Universidade Federal de São Paulo - Campus São José dos Campos
Disciplina: Projeto e Análise de Algoritmos

Problema 3: Café lucrativo

Rosângela Miyeko Shigenari, 92334

4 de junho de 2018

1 Introdução

Este trabalho visa a resolução de um problema do café lucrativo com a aplicação dos conceitos de Programação dinâmica, vistos em aula, em linguagem C/C++, com um tempo limite de 0,5s, e em seguida, realizar o estudo de sua complexidade.

2 Problema

O Brasil é o maior produtor e exportador de café do mundo. Com o crescimento da demanda por cafés diferenciados, muitos produtores têm visto grande potencial de renda na produção de cafés especiais. O produtor de café Sebastião produz grãos de café especial que possui qualidade comprovada por certificações e reconhecida por diferentes compradores regulares que estão sempre interessados em comprar a quantidade de lotes de café que Sebastião puder lhes vender. O problema é que cada comprador determina o valor a ser pago e o tamanho de cada lote que ele pode comprar, sendo que o tamanho de um lote definido pelo comprador é dado em número específico de sacas. Satisfeitas essas condições de valor e tamanho do lote, cada comprador pode comprar a quantidade de lotes que Sebastião desejar vender. Dados os valores que os possíveis compradores pagam por lote e a definição de tamanho do lote, você deve escrever um programa que calcule o valor da receita máxima que Sebastião pode obter com a venda da produção de café deste ano para esses compradores.

Entrada: A primeira linha de um caso de teste contém os inteiros P ($1 \leq P \leq 100$), indicando o número de sacas de café especial produzidas neste ano por Sebastião, e N, indicando o número de possíveis compradores ($1 \leq N \leq 100$). Em cada uma das próximas N linhas, são apresentados o tamanho do lote em número de sacas ti ($1 \leq ti \leq 100$) e o valor a ser pago pelo lote vi ($1 \leq vi \leq 1000$) por um dado comprador i.

Saída:
Você deve imprimir o valor máximo de receita que é possível obter com a produção deste ano de café do produtor Sebastião.

Exemplos de entrada e saída

- **Entrada 1**:
  30 4
  25 100
  10 30
  3 6
  11

Saída 1:
108

- **Entrada 2**:
40 5
10 10
15 30
3 6
7 22
25 75
Saída 2:
119

3 Solução

A solução é baseada em uma variação de um problema clássico de programação: o problema da mochila. A alteração consiste no seguinte: ao invés de selecionar apenas um item de cada objeto, é possível haver repetições. Assim, aplicamos ao nosso problema como se cada entrada ti fosse o peso de um objeto e P o peso máximo da mochila.

3.1 Implementação da Solução

Utilizando os conceitos de programção dinãmica vistos em aula, foi realizada a implementação da solução. A solução é armazenada no vetor chamado *maxVet*, como apresentado no trecho de código da Figura 1.

```
int ProblemSolution(int W, int n, int *v, int *t)
{
    int maxVet[W+1]; //armazenas os maximos
    int i, j;
    for(i = 0; i<W+1; i++){//inicializa o vetor
      maxVet[i] = 0;
    }
//variacao do problema da mochila
//busca do maior lucro
    for (i=0; i<W+1; i++)
      for (j=0; j<n; j++)
         if (t[j] <= i)
            maxVet[i] = ReturnMax(maxVet[i], maxVet[i - t[j]] + v[j]);// atualiza o maximo

    return maxVet[W];
}
```

Figura 1: Função Solução

Primeiramente, o vetor é inicializado com 0, que é o valor mínimo possível. No primeiro laço, atualizamos o valor de *maxVet*[i], que memoriza o máximo de receita testando todas as combinações possíveis com os "pesos"disponíveis, considerando uma capacidade i. Esse procedimento realizado com os N itens disponíveis (laço interno). Ao final, o lucro máximo obtido com a venda das P sacas disponíveis é retornada.

3.2 Função Principal

A função principal do algoritmo é apresentada na Figura 2. Inserimos os valor de P (máximo de sacas que podem ser vendidas), N (número de compradores possíveis), sendo alocados 2 vetores de tamanho N, que representa para cada vendedor, o número de sacas que deseja comprar e seu valor correspondente. Em seguida, esses valores são aplicados na função solução.

```c
int main()
{
    int P, N, i;
    int result;

    scanf("%d", &P);
    scanf("%d", &N);

    int *t = (int *) malloc(N * sizeof(int));
    int *v = (int *) malloc(N * sizeof(int));

    for(i = 0; i<N; i++){
      scanf("%d", &t[i] );
      scanf("%d", &v[i] );
    }
    result = ProblemSolution(P, N, v, t);
    printf("%d\n", result);


    return 0;
}
```

Figura 2: Função Principal

4 Análise do Algoritmo

O algoritmo implementado possui 2 laços de repetição. O laço mais externo itera P vezes, e para cada uma dessas iterações, é realizada N iterações, logo temos um custo NP.