# 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 de BSI?)

# 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

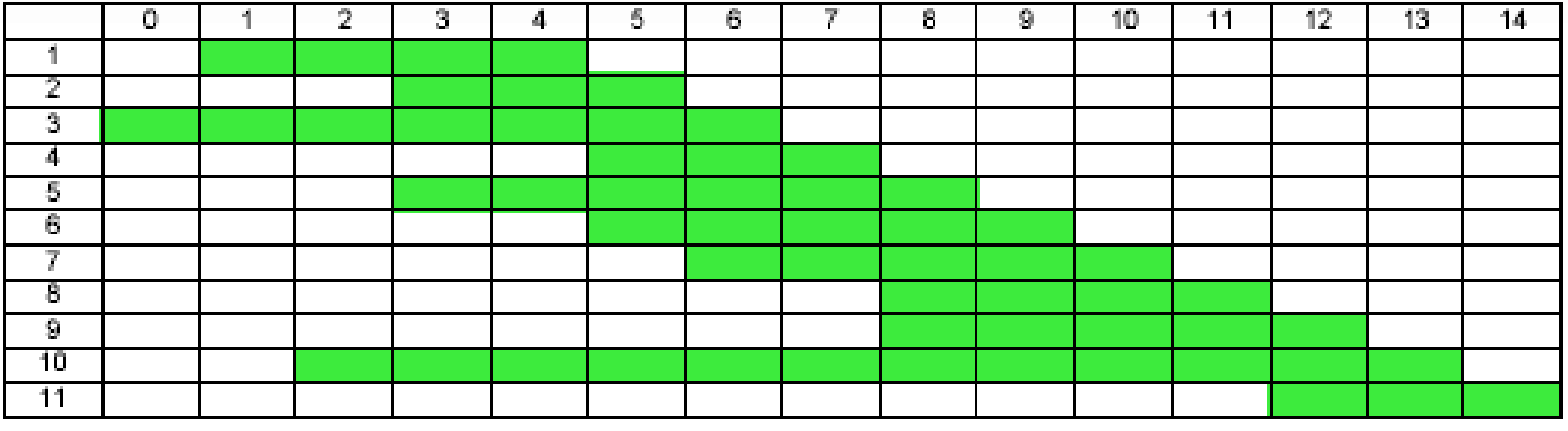
- Exemplo 1: locação de atividades em uma sala
  - existem diversas atividades (por exemplo aulas) que querem usar uma mesma sala;
  - cada atividade tem um horário de início e um horário de fim;
  - só existe uma sala disponível;
  - duas aulas não podem ser ministradas na mesma sala ao mesmo tempo.

# 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.

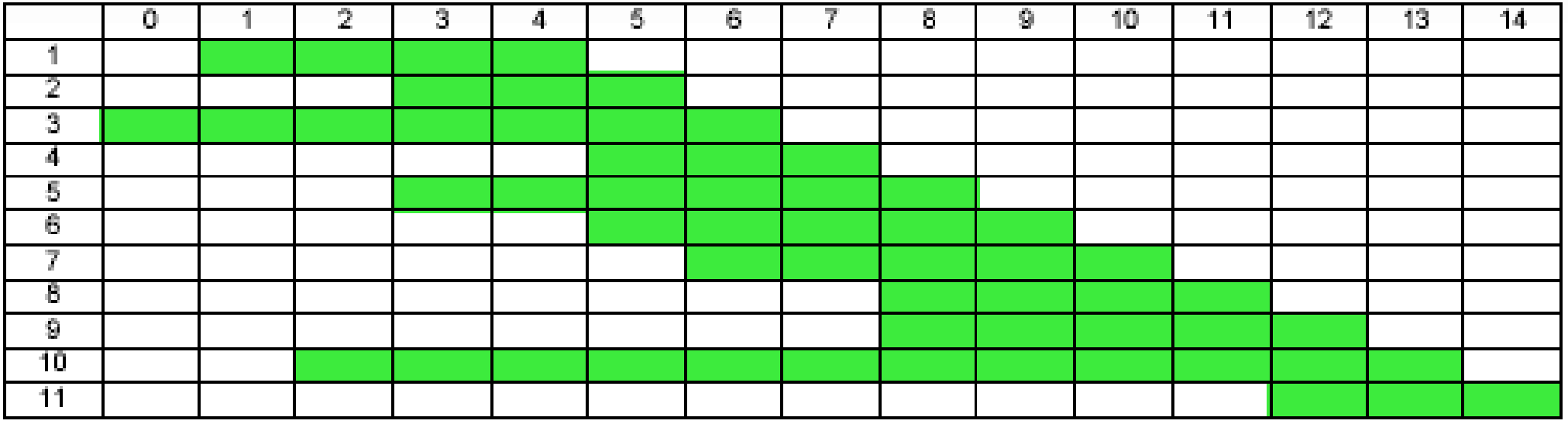
  - Sugestões ???



# Algoritmos gulosos

- Exemplo 1: locação de atividades em uma sala

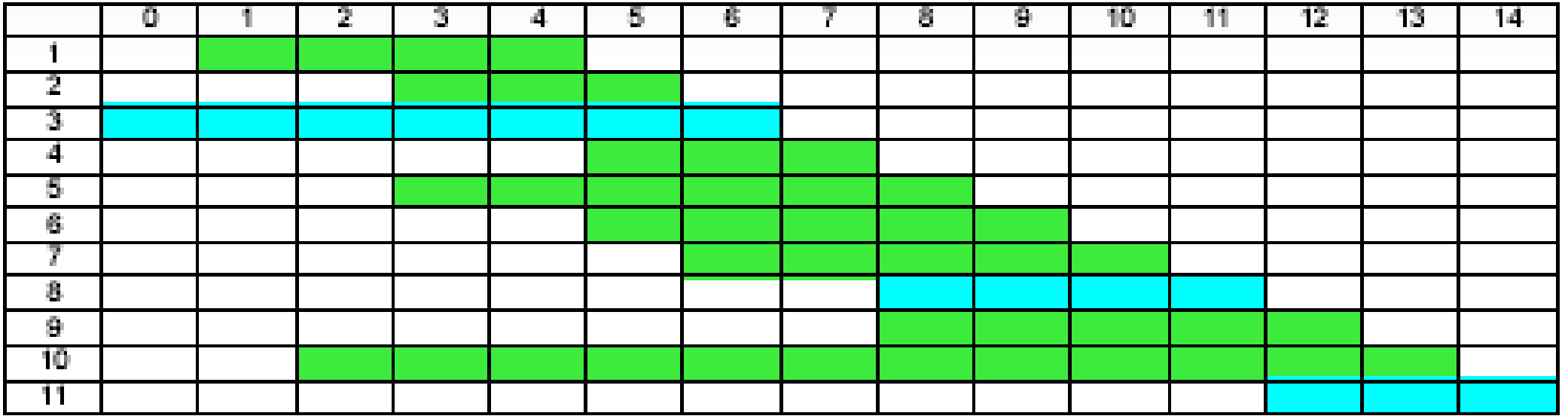
  - Sugestões ???
    - começar pelas atividades que começam primeiro
    - começar pelas atividades que terminam primeiro
    - começar pelas atividades mais longas primeiro
    - começar pelas atividades mais curtas primeiro



# Algoritmos gulosos

- Exemplo 1: locação de atividades em uma sala

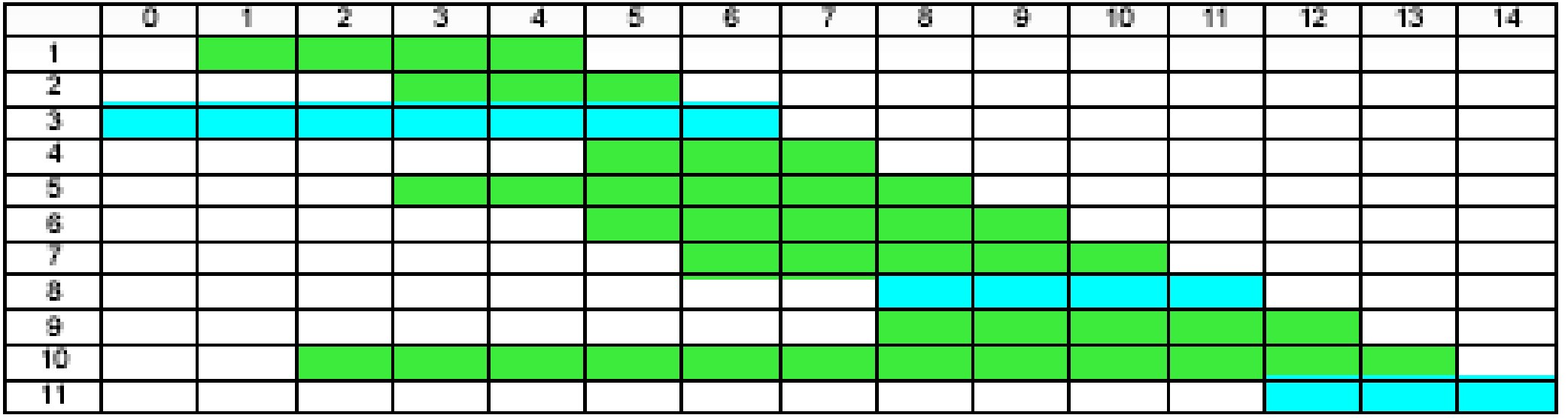
  - Primeira tentativa:
    - começar pelas atividades que começam primeiro
    - Solução foi boa?



# Algoritmos gulosos

- Exemplo 1: locação de atividades em uma sala

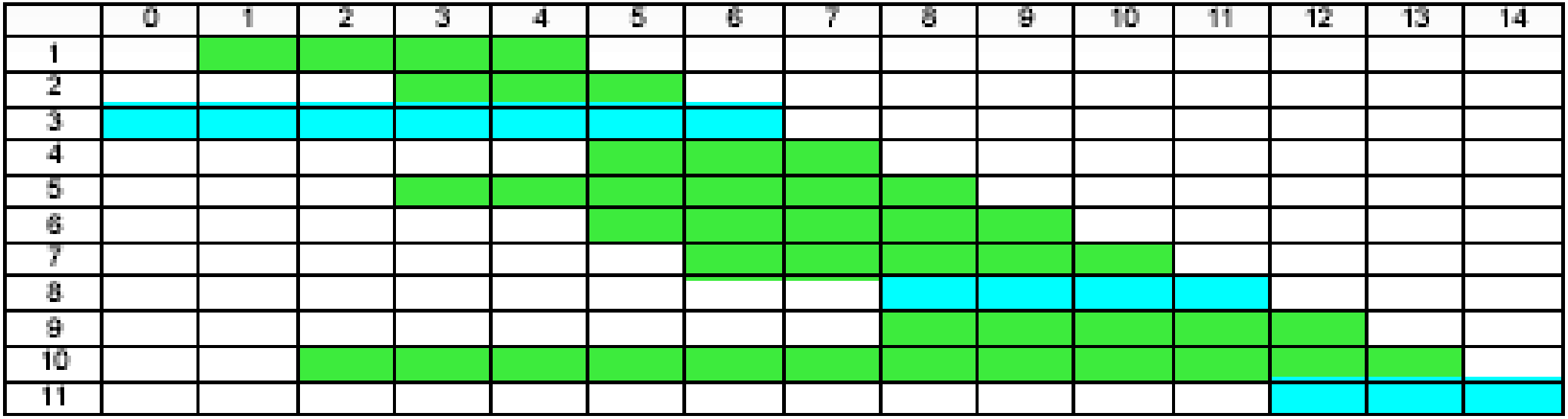
  - Primeira tentativa:
    - começar pelas atividades que começam primeiro
    - Solução foi boa?
      - escolheu 3 atividades: 3, 8 e 11
      - quantas poderiam ter sido escolhidas?



# Algoritmos gulosos

- Exemplo 1: locação de atividades em uma sala

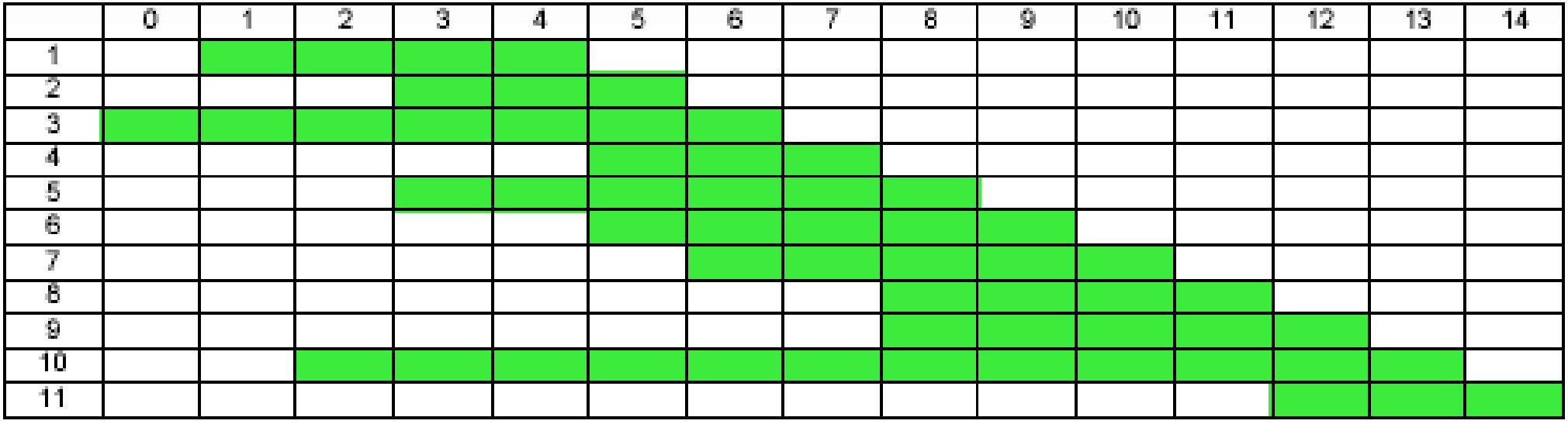
  - Primeira tentativa:
    - começar pelas atividades que começam primeiro
    - Solução foi boa?
      - escolheu 3 atividades: 3, 8 e 11
      - quantas poderiam ter sido escolhidas?
        - 4 atividades: 1, 4, 8 e 11



# Algoritmos gulosos

- Exemplo 1: locação de atividades em uma sala

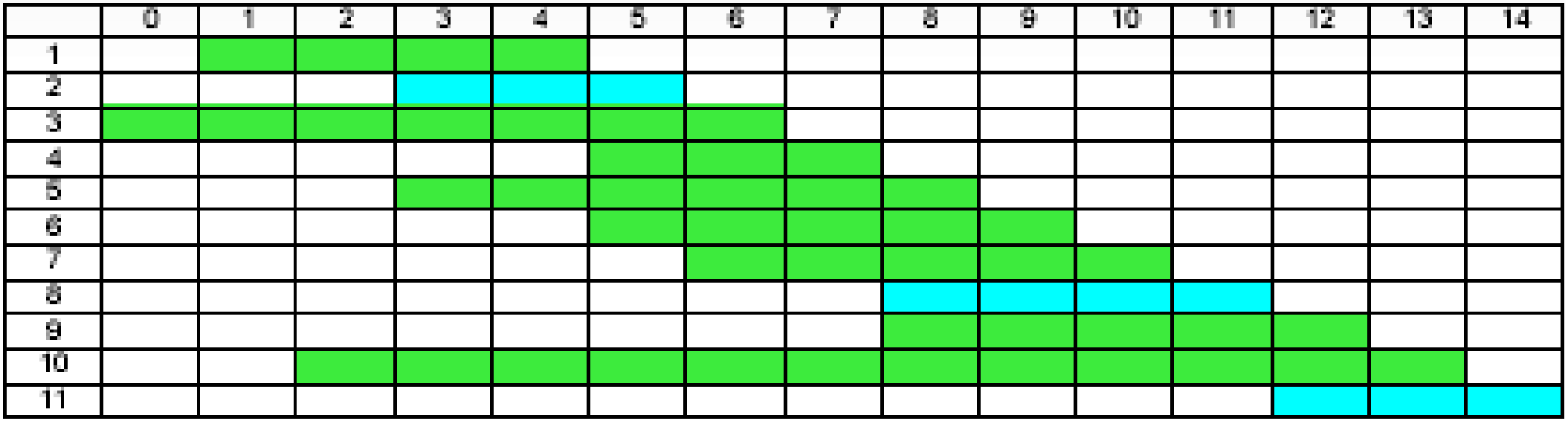
  - Segunda tentativa:
    - começar pelas atividades que duram menos tempo
    - Como fica?



# Algoritmos gulosos

- Exemplo 1: locação de atividades em uma sala

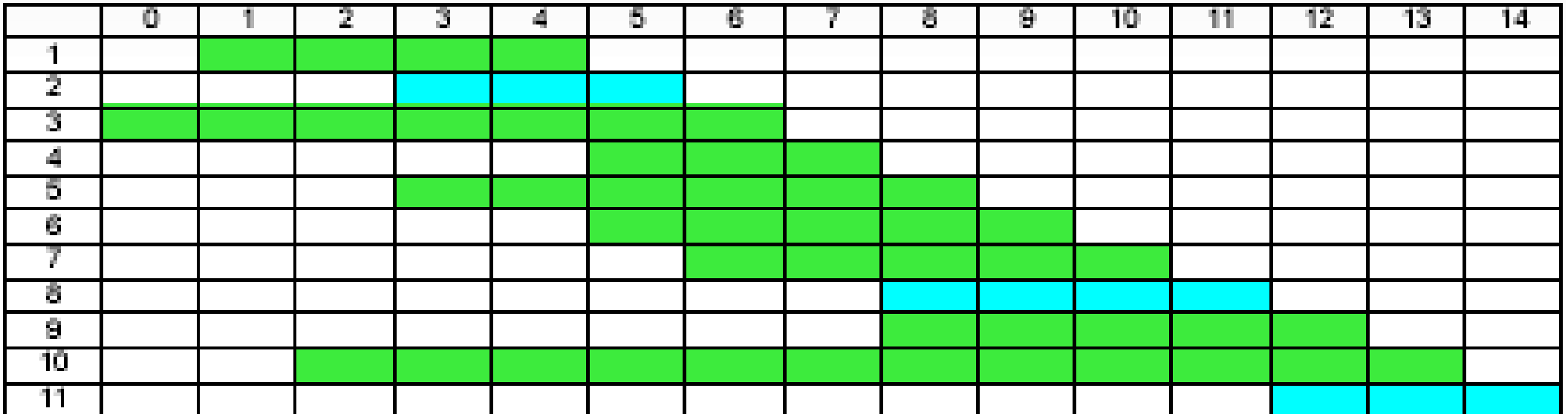
  - Segunda tentativa:
    - começar pelas atividades que duram menos tempo
    - Como fica?



# Algoritmos gulosos

- Exemplo 1: locação de atividades em uma sala

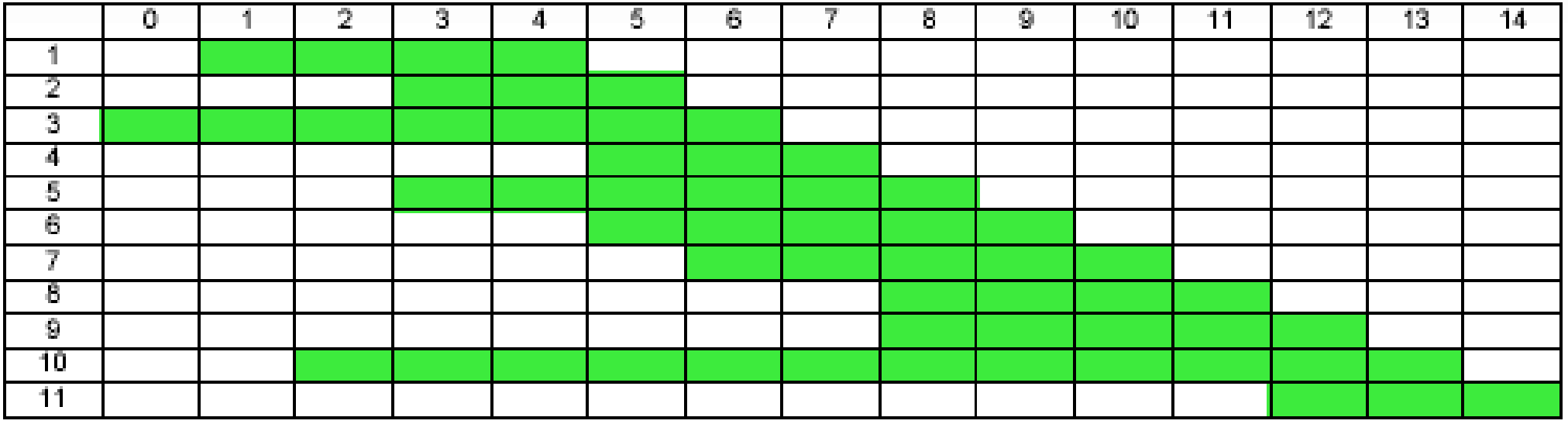
  - Segunda tentativa:
    - começar pelas atividades que duram menos tempo
    - Solução foi boa?
      - escolheu 3 atividades: 2, 8 e 11
      - quantas poderiam ter sido escolhidas?
        - 4 atividades: 1, 4, 8 e 11



# Algoritmos gulosos

- Exemplo 1: locação de atividades em uma sala

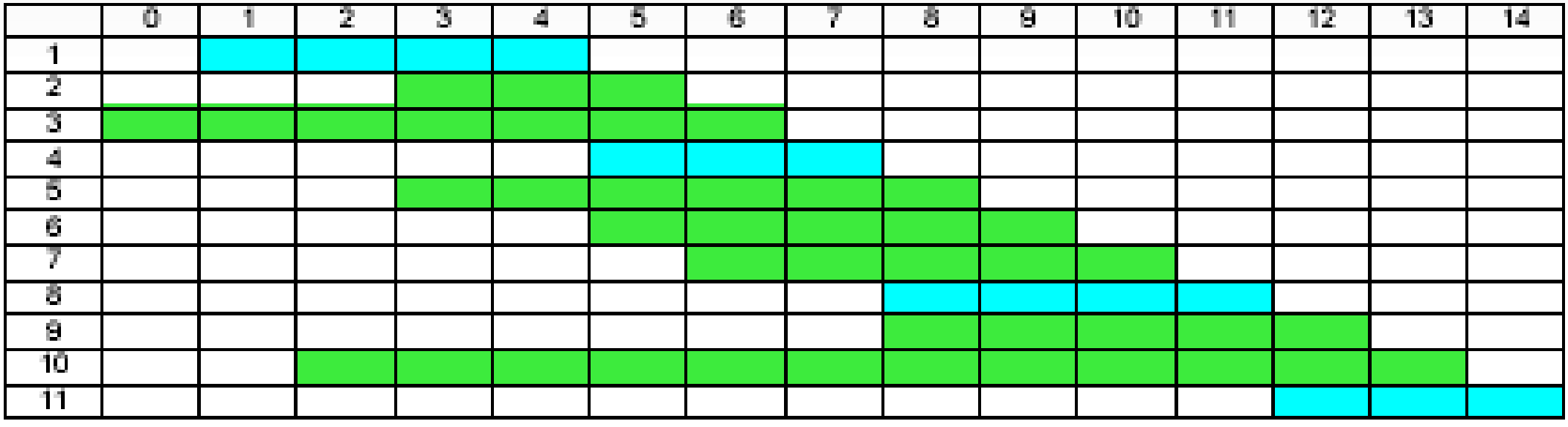
  - Terceira tentativa:
    - começar pelas atividades que terminam primeiro
    - Como fica?



# Algoritmos gulosos

- Exemplo 1: locação de atividades em uma sala

  - Terceira tentativa:
    - começar pelas atividades que terminam primeiro
    - Como fica?
    - Solução foi boa?
      - escolheu 4 atividades!



# Algoritmos gulosos

- Exemplo 1: locação de atividades em uma sala

  - Algoritmo para o problema de locação de atividades

# 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

```
public class SelecaoDeAtividadesGuloso
{
     static int selecaoGulosa(int[] ini, int[] fim, int n)
     {
          int ultimaSelecionada=0;
          int selecionadas = 0;
          if (n==0) return 0;
          // a primeira atividade é sempre selecionada
          System.out.print("a"+0 + " ");
          selecionadas++;
          for (int i=1;i<n;i++)
             if (ini[i]>=fim[ultimaSelecionada])
             {
                   System.out.print("a"+i + " ");
                   selecionadas++;
                   ultimaSelecionada = i;
             }
          System.out.println();
          return selecionadas;
     }
```

# Algoritmos gulosos

- Exemplo 1: locação de atividades em uma sala

```
...

// as atividades devem ser ordenadas pelo campo fim
// ou seja, as atividades que acabam primeiro ficam na frente
private static int[] inicio = { 1,3,0,5,3,5, 6, 8, 8, 2,12 };
private static int[] fim = { 4,5,6,7,8,9,10,11,12,13,14 };
private static int numeroDeAtividades = 11;

public static void main(String[] args)
{
     int total=selecaoGulosa(inicio,fim,numeroDeAtividades);
     System.out.println("Foram selecionadas " + total + "
          atividades.");
}

}
     ...
```

# 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

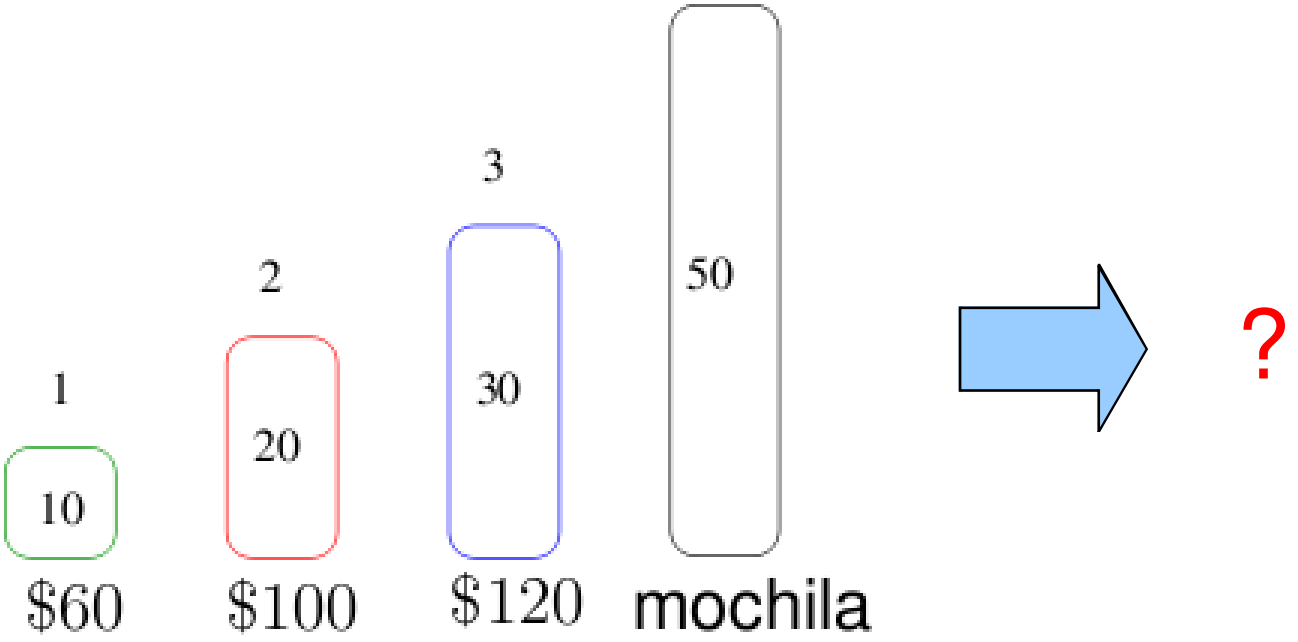
## • Exemplo 2: problema da mochila

- 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

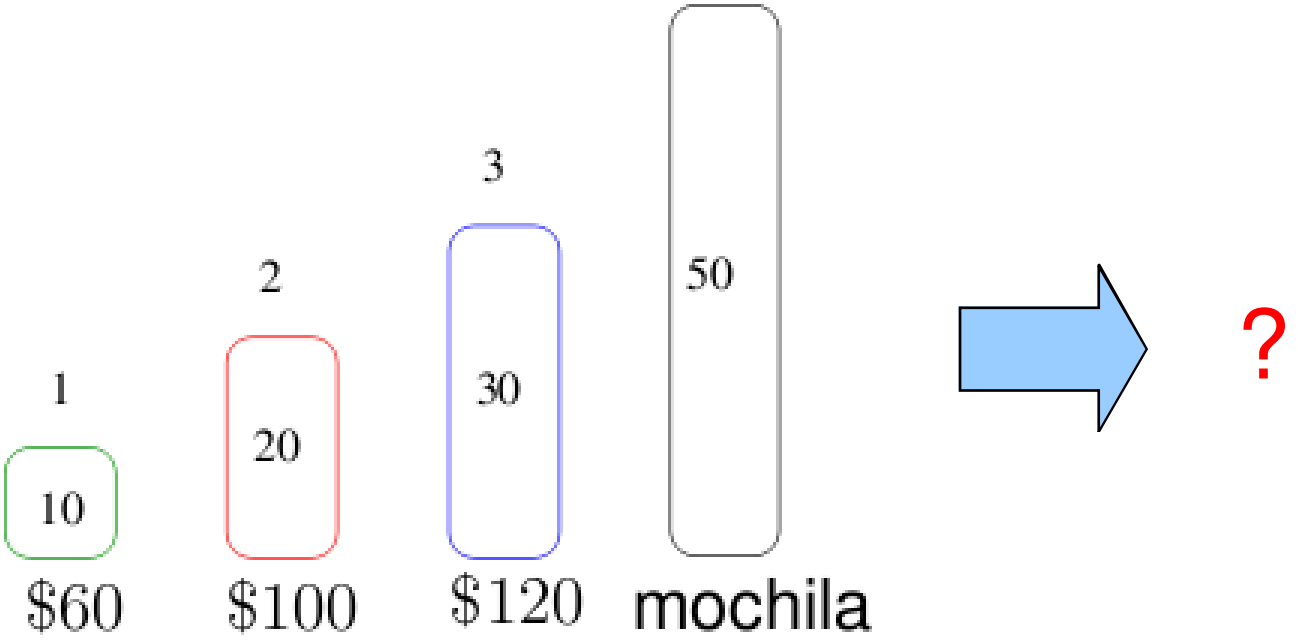
- Problema da mochila fracionada:



# Algoritmos gulosos

- Exemplo 2: problema da mochila

- Problema da mochila fracionada:

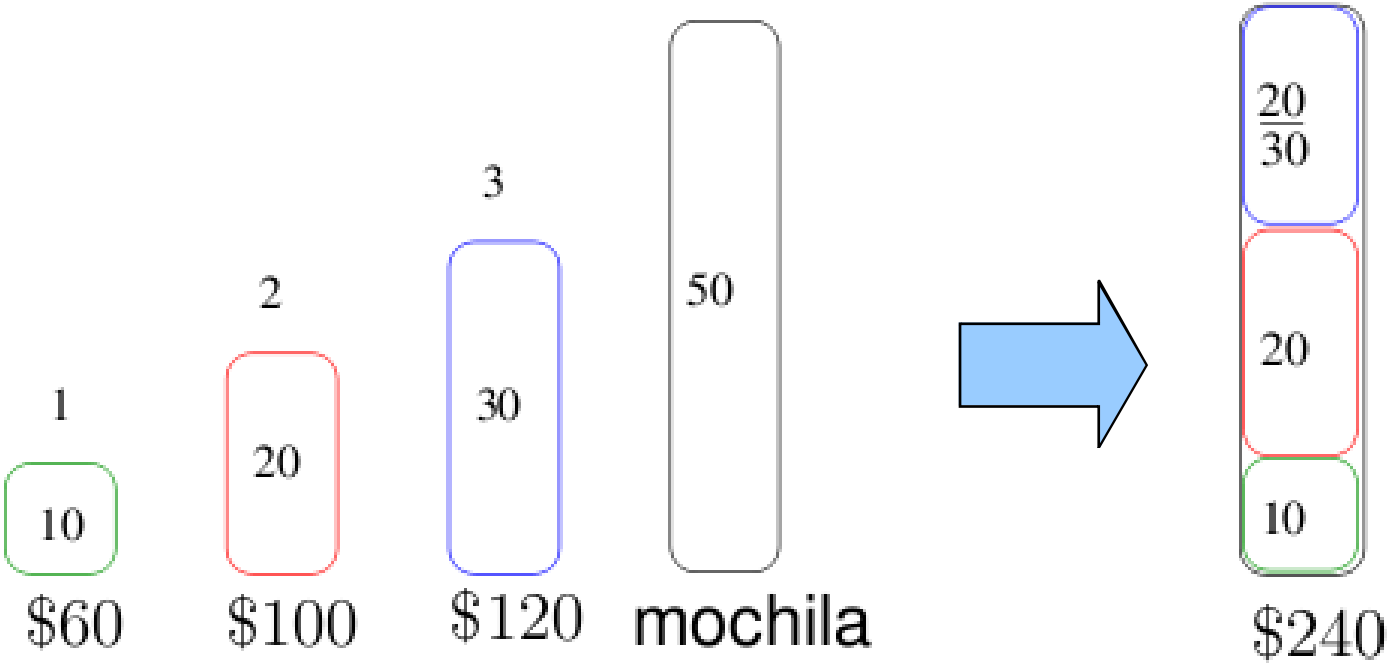



valor/peso = 6 5 4

# 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

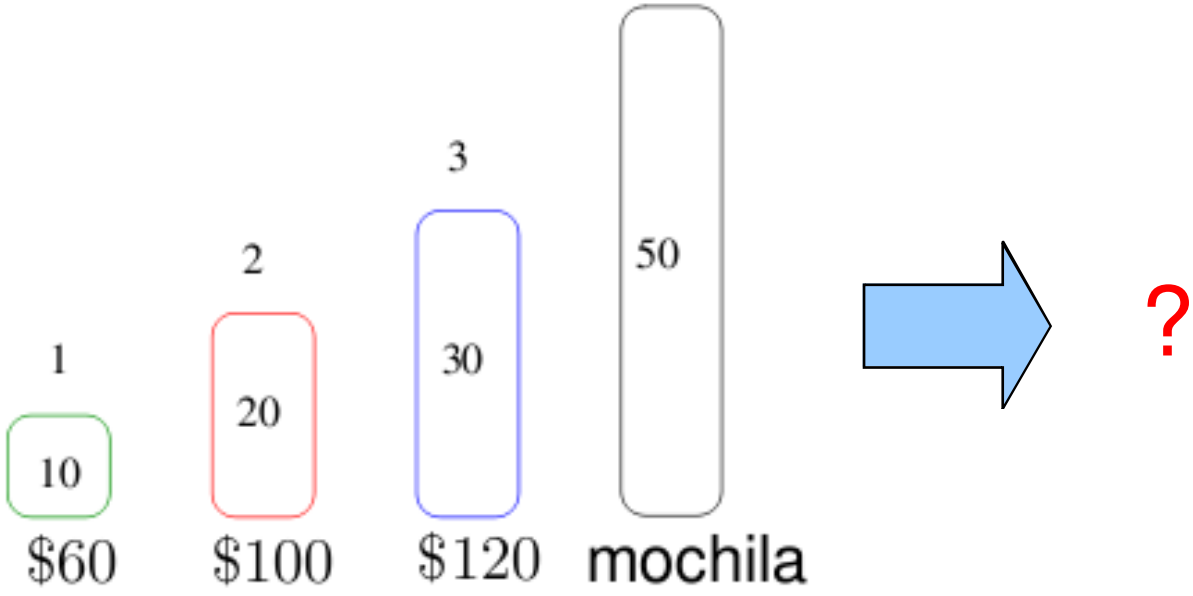
## • Exemplo 2: problema da mochila

- 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

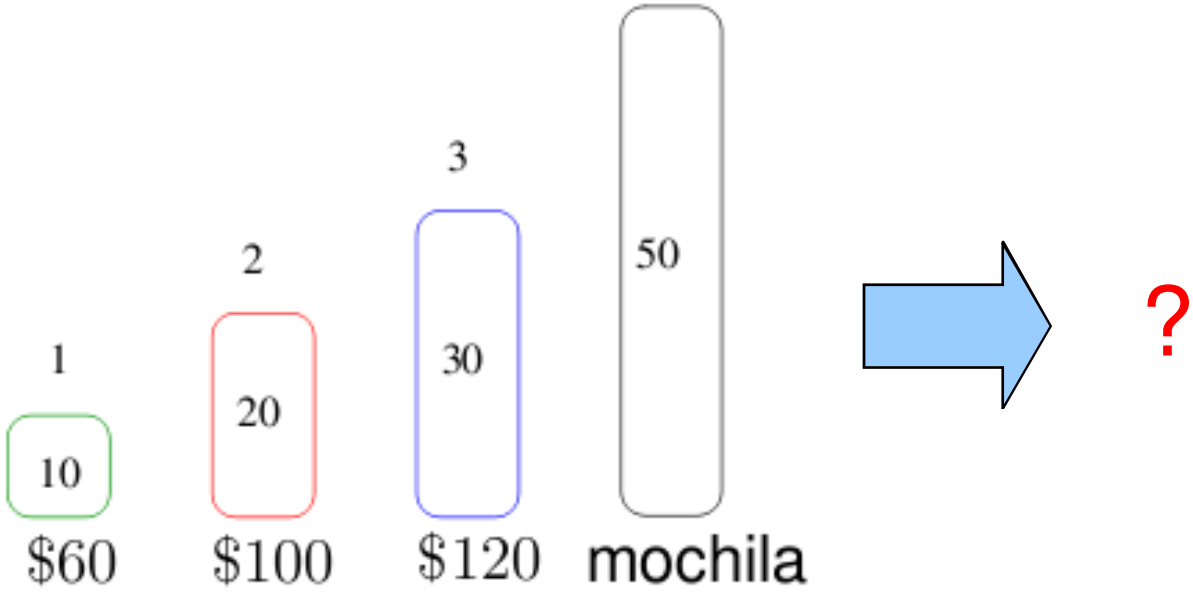
- Problema da mochila binária:



# Algoritmos gulosos

- Exemplo 2: problema da mochila

- Problema da mochila binária:

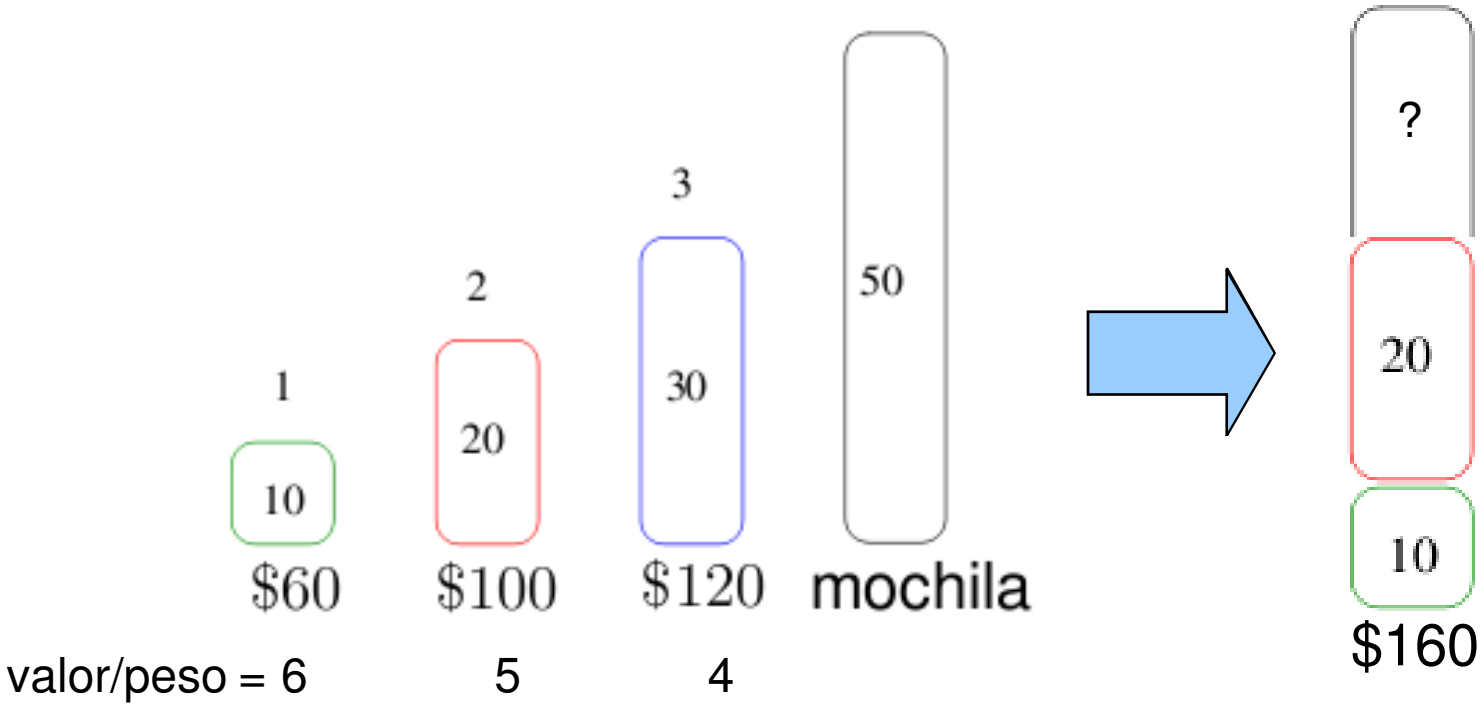



valor/peso = 6 5 4

# Algoritmos gulosos

- Exemplo 2: problema da mochila

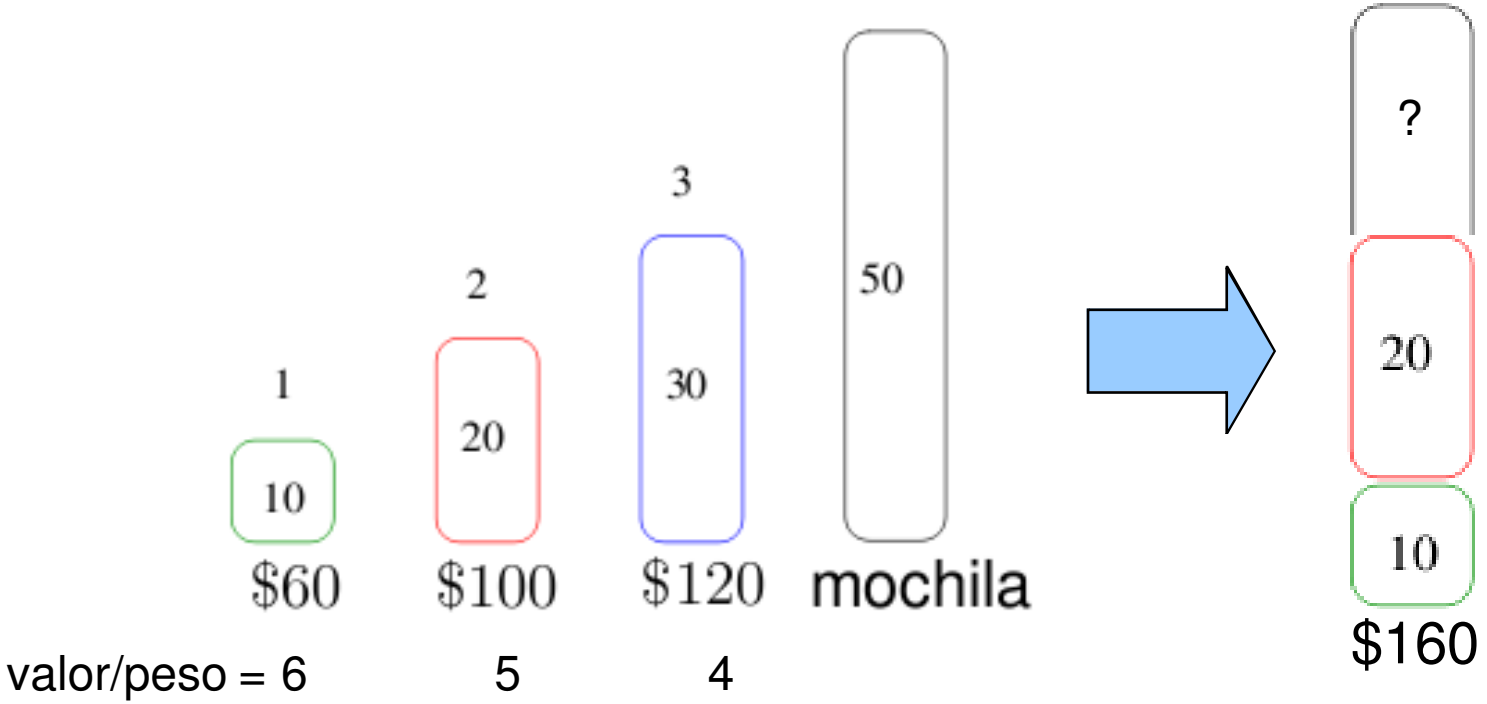
- Problema da mochila binária:



# Algoritmos gulosos

- Exemplo 2: problema da mochila

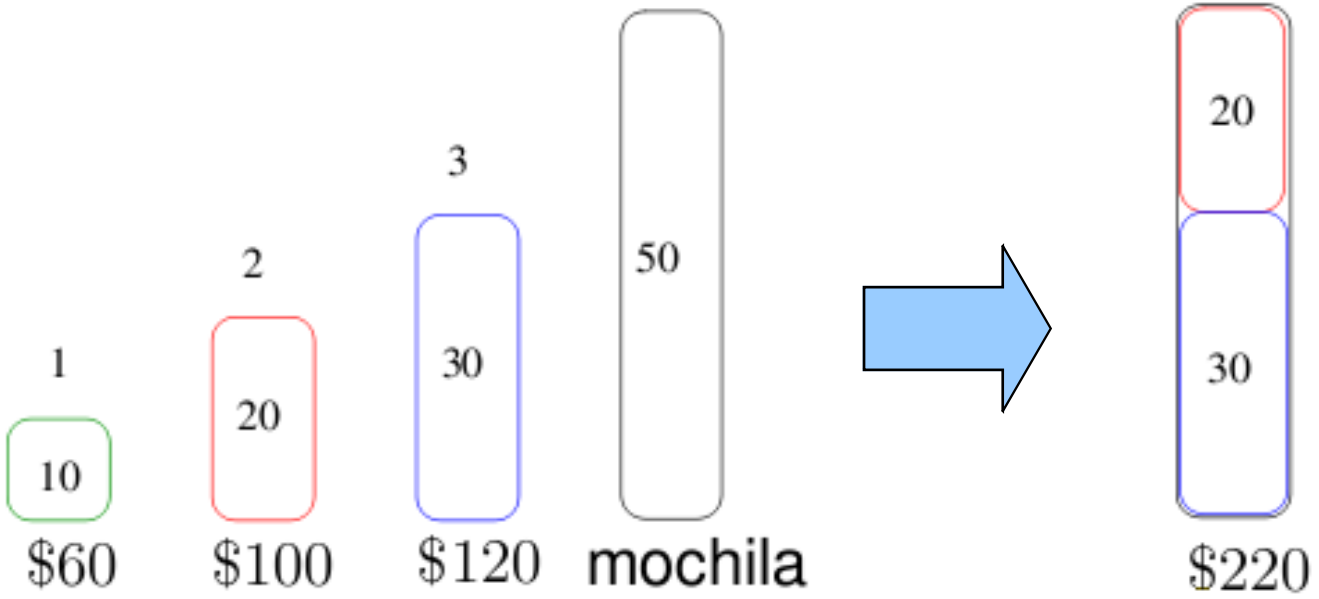
- Problema da mochila binária:
  - A solução gulosa foi ótima?



# 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

# Algoritmos gulosos

- Formalizando:
  - Dado um conjunto *C*, deseja-se determinar um subconjunto $S \subseteq C$, tal que:
    - *(i)* *S* satisfaça uma data propriedade *P;*
    - *(ii)* *S* é mínimo (ou máximo) em relação a algum critério $\alpha$ (*S* é o menor ou maior subconjunto de *C*, segundo $\alpha$, que satisfaz *P.*
- Para resolver este problema, o algoritmo guloso:
  - executa processo iterativo em que *S* é construído adicionando-se elementos de *C*, um a um.

# Algoritmos gulosos

- Como fazer um algoritmo geral?

# Algoritmos gulosos

- Como fazer um algoritmo geral?

```
Guloso (conjunto C)
{
 S= ∅;
 enquanto  (C <> 0) e não encontrou solução faça
 {
     x = seleciona (C);
     C = C – x;
     se viável (S + x)
        S = S + x;
 }
 se S é soluçao
     retorna S
 senão
     imprime (‘Não existe solução’);
}
```

# 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**