Algoritmos Gulosos

Norton T. Roman

Apostila baseada no trabalho de Delano M. Beder, Luciano Digianpietri, David Matuszek, Marco Aurelio Stefanes e Nivio Ziviani

Algoritmos Gulosos

- São aqueles que, a cada decisão:
  - Sempre escolhem a alternativa que parece mais promissora naquele instante
  - Nunca reconsideram essa decisão
    - Uma escolha que foi feita nunca é revista
    - Não há backtracking
    - Come e nunca vomita (argh!)

Algoritmos Gulosos

- Alternativa mais promissora?
  - Depende do problema, do que se quer maximizar ou minimizar
    - Ex: caminho mais curto, menor número de jogadas etc
  - Há que se ter um modo de avaliar as diferentes opções
    - Uma função que diga qual delas vale mais, diante do que se considera importante para o problema
- Por fazer a escolha que parece ser a melhor a cada iteração, diz-se que a escolha é feita de acordo com um critério guloso – decisão localmente ótima!

Algoritmos Gulosos

- Características:
  - Para construir a solução ótima existe um conjunto ou lista de candidatos.
  - São acumulados um conjunto de candidatos considerados e escolhidos, e o outro de candidatos considerados e rejeitados.
  - Existe função que verifica se um conjunto particular de candidatos produz uma solução (sem considerar otimalidade no momento).

Algoritmos Gulosos

- Características:
  - Outra função verifica se um conjunto de candidatos é viável (também sem preocupar com a otimalidade).
  - Uma função de seleção indica a qualquer momento quais dos candidatos restantes é o mais promissor.
  - Uma função objetivo fornece o valor da solução encontrada
    - Como o comprimento do caminho construído
    - Não aparece de forma explicita no algoritmo guloso

Algoritmos Gulosos

- A função de seleção é geralmente relacionada com a função objetivo.
- Se o objetivo é:
  - maximizar ⇒ provavelmente escolherá o candidato restante que proporcione o maior ganho individual.
  - minimizar ⇒ então será escolhido o candidato restante de menor custo.
  - Tipicamente, um dos “segredos” dos algoritmos gulosos é a escolha de como o conjunto de entrada será ordenado.

Algoritmos Gulosos

- O algoritmo nunca muda de idéia:
  - Uma vez que um candidato é escolhido e adicionado à solução ele lá permanece para sempre.
  - Uma vez que um candidato é excluído do conjunto solução, ele nunca mais é reconsiderado.

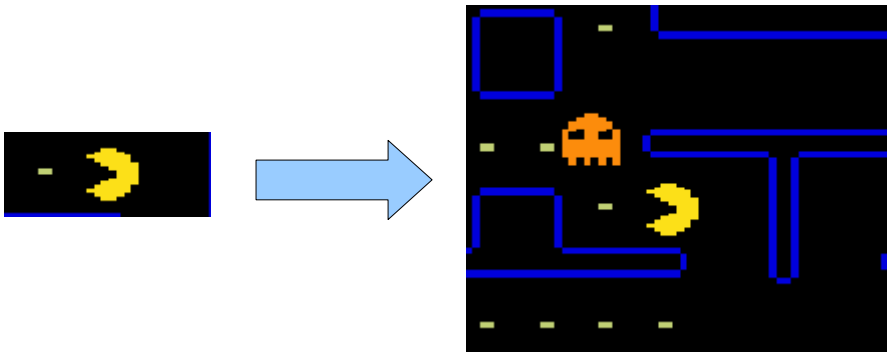
Algoritmos Gulosos

- Tipicamente algoritmos gulosos são utilizados para resolver problemas de otimização que funcionem através de uma seqüência de passos.
- Nem sempre dão soluções ótimas, embora muitas vezes o façam

???
Para onde ir
???

Algoritmos Gulosos

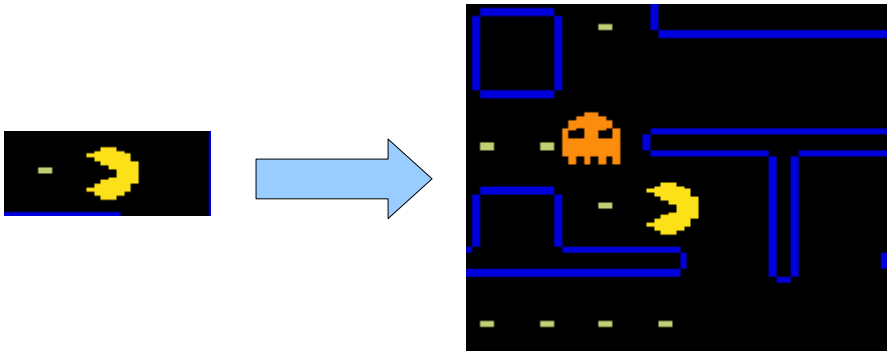
- Tipicamente algoritmos gulosos são utilizados para resolver problemas de otimização que funcionem através de uma seqüência de passos.
- Nem sempre dão soluções ótimas, embora muitas vezes o façam



Ficaria mais fácil decidir se tivéssemos mais informação

Algoritmos Gulosos

- Tipicamente algoritmos gulosos são utilizados para resolver problemas de otimização que funcionem através de uma seqüência de passos.
- Nem sempre dão soluções ótimas, embora muitas vezes o façam
  - Se pudermos provar que a escolha gulosa, combinada com as escolhas feitas até então, é ótima, então ele dará a resposta ótima



Ficaria mais fácil decidir se tivéssemos mais informação

Algoritmos Gulosos

- A idéia básica da estratégia gulosa é construir por etapas uma solução ótima.
  - Em cada passo, após selecionar um elemento da entrada (o melhor), decide-se se ele é viável (caso em que virá a fazer parte da solução) ou não.
  - Após uma seqüência de decisões, uma solução para o problema é alcançada.
  - Nessa seqüência de decisões, nenhum elemento é examinado mais de uma vez: ou ele fará parte da solução, ou será descartado.

Algoritmos Gulosos

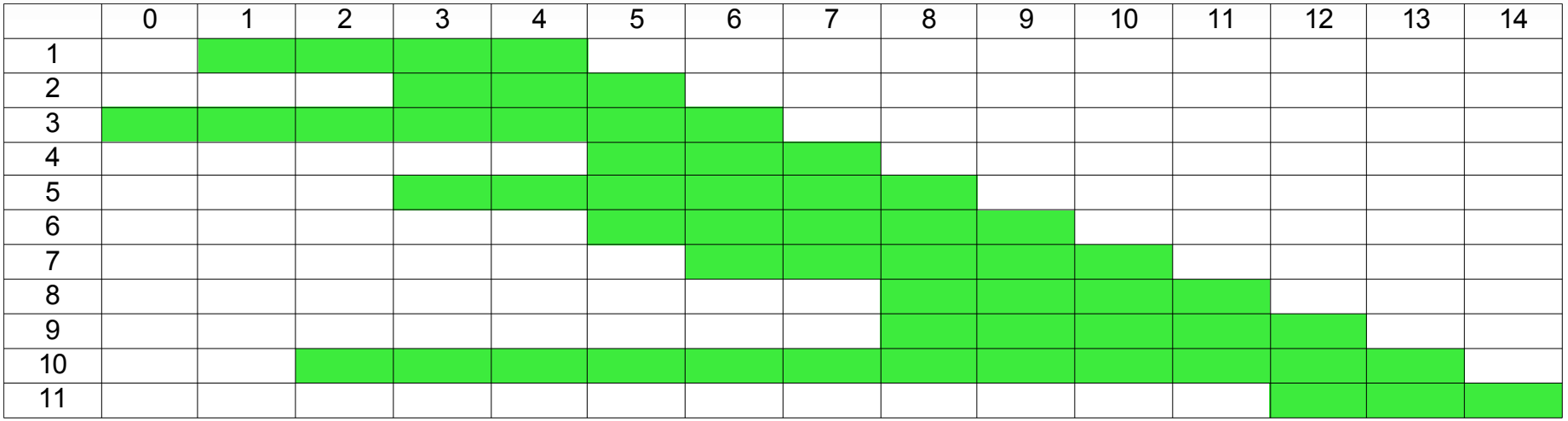
- Ingredientes chaves de um algoritmo guloso:
  - Subestrutura ótima
    - Se uma solução ótima para o problema contém, dentro dele, soluções ótimas para seus sub-problemas
  - Característica gulosa: Solução ótima global pode ser produzida a partir de uma escolha ótima local.
    - Da escolha, em um dado momento, da melhor opção dentre as existentes.

Exemplos

- Seleção de atividades:
  - Existem diversas atividades (por exemplo aulas) que querem usar um mesmo recurso (por exemplo uma sala de aulas).
  - Cada atividade tem um horário de início e um horário de fim.
  - Só existe uma sala disponível.
  - Duas aulas não podem podem ser ministradas na mesma sala ao mesmo tempo.

Exemplos

- Seleção de atividades:
  - 11 atividades, em 14 unidades de tempo



Exemplos

- Seleção de atividades:
  - Objetivo: selecionar um conjunto máximo de atividades compatíveis.
    - Atividades compatíveis são atividades que não têm sobreposição de tempo.
    - Então, criar o maior grupo de atividades sem que haja sobreposição de tempo

Exemplos

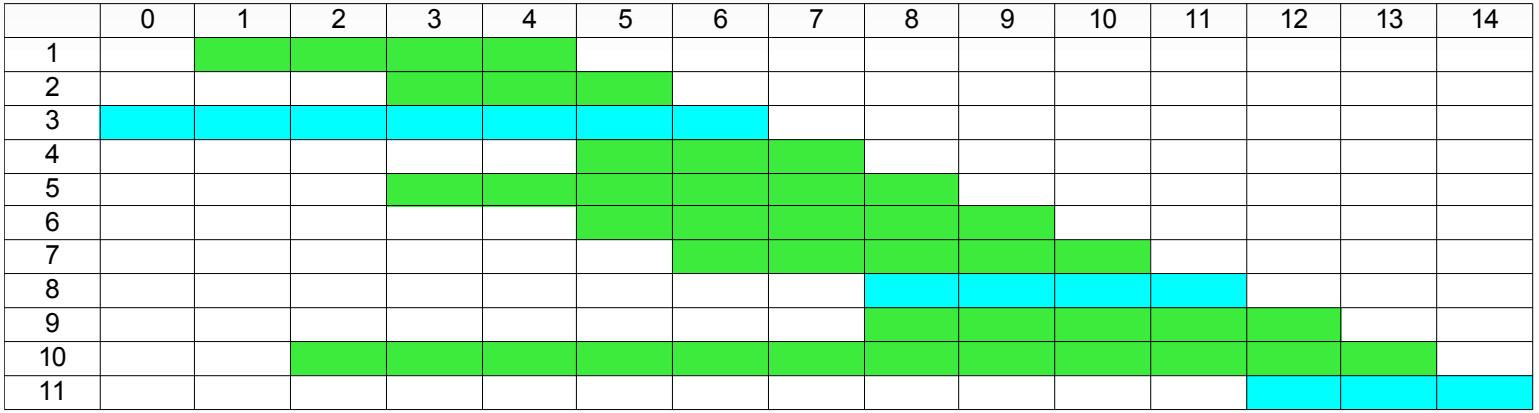
- Seleção de atividades:
  - Como procederíamos?
  - O que fará de uma atividade mais promissora que outra?

Exemplos

- Seleção de atividades:
  - Tentativa 1:
    - Escolher primeiro as atividades começam primeiro.
    - Como fica?

Exemplos

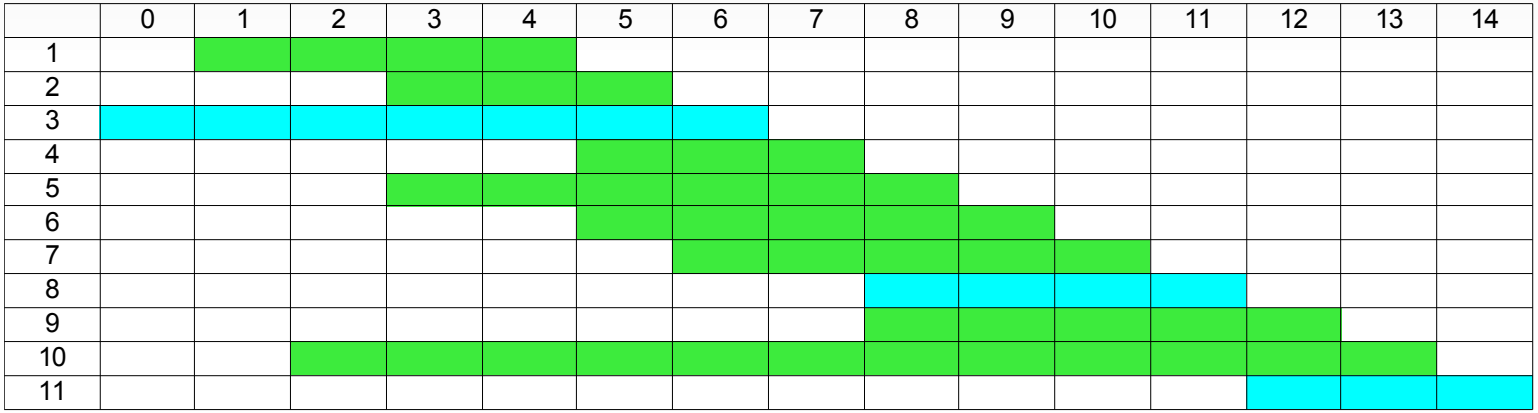
- Seleção de atividades:
  - Tentativa 1:
    - Escolher primeiro as atividades começam primeiro.



    - Foi boa a solução?

Exemplos

- Seleção de atividades:
  - Tentativa 1:
    - Escolher primeiro as atividades começam primeiro.



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

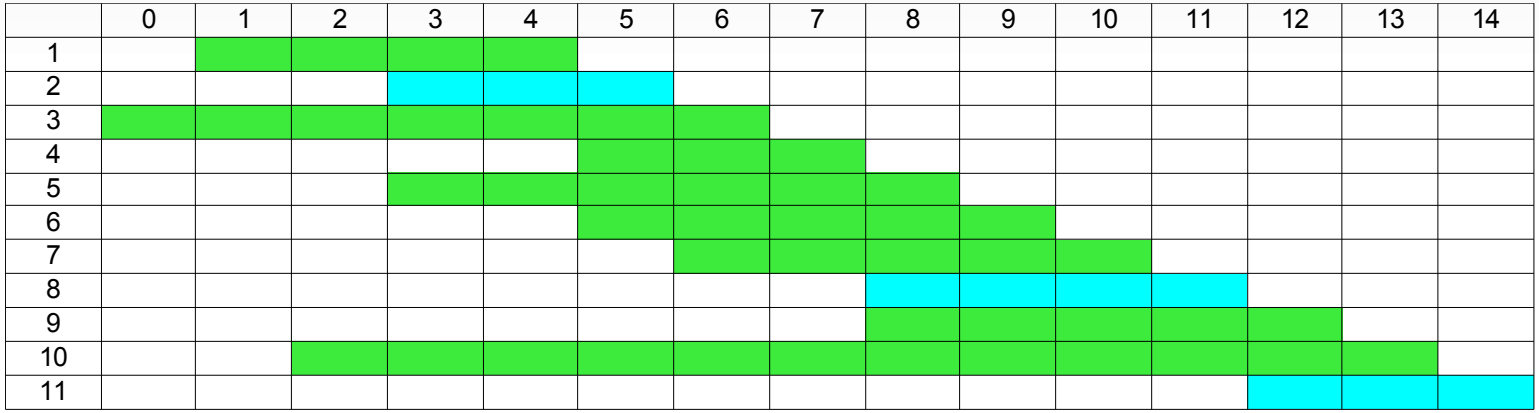
    - Foi boa a solução?
      - Não foi muito bom: escolheu apenas as atividades 3, 8 e 11, quando poderia ter escolhido quatro (1, 4, 8 e 11).

Exemplos

- Seleção de atividades:
  - Tentativa 2:
    - Escolher primeiro as atividades que demoram menos tempo
    - Como fica?

Exemplos

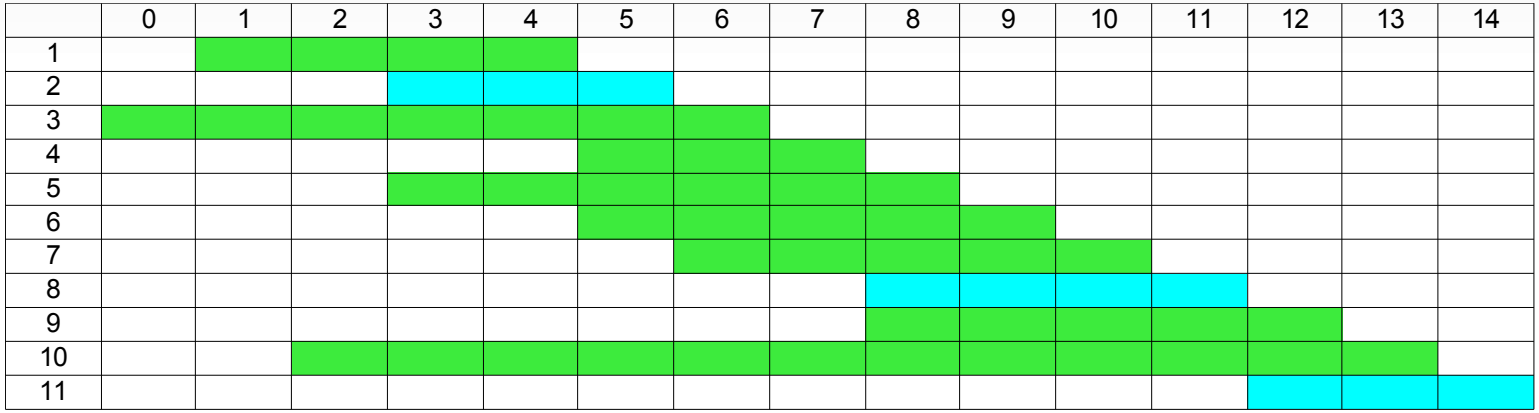
- Seleção de atividades:
  - Tentativa 2:
    - Escolher primeiro as atividades que demoram menos tempo



    - Foi boa a solução?

Exemplos

- Seleção de atividades:
  - Tentativa 2:
    - Escolher primeiro as atividades que demoram menos tempo



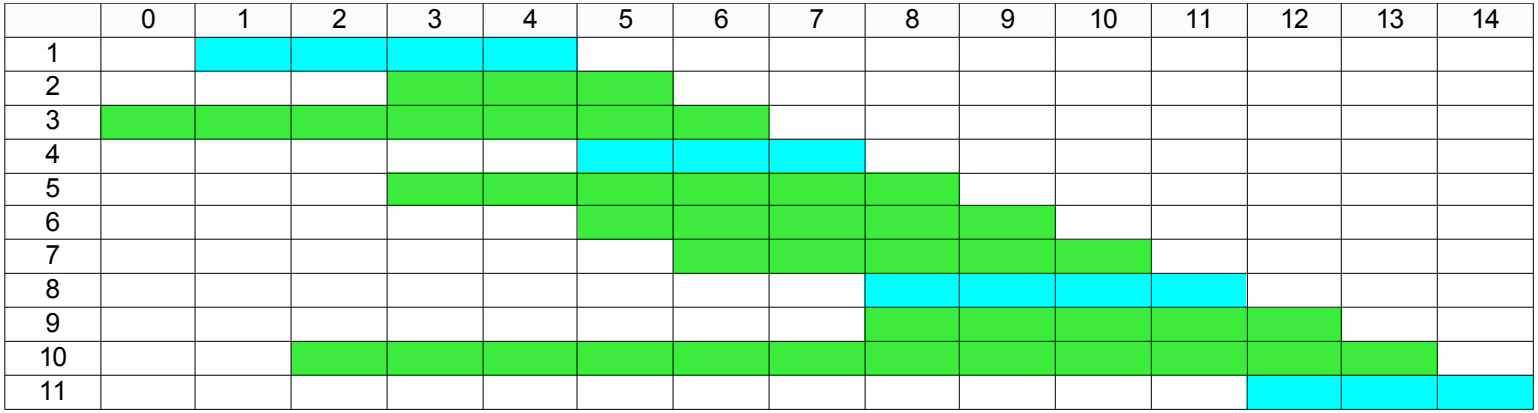
    - Foi boa a solução?
      - Não foi muito bom: escolheu apenas as atividades 2, 8 e 11, quando poderia ter escolhido quatro (1, 4, 8 e 11).

Exemplos

- Seleção de atividades:
  - Tentativa 3:
    - Escolher primeiro as atividades terminam primeiro.
    - Como fica?

Exemplos

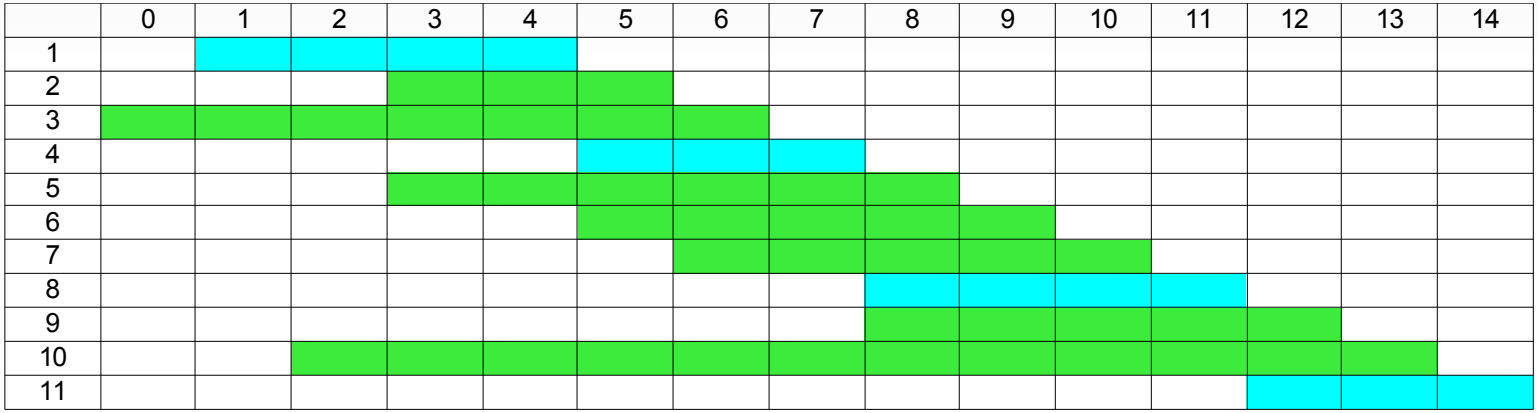
- Seleção de atividades:
  - Tentativa 3:
    - Escolher primeiro as atividades terminam primeiro.



    - Foi boa a solução?

Exemplos

- Seleção de atividades:
  - Tentativa 3:
    - Escolher primeiro as atividades terminam primeiro.



    - Foi boa a solução?
      - Agora sim. É possível demonstrar que essa solução é ótima.

Exemplos

- Seleção de atividades – funcionamento do algoritmo guloso:
  - 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
    - Note que a atividade atual é a que termina primeiro, dentre as atividades restantes

Exemplos

- Seleção de atividades:

```
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;
    }
    ...
```

Exemplos

- Seleção de atividades:

```
    ...

    // 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.");
    }
}
```

Exemplos

- Problema da mochila:
  - Dados:
    - Uma mochila que admite um certo peso;
    - Um conjunto de objetos, cada um com um valor e um peso;
  - Objetivo:
    - Selecionar o conjunto de objetos que caibam dentro da mochila de forma a maximizar o valor total dentro da mochila.

Exemplos

- Problema da mochila:
  - Este problema se divide em dois subproblemas distintos:
    - Os 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

Exemplos

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

Exemplos

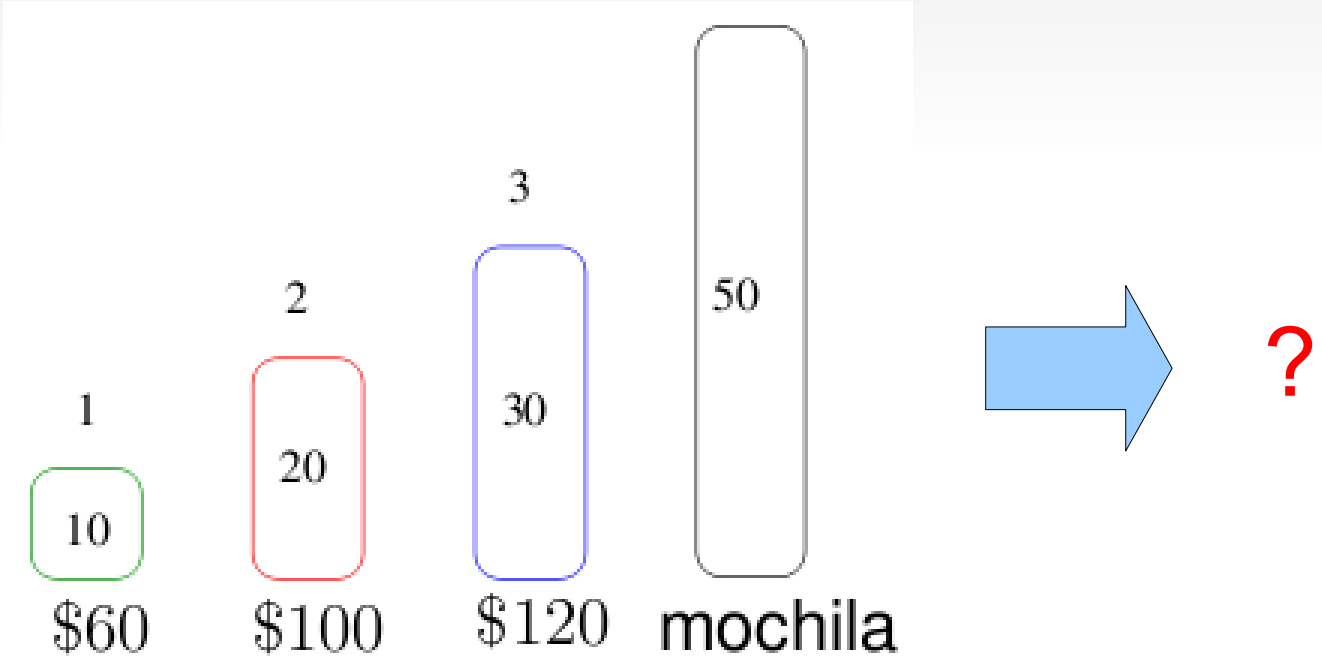
- Problema da mochila fracionada:
  - Qual seria a melhor ordenação da entrada?
    - ordenar pelo valor/peso
  - A solução gulosa será ótima?
    - Sim (é demonstrável)
  - Algoritmo:

Exemplos

- Problema da mochila fracionada:
  - Qual seria a melhor ordenação da entrada?
    - ordenar pelo valor/peso
  - A solução gulosa será ótima?
    - Sim (é demonstrável)
  - Algoritmo:
    - Para resolver o Problema da Mochila Fracionada ordene os itens por valor/peso decrescentemente
    - Começando em i = 1 coloque na mochila o máximo do item i que estiver disponível e for possível, e se puder levar mais passe para o próximo item.

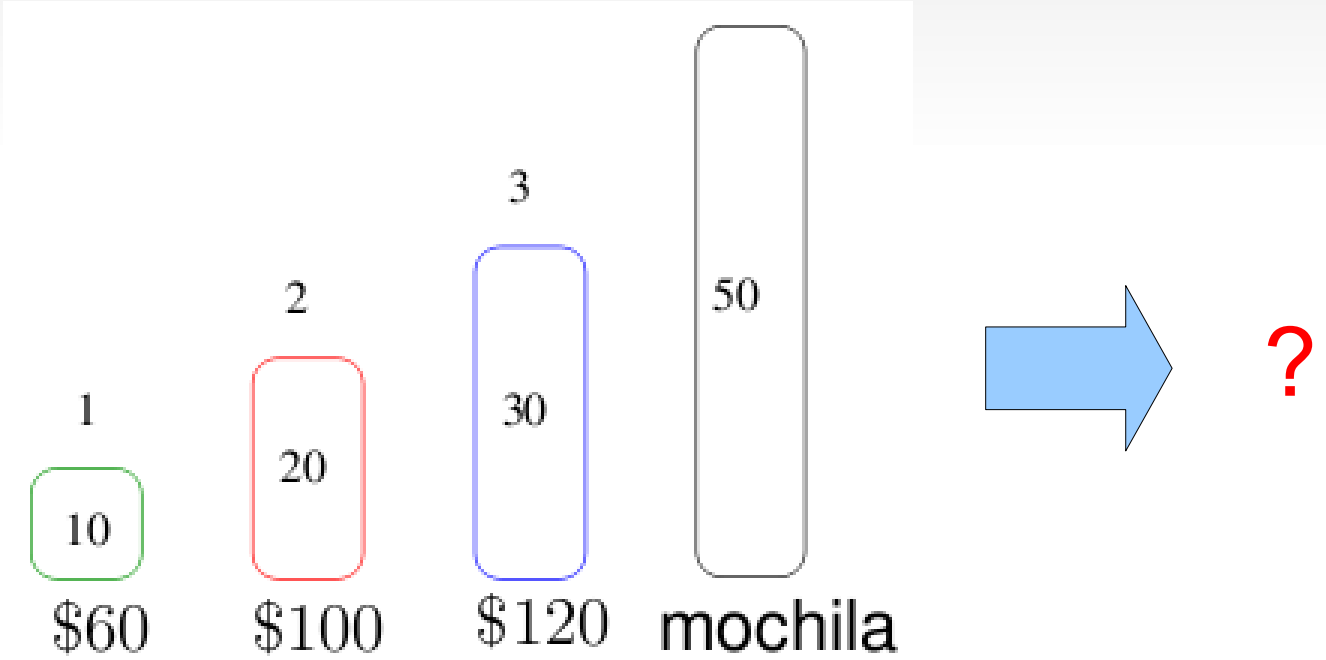
Exemplos

- Problema da mochila fracionada:



Exemplos

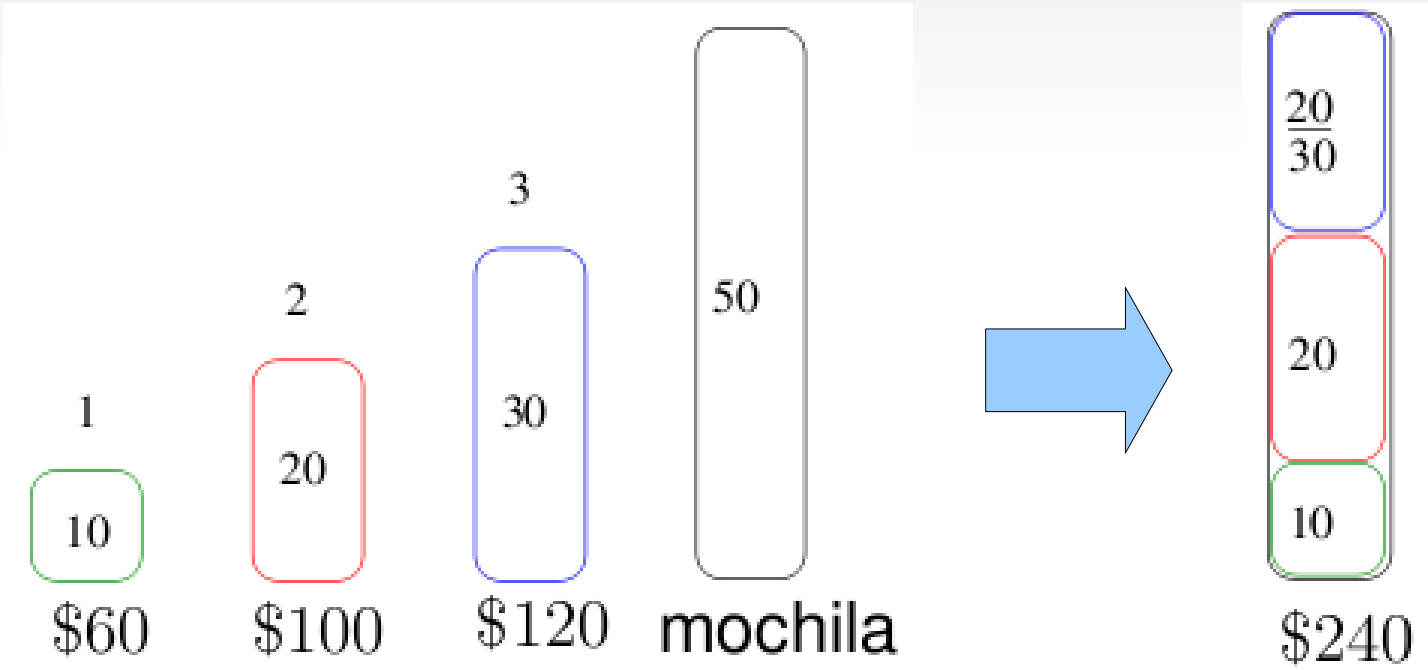
- Problema da mochila fracionada:



valor/peso = 6 5 4

Exemplos

- Problema da mochila fracionada:



valor/peso = 6 5 4

Exemplos

- 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++
}
```

Exemplos

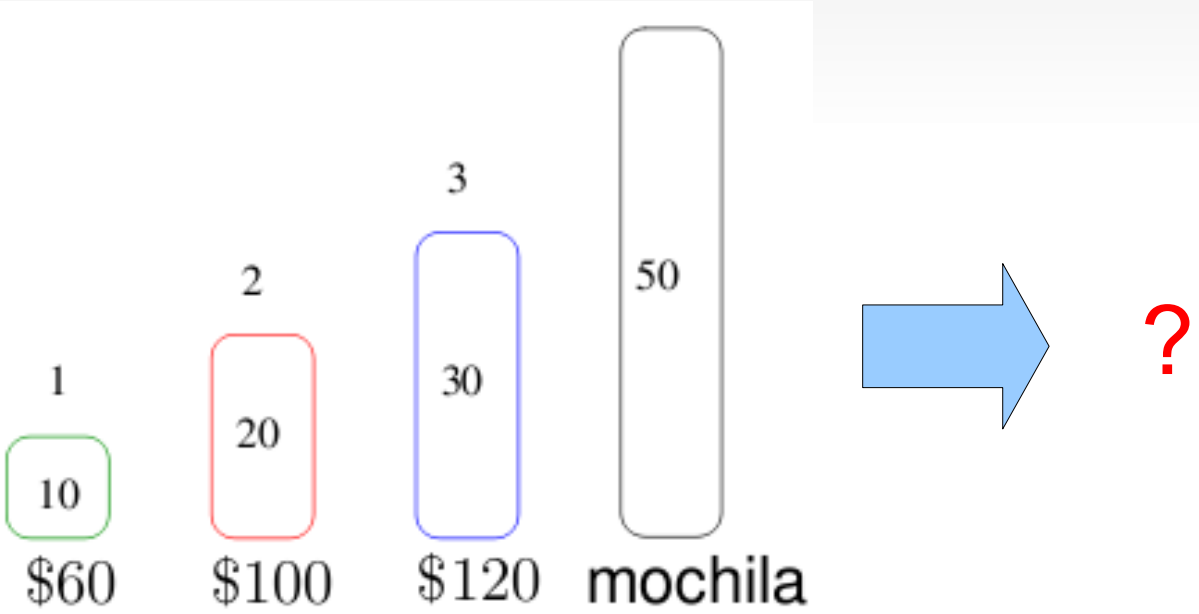
- Problema da mochila binária:
  - Qual seria a melhor ordenação da entrada?

Exemplos

- Problema da mochila binária:
  - Qual seria a melhor ordenação da entrada?
    - ordenar pelo valor/peso
  - Algoritmo:
    - Para resolver o Problema da Mochila Fracionada ordene os itens por valor/peso decrescentemente
    - Começando em i = 1 coloque na mochila o máximo do item i que estiver disponível e for possível, e se puder levar mais passe para o próximo item.
  - A solução gulosa será ótima?

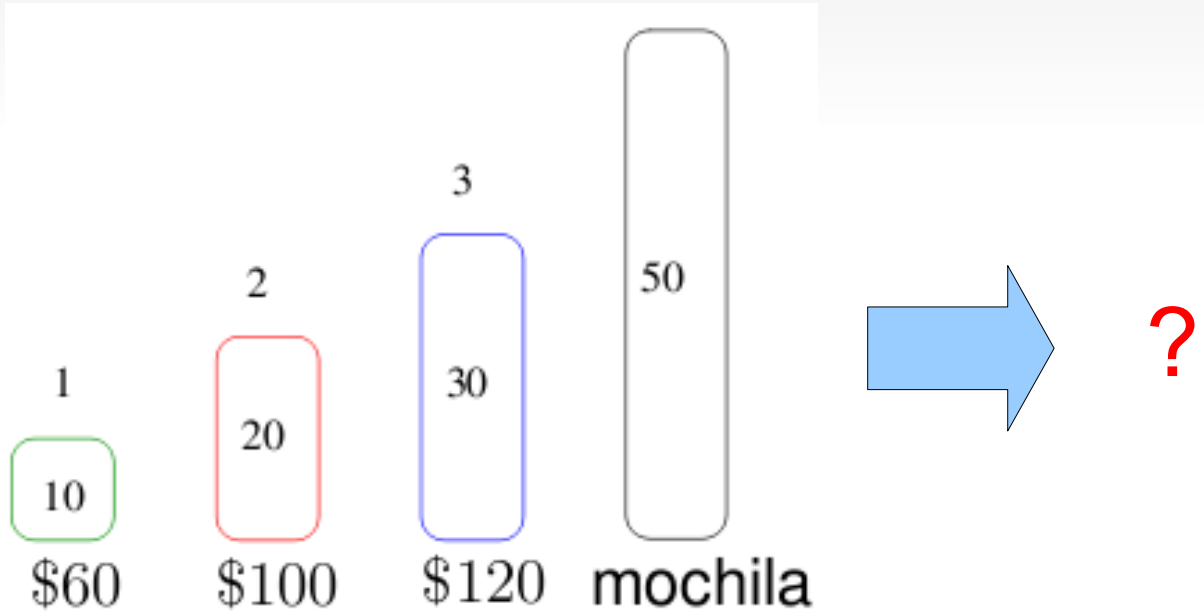
Exemplos

- Problema da mochila binária:



Exemplos

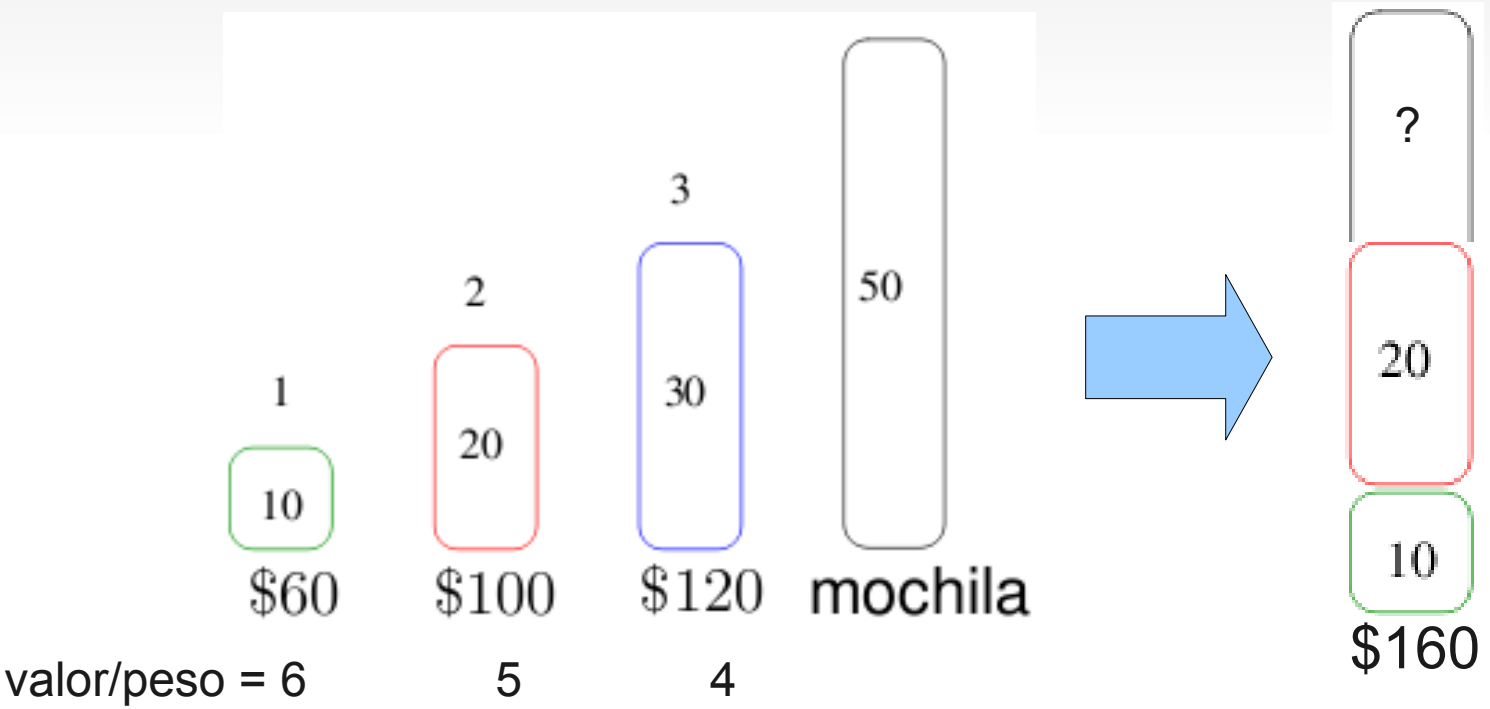
- Problema da mochila binária:



valor/peso = 6 5 4

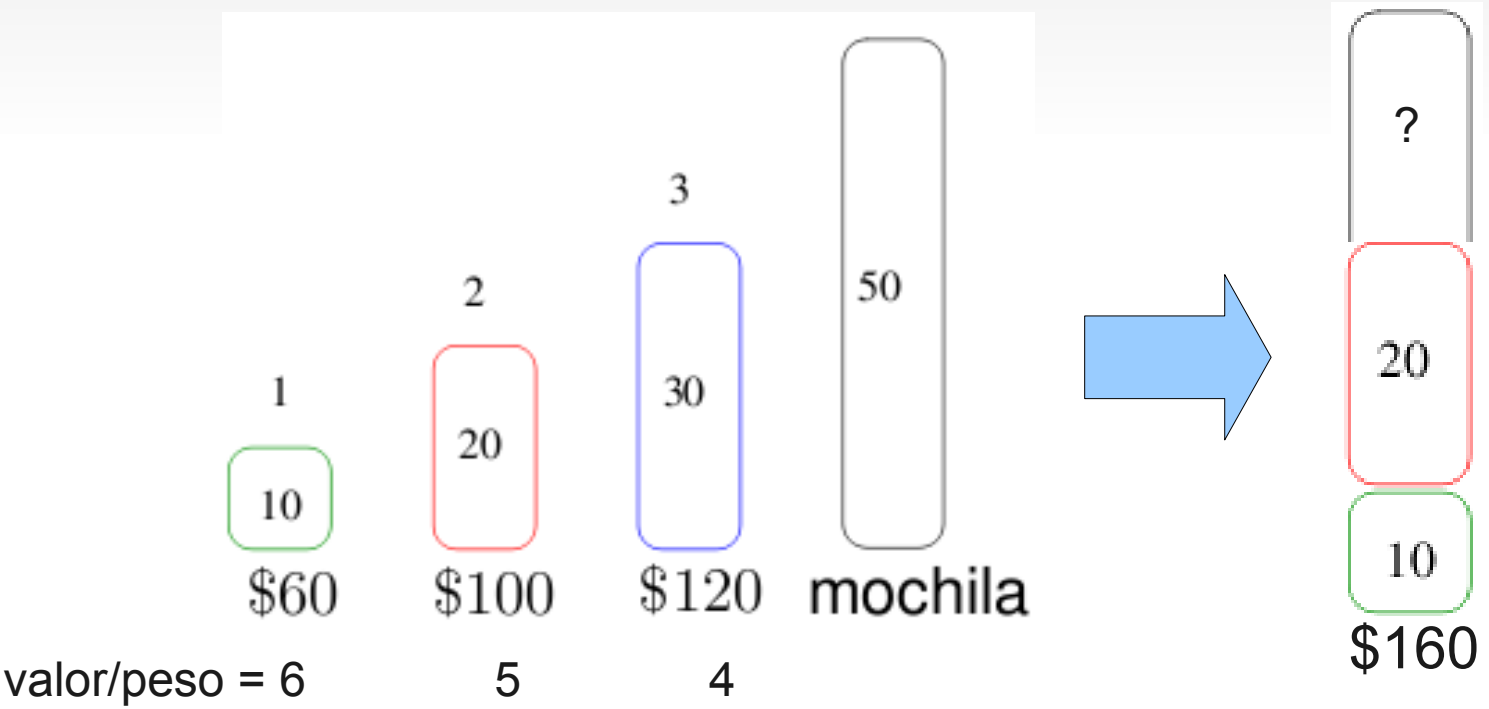
Exemplos

- Problema da mochila binária:



Exemplos

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



Exemplos

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