Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
AD2 de Programação III
2° semestre de 2013

Nome:
Pólo:
Exercício:

Considere que você é dono de uma empresa de software e que sua empresa foi contratada por uma empresa de transportadora para resolver o problema de encaixotamento de objetos em mudanças.

O problema de encaixotamento de objetos em mudanças consiste em, dados caixas que suportam um peso fixo `P` e um conjunto de objetos quaisquer com pesos, menores ou iguais a `P`, associados a eles, determinar o menor número de caixas a serem usadas.

Uma das estratégias aproximadas (isto é, implementações que não garantem a melhor solução para o problema) é ordenar os objetos de maneira crescente, montar a primeira caixa e inserir os menores objetos na mesma, até chegar ao limite `P`. Quando a primeira caixa estiver cheia, seu programa deve montar a segunda caixa e inserir os menores objetos, que ainda não foram encaixotados. Seu programa termina quando todos os objetos estiverem contidos em caixas.

Seu programa deve receber, como parâmetro de entrada, o nome do arquivo com os objetos e seus respectivos pesos e o limite `P` que uma caixa suporta e deve retornar todas as caixas, com seus respectivos objetos.

Por exemplo **(este programa tem que funcionar para QUAISQUER arquivos no mesmo formato)**, dado o arquivo `objetos.txt`, composto de:

```
A     4
O     6
I     2
M     8
U     4
V     2
E     4
N     3
S     1
W     7
```

e `P = 10`, uma execução deste programa (`java caixas objetos.txt 10`) deve responder:

```
TOTAL: 5
C1 (peso = 8): S I V N
C2 (peso = 8): A U
C3 (peso = 10): E O
C4 (peso = 7): W
C5 (peso = 8): M
```

RESPOSTA:

```
// A solução apresentada será dividida nos seguintes passos:
// inicialmente, os objetos serão ordenados em uma lista crescente.
// Logo a seguir, o primeiro objeto será colocado na primeira caixa.
// O segundo objeto será colocado na primeira caixa, se for possível.
// Senão, uma nova caixa será aberta e esse objeto será guardado em C2
// (e, assim sucessivamente).

import java.io.*;

// Classe que representa um Objeto. Ele possui uma referência para o
// próximo objeto.
class Objeto{
  String nome;
  int peso;
  Objeto prox = null;

  Objeto(String n, int p){
    nome = n;
    peso = p;
  }

  int retornaPeso(){ return peso; }

  public String toString(){ return nome + " "; }
}

// Classe que implementa a lista de objetos ordenados de maneira
// crescente em relação ao peso do objeto.
class Lista_Objetos{
  Objeto prim;

  Lista_Objetos(){ prim = null; }

  Objeto retornaPrim(){ return prim; }

  // inserção ordenada de Objetos
  void insereObjeto(Objeto obj){
    Objeto novo = new Objeto(obj.nome, obj.peso);
    if(prim == null){
      prim = novo;
      return;
    }
    Objeto p = prim, ant = null;
    while((p != null) && (p.peso <= obj.retornaPeso())){
      ant = p;
      p = p.prox;
    }
    if(ant == null){
      novo.prox = prim;
      prim = novo;
    }
    else{
      novo.prox = p;
      ant.prox = novo;
    }}
```

```
  public String toString(){
    String resp = "";
    Objeto p = prim;
    while(p != null){
      resp += p.toString();
      p = p.prox;
    }
    return resp + "\n";
  }
}

// Classe que representa somente uma Caixa. Ela tem uma referencia para
// a próxima caixa
class Caixa{
  static int prox_id = 1;
  String nome;
  int limite_peso, peso_total;
  Lista_Objetos l;
  Caixa prox = null;

  Caixa(int p){
    nome = "C" + prox_id;
    prox_id++;
    peso_total = p;
    limite_peso = 0;
    l = new Lista_Objetos();
  }

  int retornaPeso(){ return limite_peso; }

  // Inserção do objeto na caixa, se existe espaço. Senão, retorne false
  boolean insereObjetoCaixa(Objeto obj){
    if((obj.retornaPeso() + limite_peso) <= peso_total){
      l.insereObjeto(obj);
      limite_peso += obj.retornaPeso();
      return true;
    }
    return false;
  }

  public String toString(){
    String resp = nome + "  (peso = " + limite_peso + "): ";
    return resp + l.toString();
  }
}

// Classe que representa a lista de caixas.
class Lista_Caixas{
  Caixa prim;
  int total;

  Lista_Caixas(){
    prim = null;
    total = 0;
  }
```

```
  // A inserção de uma nova caixa no fim da lista.
  void insereCaixa(Caixa cx){
    total++;
    if(prim == null){
      prim = cx;
      return;
    }
    Caixa p = prim;
    while(p.prox != null) p = p.prox;
    p.prox = cx;
  }

  public String toString(){
    String resp = "TOTAL = " + total + "\n";
    Caixa p = prim;
    while(p != null){
      resp += p.toString();
      p = p.prox;
    }
    return resp;
  }
}

public class Caixas{
  public static void main(String[] args) throws Exception{
    // Abertura do arquivo de dados e definição da capacidade total de
    // todas as caixas que serão criadas
    BufferedReader in = new BufferedReader(new FileReader(args[0]));
    int tam_caixa = Integer.parseInt(args[1]);
    try {
      // Criação da lista de objetos
      Lista_Objetos lo = new Lista_Objetos();
      String s, vs[];
      while((s = in.readLine()) != null){
        vs = s.split("\t");
        // Inserção ordenada dos objetos
        lo.insereObjeto(new Objeto(vs[0], Integer.parseInt(vs[1])));
      }
      in.close();
      // Criação da lista de caixas. Inicialmente a lista está vazia.
      Lista_Caixas lc = new Lista_Caixas();
      // Leitura do primeiro objeto da lista ordenada de objetos
      Objeto obj = lo.retornaPrim();
      while(obj != null){
        // cx indica a primeira caixa
        Caixa cx = lc.prim;
        while(cx != null){
          // se o objeto foi inserido em cx, saia do loop.
          // senão, tente a próxima caixa.
          if(cx.insereObjetoCaixa(obj)) break;
          cx = cx.prox;
        }
```

```
        // se o algoritmo percorreu todas as caixas e o objeto não foi
        // inserido, criar a nova caixa e inserir o objeto nela.
        if(cx == null){
          cx = new Caixa(tam_caixa);
          cx.insereObjetoCaixa(obj);
          lc.insereCaixa(cx);
        }
        // passar para o próximo objeto a ser inserido nas caixas.
        obj = obj.prox;
      }
      System.out.println(lc);
    }
    catch (Exception e){
      System.out.println("Excecao\n");
    }
  }
}
```