UNIVERSIDADE FEDERAL DE SÃO CARLOS

CAMPUS SOROCABA

ALGORITMOS E COMPLEXIDADE

SOLUÇÕES PARA OS PROBLEMAS 1286, 1310 E 1524

Gabriel Stankevix Soares | 511340

novembro de 2015

SUMÁRIO

1. INTRODUÇÃO

1.1. PROBLEMA 1286 - MOTOBOY

1.2. PROBLEMA 1310 - LUCRO

1.3. PROBLEMA 1524 - FILA DO BANDEJÃO

2. CONCLUSÃO E DIFICULDADES

## **1. INTROUÇÃO**

Neste trabalho desenvolvido para a matéria de Algoritmos e Complexidade, tem como objetivos encontrar soluções para os três exercícios propostos(1286 - MOTOBOY, 1310 - LUCRO, 1524 - FILA DO BANDEJÃO). A ideia é desenvolver as soluções conforme os algoritmos explicados e ensinados pelo professor Mario Lizier em sala de aula, buscando assim a melhor eficiência para cada solução.

## 

## 

## **1.1. PROBLEMA 1286 - MOTOBOY**

O problema consiste em maximizar a entrega de pedidos de pizza conforme a quantidade e o tempo atrelado a cada pedido realizado na pizzaria. Para este problema adotei a logica do problema da mochila ou Knapsack problem em inglês, que consiste na solução de problemas que envolvam otimização combinatória.

Sendo assim utilizei como base o pseudocódigo da solução do problema da mochila fornecido em aula para implementar a minha solução para este problema.

No algoritmo foi adaptado o pseudocódigo em uma função recursiva que realiza as n operações combinatórias, segue a função abaixo :

int entrega(int i, int j){

if(F[i][j] < 0){

if( j < quantidade[i-1]){

valor = entrega(i-1, j);

}else{

valor = max(entrega(i-1,j),

tempo[i-1] + entrega(i-1, j - quantidade[i-1]));

}

F[i][j] = valor;

}

return F[i][j];

}

## 

## 

## **1.2. PROBLEMA 1310 - LUCRO**

Este problema consiste em calcular o melhor e maior lucro, conforme o custo e lucro diário de um circo durante uma semana de exibições. Notamos que este problema consiste em encontrar uma sublista contígua de maior soma a partir de uma lista de números, um problema bem conhecido, logo com diversas soluções que vão de cubicas ate linear.

Preferi adotar a solução quadrática que consiste em dois laços e algumas operações de aritméticas. A logica desenvolvida é que, com os dois laços é realizada as n operações somando os lucros diários( que é a receita) mais a operação de subtração de custo diário ( receitaAtual - ( quantidadeDeDias \* custoPorDia). Conforme o algoritmo acha uma receita máxima (de maior), o valor é armazenado na variável chamada *maior* ( que contem o maior lucro atual), o valor desta variável pode alterar conforme as operações aritméticas realizadas nos laços forem maior que o valor atual desta variável ( *if(receitaAtual > maior)*). Segue o trecho do código para melhor entendimento:

for(i = 0; i < N; i++){

somaAtual = receita[i];

dias = 1;

receitaAtual = somaAtual - ( dias \* custoPorDia);

if(receitaAtual > maior){

maior = receitaAtual;

}

for( j = i + 1; j < N ; j++){

somaAtual = somaAtual + receita[j];

dias++;

receitaAtual = somaAtual - ( dias \* custoPorDia);

if(receitaAtual > maior){

maior = receitaAtual;

}

## 

## **1.3. PROBLEMA 1524 - FILA DO BANDEJÃO**

Problema que consiste em dado um conjunto de valores subdivididos em k grupos, é necessário encontrar a uma soma das distancias dos elementos com relação ao primeiro emento ( de cada grupo) em cada subgrupos respectivamente. Obviamente, o problema foi adaptado para q questão da fila do bandejão.

A solução proposta para esse problema consiste em criar uma fila de tamanho N que contem os N elementos dos K grupos existentes na fila. O vetor distancia armazena as distancia entre os N elementos de cada K grupo. Sendo assim, passamos para a função bandejão, nela foi implementada a logica para solucionar este problema.

O primeiro laço varre o vetor distancia conforme o numero de grupos existentes. As variáveis *maior* e *flagMaior* servem para identificar o primeiro elemento de cada subgrupo, que deve ser zero conforme o enunciado. Logo, o segundo laço serve para varrer os N elementos da fila, achar o maior salvar a sua posição e por fim zerar sua distancia.

Ao final desta operações o vetor distancia conterá exatamente as distancias corretas sobrando apenas a operação de soma deste valores para retornar o valor correto. Segue o código para melhor entendimento:

i = 1;

while(i < K){

maior = 0; flagDist = 0;

for(j=1; j < N; j++){

if(distancia[j] > maior){

maior = distancia[j];

flagDist = j;

}

}

distancia[flagDist] = 0;

i++;

}

## 

## **2. CONCLUSÃO E DIFICULDADES**

A realização deste algoritmos provou a importância dos algoritmos estudados em sala de aula, para encontrar soluções na tentativa de resolver problemas propostos pelo professor.

O Problema do Lucro foi o de mais fácil entendimento e desenvolvimento muito por conta de ser um problema já difundido e de implementação rápida.

O problema da mochila pode ser considerado de media dificuldade, por conta da logica que solução envolve, tive um pouco de dificuldade para entender esta logica porem apos compreender seu funcionamento a solução para o problema fluiu, e a implementação não foi dificultosa.

Encontrar a solução para o problema do bandejão foi a de maior dificuldade muito por conta a logica que a envolve, porem apos entender como funcionava as relações de distancia entre os n elementos de cada k grupo contidos na fila, a implementação fluiu chegando a na correta solução do problema.

Por fim, resolver estes problemas fixaram melhor o funcionamento dos algoritmos ensinados, melhorando a pratica de programação e entendimento sobre algoritmos e suas complexidades.