Problema da mochila com abordagem gulosa

Aluno: Raylison Nunes dos Santos

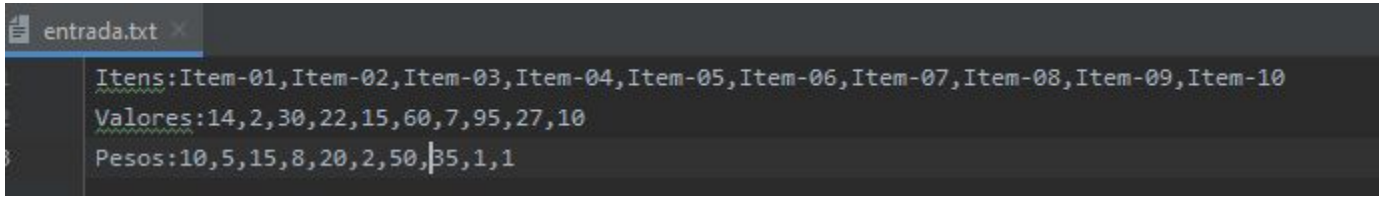
Problema da mochila

O problema da mochila (em inglês, *Knapsack problem*) é um problema de optimização combinatória. O nome dá-se devido ao modelo de uma situação em que é necessário preencher uma mochila com objetos de diferentes pesos e valores. O objetivo é que se preencha a mochila com o maior valor possível, não ultrapassando o peso máximo.

Abordagem Gulosa

um algoritmo guloso escolhe, em cada iteração, o objeto mais apetitoso que vê pela frente. O objeto escolhido passa a fazer parte da solução que o algoritmo constrói. Um algoritmo guloso é míope: ele toma decisões com base nas informações disponíveis na iteração corrente, sem olhar às consequências que essas decisões terão no futuro. Um algoritmo guloso jamais se arrepende ou volta atrás: as escolhas que faz em cada iteração são definitivas.

Algoritmo - entrada



Algoritmo - código

```java
public class Mochila {

    public static void main(String[] args) {

        Scanner imput = new Scanner(System.in);
        System.out.println("Informe a capacidade da mochila: ");

        int capacidade = imput.nextInt();   //Capacidade da mochila.
        int enchendo = 0;   // Variavel auxiliar para verificar o volume atual da mochila.

        ArrayList<String> listItens = null;  // Lista de itens carregada do arquivo de entrada
        ArrayList <String> listValores = null;  // Lista de itens carregada do arquivo de entrada.
        ArrayList<String> listPesos = null;  //Lista de pesos carregada do arquivo de entrada

        //Map de representação da mochila
        TreeMap<String,  TreeMap<Integer, Integer>> mochila = new TreeMap<String,  TreeMap<Integer, Integer>>();

        //Map de representação dos itens fora da mochila
        TreeMap<String, TreeMap<Integer, Integer>> foraDaMochila = new TreeMap<String, TreeMap<Integer, Integer>>();

        int totalFora = 0; //Variavel auxiliar de contagem
        int totalDentro = 0;  //Variavel auxiliar de contagem
```

Algoritmo - código

```
//Inicio da leitura do arquivo de entrada
FileReader file;
try {
    file = new FileReader( fileName: "entrada.txt");
    BufferedReader readFile = new BufferedReader(file);
    String line = readFile.readLine();
    int linha = 1;

    while (line != null) {
        if (linha == 1) {
            String[] itens = line.split( regex: ":");
            listItens = new ArrayList(Arrays.asList(itens[1].split( regex: ",")));
        }
        else if (linha == 2) {
            String[] itens = line.split( regex: ":");
            listValores = new ArrayList(Arrays.asList(itens[1].split( regex: ",")));
        }
        else if (linha == 3) {
            String[] pesos = line.split( regex: ":");
            listPesos = new ArrayList(Arrays.asList(pesos[1].split( regex: ",")));

        }
        linha++;
        line = readFile.readLine();
    }

    readFile.close();

} catch (IOException e) {
    e.printStackTrace();
}

//Fim da leitura do arquivo de entrada
```

Algoritmo - código

```
int valorGuloso = 0;    //Auxiliar valor guloso na lista
int index = 0;      //index valor guloso

//Laço para inserção de itens na mochila
while (true) {

    //Laço de validação de item da lista de entrada
    for (int item = 0; item < listValores.size(); ) {

        //Se o valor guloso for apto ele é atualizado e fica disponivel pra entrar na mochila
        if (valorGuloso <= Integer.parseInt(listValores.get(item)) && (enchendo + Integer.parseInt(listPesos.get(item))) <= capacidade) {
            valorGuloso = Integer.parseInt(listValores.get(item));
            index = item;
        }

        //Se o valor não for apto ele é removido da lista e adiconado ao map dos excluidos
        if ((Integer.parseInt(listPesos.get(item)) + enchendo) > capacidade) {

            TreeMap<Integer, Integer> valorPesoFora = new TreeMap<>();
            valorPesoFora.put(Integer.parseInt(listValores.get(item)),Integer.parseInt(listPesos.get(item)));
            foraDaMochila.put(listItens.get(item),valorPesoFora);
            totalFora += Integer.parseInt(listValores.get(item));
            listItens.remove(item);
            listValores.remove(item);
            listPesos.remove(item);

        } else {
            item++;
        }
    }
```

Algoritmo - código

```
    if (!listValores.isEmpty()) {
        //insere o valor guloso na mochila
        if ((enchendo + Integer.parseInt(listPesos.get(index))) <= capacidade) {

            TreeMap<Integer, Integer> valorPesoDentro = new TreeMap<>();
            valorPesoDentro.put(valorGuloso,Integer.parseInt(listPesos.get(index)));
            mochila.put(listItens.get(index), valorPesoDentro);
            totalDentro += valorGuloso;
            enchendo += Integer.parseInt(listPesos.get(index));
            listItens.remove(index);
            listValores.remove(index);
            listPesos.remove(index);

        } else {
            break;
        }
    } else {
        break;
    }
    valorGuloso = 0;
    index = 0;
}
```

Algoritmo - saída

```
*****************************************************************************************

Capacidade da mochila: 34
-----------------------------------------------------------------------------------------
Itens inclusos na mochila: {Item-02={2=5}, Item-03={30=15}, Item-04={22=8}, Item-06={60=2}, Item-09={27=1}, Item-10={10=1}}
Total de itens na mochila: 6
Valor total da mochila: 151
-----------------------------------------------------------------------------------------
Itens de fora da mochila: {Item-01={14=10}, Item-05={15=20}, Item-07={7=50}, Item-08={95=35}}
Total de itens fora da mochila: 4
Valor total fora da  mochila: 131

*****************************************************************************************

Process finished with exit code 0
```

OBRIGADO