Centro Federal de Educação Tecnológica de Minas Gerais
CEFET-MG

Laboratório de Algoritmos e Estruturas de Dados II

Trabalho Prático II:
Linhas de Montagem

Data de entrega: 10/06/2019

Illyana Guimarães de Avelar
Luan Ferreira de Almeida

2019

Código Fonte Programação Gulosa:

```
package TrabalhoPratico02;

import java.io.*;
import java.util.*;

public class Guloso {
  private ArrayList<Integer> A1 = new ArrayList();
  private ArrayList<Integer> A2 = new ArrayList();
  private ArrayList<Integer> T1 = new ArrayList();
  private ArrayList<Integer> T2 = new ArrayList();
  private ArrayList<String> Path = new ArrayList();
  private int lv = 0;

  public Guloso(ArrayList A1, ArrayList A2, ArrayList T1, ArrayList T2) throws
IOException{

    this.A1 = A1;
    this.A2 = A2;
    this.T1 = T1;
    this.T2 = T2;
  }

  public void caminho(){
    //enquanto houverem elementos antes do final da linha
    for(int i=0; i < A1.size()-2; i++){
      if(i==0){//1a
        //se a linha um formar um caminho menor que a linha 2, o caminho formado por ela é
adicionado ao caminho final
        if((A1.get(0) + A1.get(1)) <= (A2.get(0) + A2.get(1))){
          Path.add(Integer.toString(A1.get(0)));
          Path.add(Integer.toString(A1.get(1)));
          lv = 1;
        }
        else{
          //senão o caminho formado pela linha 2 é escolhido
          lv = -1;
          Path.add(Integer.toString(A2.get(0)));
          Path.add(Integer.toString(A2.get(1)));
        }
      }
        else//2a
          //se estiver no meio da linha
          if(i > 0 && i < A1.size()-3){
```

```
 if(lv==1){
  //se passar pela primeira linha for mais rápido que descer para a segunda
  if(A1.get(i+1) <= (T1.get(i-1) + A2.get(i+1))){
   //o caminho continua por ela
   Path.add(Integer.toString(A1.get(i+1)));
  }
  else{
   //senão desce para a linha de baixo
   lv = -1;
   //para identificar quando desce para a linha de baixo o array terá um \n
   Path.add("\n" + Integer.toString(T1.get(i-1)));
   //adiciona a linha de baixo ao caminho final
   Path.add(Integer.toString(A2.get(i+1)));
  }

 }
 else{//3a
  //se continuar na segunda linha gastar menos tempo
  if(A2.get(i+1) <= (T2.get(i-1) + A1.get(i+1))){
   //adicionamos o próximo da segunda linha ao caminho final
   Path.add(Integer.toString(A2.get(i+1)));
  }
  else{
   //senão subimos para a linha 1
   lv = 1;
   //e adicionamos o intermediário mais o próximo da linha 1 ao caminho final
   Path.add("\n" + Integer.toString(T2.get(i-1)));
   Path.add(Integer.toString(A1.get(i+1)));
  }
 }
}
//no final do caminho
else{
 //se estiver na linha 1
 if(lv==1){
  //se continuar na linha 1 for menos custoso
  if((A1.get(i+1) + A1.get(i+2)) <= (T1.get(i-1) + A2.get(i+1) + A2.get(i+2))){
   //adicionamos o final referente a continuar na linha 1
   Path.add(Integer.toString(A1.get(i+1)));
   Path.add(Integer.toString(A1.get(i+2)));
  }
  else{
   //senão descemos para a linha 2
   lv = -1;
   Path.add("\n" + Integer.toString(T1.get(i-1)));
```

```
            Path.add(Integer.toString(A2.get(i+1)));
            Path.add(Integer.toString(A2.get(i+2)));
           }
          }
          //se estiver na linha 2
          else{
           //se continuar na linha 2 for menos custoso acabamos nela
           if((A2.get(i+1) + A2.get(i+2)) <= (T2.get(i-1) + A1.get(i+1) + A1.get(i+2))){
            Path.add(Integer.toString(A2.get(i+1)));
            Path.add(Integer.toString(A2.get(i+2)));
           }
           //senão acabamos na linha 1
           else{
            lv = 1;
            Path.add("\n" + Integer.toString(T2.get(i-1)));
            Path.add(Integer.toString(A1.get(i+1)));
            Path.add(Integer.toString(A1.get(i+2)));
           }
          }
         }
    }
    //Imprime todo o caminho percorrido
    System.out.print("Path: ");
    Path.forEach((str) -> {
        System.out.print(str + "  ");
    });
    System.out.println("\n");
   }
}
```

Código Fonte Programação Dinâmica:

```
package TrabalhoPratico02;
import java.io.*;
import java.util.*;

public class Dinamico {

  int t = 99;
  private ArrayList<Integer> A1 = new ArrayList();
  private ArrayList<Integer> A2 = new ArrayList();
  private ArrayList<Integer> T1 = new ArrayList();
  private ArrayList<Integer> T2 = new ArrayList();
  private ArrayList<String> Path = new ArrayList();
  private ArrayList<String> CPath = new ArrayList();

  public Dinamico(ArrayList A1, ArrayList A2, ArrayList T1, ArrayList T2) throws
IOException {

    this.A1 = A1;
    this.A2 = A2;
    this.T1 = T1;
    this.T2 = T2;
  }

  public void caminho(int lv, int index, int peso){

    //se estiver na linha 1
    if(lv == 1){
      //se estiver no final da linha
      if(index == A1.size()-1){
        //se a soma atual for menor do que a total até o momento, a total anterior é substituída
        //e um novo caminho é criado com o formado pelo menor peso
        if(peso < t){
          t = peso;
          CPath = new ArrayList(Path);
        }
      }
      //Linha de montagem
      else{
        if(index==0){
          //se estiver no inicio adiciona o primeiro peso
          peso = A1.get(index);
          Path.add(Integer.toString(peso));
        }
        //adiciona o próximo item na linha 1 e realiza a chamada recursiva
```

```
    Path.add(Integer.toString(A1.get(index+1)));
    caminho(lv,index+1,peso+A1.get(index+1));
    try{
    //retira o último elemento
    Path.remove(Path.size()-1);
    }catch(IndexOutOfBoundsException e){
      System.out.println("Erro de posição inválida");
      e.getMessage();
    }
    //se estiver no meio
    if(index < A1.size()-2 && index > 0){
      //vai para a linha 2 e adiciona seu caminho ao caminho total
      Path.add("\n" + Integer.toString(T1.get(index-1)));
      Path.add(Integer.toString(A2.get(index+1)));
      //realiza chamada recursiva
      caminho(-1,index+1,peso+T1.get(index-1)+A2.get(index+1));
        try{
        //retira o último e penúltimo elementos
        Path.remove(Path.size()-1);
        Path.remove(Path.size()-1);
        }catch(IndexOutOfBoundsException e){
          System.out.println("Erro de posição inválida");
          e.getMessage();
        }
    }
  }
}
  //se estiver na linha 2
  else{
    //se estiver no final da linha
    if(index==A2.size()-1){
      //se a soma atual for menor do que a total até o momento, a total anterior é substituída
      //e um novo caminho é criado com o formado pelo menor peso
      if(peso<t){
        t=peso;
        CPath = new ArrayList(Path);
      }
    }
    //se estiver no meio, realiza passos análogos aos da linha 1
    else{
      if(index==0){
        peso = A2.get(index);
        Path.add(Integer.toString(peso));
      }
```

```
        Path.add(Integer.toString(A2.get(index+1)));
        caminho(lv,index+1,peso+A2.get(index+1));
        try{
        Path.remove(Path.size()-1);
        }catch(IndexOutOfBoundsException e){
          System.out.println("Erro de posição inválida");
          e.getMessage();
        }

        if(index<A2.size()-2 && index>0){
          Path.add("\n" + Integer.toString(T2.get(index-1)));
          Path.add(Integer.toString(A1.get(index+1)));
          caminho(1,index+1,peso+T2.get(index-1)+A1.get(index+1));
          try{
          Path.remove(Path.size()-1);
          }catch(IndexOutOfBoundsException e){
            System.out.println("Erro de posição inválida");
            e.getMessage();
          }
        }
      }
    }
   //imprime todo o caminho percorrido
   System.out.print("Path -> ");
   CPath.forEach((item) -> {
      System.out.print(item + ", ");
   });
   System.out.println("\n");
  }
}
```

Primeira Instância:

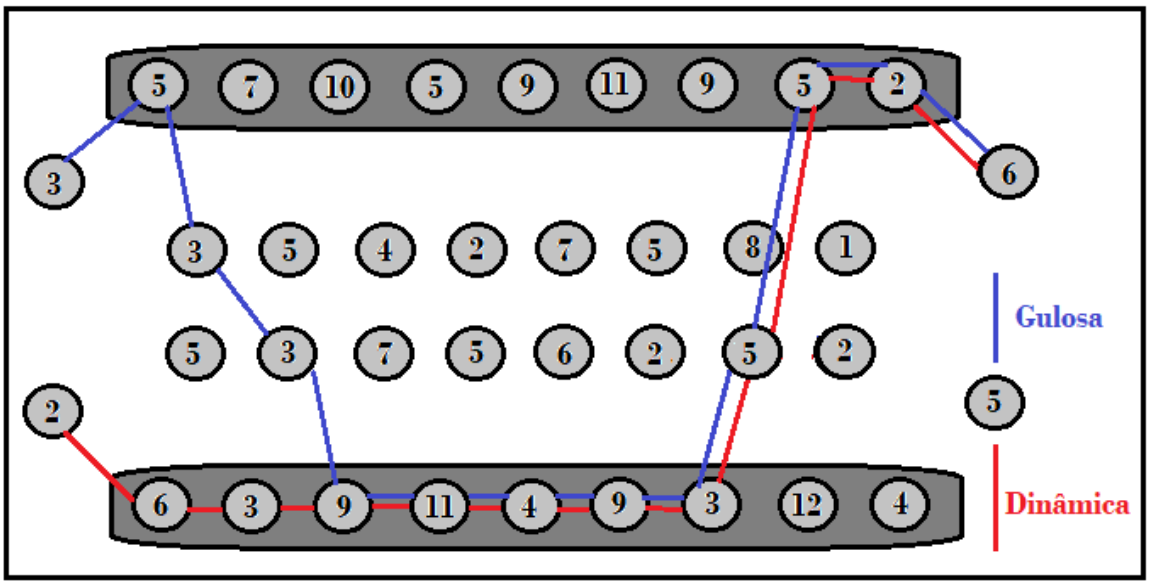
```
Output - TrabalhoPratico02 (run)
run:
Insira a montagem:3,5,7,10,5,9,11,9,5,2,6
2,6,3,9,11,4,9,3,12,4,5
3,5,4,2,7,5,8,1
5,3,7,5,6,2,5,2
A1 = [3, 5, 7, 10, 5, 9, 11, 9, 5, 2, 6]
A2 = [2, 6, 3, 9, 11, 4, 9, 3, 12, 4, 5]
T1 = [3, 5, 4, 2, 7, 5, 8, 1]
T1 = [5, 3, 7, 5, 6, 2, 5, 2]

DINAMICO:
Tempo: 0 milissegundos
Path: 2 -> 6 -> 3 -> 9 -> 11 -> 4 -> 9 -> 3 ->
5 -> 5 -> 2 -> 6

GULOSO:
Path: 3 -> 5 ->
3 -> 3 -> 9 -> 11 -> 4 -> 9 -> 3 ->
5 -> 5 -> 2 -> 6

Tempo: 0 milissegundos

BUILD SUCCESSFUL (total time: 4 seconds)
```



Segunda Instância:

```
Output - TrabalhoPratico02 (run)
run:
Insira a montagem:5,10,6,3,8,5,3,7,12,8
7,3,5,3,7,6,4,9,10,9
4,2,7,2,5,8,2
6,1,7,3,6,4,5

A1 = [5, 10, 6, 3, 8, 5, 3, 7, 12, 8]
A2 = [7, 3, 5, 3, 7, 6, 4, 9, 10, 9]
T1 = [4, 2, 7, 2, 5, 8, 2]
T1 = [6, 1, 7, 3, 6, 4, 5]

DINAMICO:
Tempo: 0 milissegundos
Path: 7 -> 3 -> 5 ->
1 -> 3 -> 8 -> 5 -> 3 -> 7 -> 12 -> 8

GULOSO:
Path: 7 -> 3 -> 5 -> 3 -> 7 -> 6 -> 4 -> 9 -> 10 -> 9

Tempo: 0 milissegundos

BUILD SUCCESSFUL (total time: 2 minutes 12 seconds)
```