Algoritmos Gulosos
(Greedy ou Gananciosos)

Prof. Dr. Luciano Antonio Digiampietri

- Mais uma técnica de resolução de problemas.

- Ao contrário das técnicas anteriores, esta técnica não garante que a solução alcançada seja a ótima para qualquer tipo de problema.

- Porém, para muitos problemas os algoritmos gulosos são úteis (por serem, tipicamente, eficientes), sendo que para alguns problemas há uma garantia de solução ótima.

- É tipicamente utilizado para problemas de otimização que funcionem através de uma seqüência de passos.

- Para cada passo, o algoritmo seleciona a solução que parece ser ótima (para aquele momento).

Problema da 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.

Objetivo

- Selecionar um conjuto máximo de atividades compatíveis.
- Atividades compatíveis são atividades que não tem sobreposição de tempo.

Tentativa 1

- Escolher primeiro as atividades começam primeiro.

Ativ	0	1	2	3	4	5	6	7
A1								
A2								
A3								

Tentativa 1

- Escolher primeiro as atividades começam primeiro.

Ativ	0	1	2	3	4	5	6	7
A1								
A2								
A3								

- Não foi muito bom: escolheu apenas a atividade A1, quando poderia ter escolhido duas atividades: A2 e A3.

Tentativa 2

- Escolher primeiro as atividades que demoram menos tempo

Tentativa 2

- Escolher primeiro as atividades que demoram menos tempo

Ativ	0	1	2	3	4	5	6	7
A1								
A2								
A3								

Tentativa 2

- Escolher primeiro as atividades que demoram menos tempo

Ativ	0	1	2	3	4	5	6	7
A1								
A2								
A3								

- Não foi muito bom: escolheu apenas a atividade A2, quando poderia ter escolhido duas atividades: A1 e A3.

Tentativa 3

- Escolher primeiro as atividades terminam primeiro.

Atividades:

Ativ	0	1	2	3	4	5	6	7	8	9	10	11	12	13
A1														
A2														
A3														
A4														
A5														
A6														
A7														
A8														
A9														
A10														
A11														

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).

```
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;
   }
  ...
```

```
// 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.");
}
```

Resultado:

Ativ	0	1	2	3	4	5	6	7	8	9	10	11	12	13
A1														
A2														
A3														
A4														
A5														
A6														
A7														
A8														
A9														
A10														
A11														

Está solução é ótima: seleção de 4 atividade.

Pode haver outras soluções com 4 atividades, mas nenhuma com 5.

- Para o problema da selção de atividades o algoritmo guloso é ótimo.

- Tipicamente, um dos "segredos" dos algoritmos gulosos é a escolha de como o conjunto de entrada será ordenado.

- Vamos estudar agora o problema da Mochila.

- Problemas da Mochila:
- Dados:
  - um mochila que admite um certo peso;
  - um conjunto de objetos, cada um com um valor e um peso;
- Selecione o conjunto de objetos que caibam dentro da mochila de forma a maximizar o valor total dentro da mochila.

- Este problema se divide em dois subproblemas distintos:
- 1. 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;
- 2. Os objetos não podem ser particionados (ou estarão dentro da mochila ou fora). Problema conhecido como “Problema da Mochila Binária ou 0-1”

- Qual seria a melhor ordenação da entrada para o problema da mochila (que permite fracionamento dos objetos)?

- Qual seria a melhor ordenação da entrada para o *problema da mochila 0-1*?

- Qual seria a melhor ordenação da entrada para o problema da mochila (que permite fracionamento dos objetos)?
  - **ordenar pelo valor/peso**

- Qual seria a melhor ordenação da entrada para o *problema da mochila 0-1*?
  - **ordenar pelo valor/peso**

- A solução gulosa será ótima?

- A solução gulosa será ótima?

- **Para o problema geral (que admite fracionamento), SIM.**

Para o problema da mochila 0-1, NÃO.

- Por exemplo, dada uma mochila que admite 10 kg e três objetos com os seguintes valores-pesos:

  O1: \$12 6 kg => 2 (valor/peso)

  O2: \$ 8 5kg => 1,6

  O3: \$ 7 5kg => 1,4

Para o problema da mochila 0-1, NÃO.

- Por exemplo, dada uma mochila que admite 10 kg e três objetos com os seguintes valores-pesos:

  O1: \$12 6 kg => 2 (valor/peso)

  O2: \$ 8 5kg => 1,6

  O3: \$ 7 5kg => 1,4

Solução gulosa: {O1}, valor total \$ 12.

Solução ótima: {O2,O3}, valor total \$ 15.