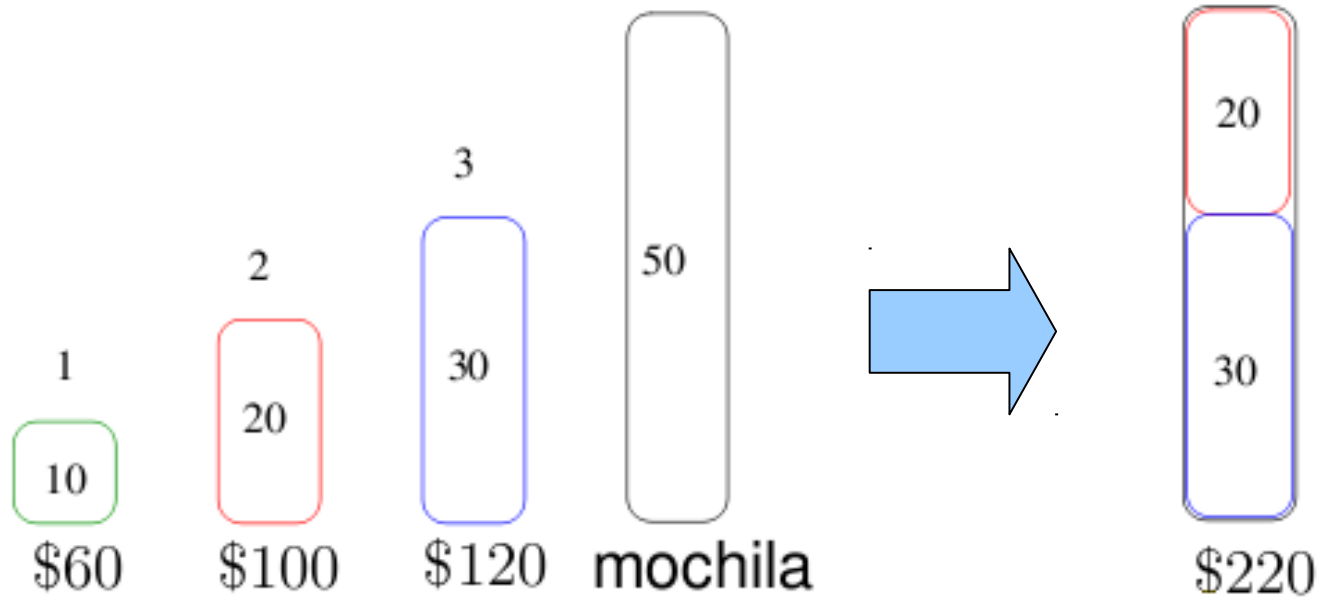
- Exemplo 2: problema da mochila

- Problema da mochila binária:
  - A solução gulosa foi ótima?

Algoritmos gulosos

- Exemplo 2: problema da mochila

- Problema da mochila binária:
  - A solução gulosa foi ótima?
    - Obviamente não. A ótima seria:



valor/peso = 6 5 4

Referências

- Nívio Ziviani. Projeto de Algoritmos com implementações em C e Pascal. Editora Thomson, 2a. Edição, 2004 (texto base)
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein. Algoritmos - Tradução da 2a. Edição Americana. Editora Campus, 2002.
- Notas de aula – Prof. Norton Roman – EACH-USP
- Notas de aula – Prof. Delano Beder – EACH-USP

Paradigmas de Projeto de Algoritmos

Algoritmos gulosos

Professora:

Fátima L. S. Nunes