UNIVERSIDADE FEDERAL DE SÃO CARLC
Centro de Ciências Exatas e de Tecnologia
Departamento de Computação
Bacharelado em Ciência da Computação
Estudante: Gabriela Vanessa Pereira Alves Mattos. RA: 551570

Problema C – Coffee Break

Implementado por *greedy*.

Para implementação do problema do Coffee Break foi utilizado como base o problema da Mochila, onde é encontrado um valor referente ao custo pela porção, que permitirá saber qual a mais cara em relação ao volume da porção.

Assim que os valores vão sendo digitados já é calculado o *cv* (custo da porção por volume da porção) e armazenado na mesma estrutura do custo e do volume da porção, essa estrutura foi nomeada por porção e foi necessário um vetor chamado *coffee* que permitirá que seja informada diversas porções.

O programa consiste em ordenar os dados das porções informadas a partir do *cv* e então os alunos irem comendo o volume possível das maiores *cv's*. Observe o algoritmo abaixo:

```
1 void coffeeGreedy(Porcao *coffee, double *e, int n, int a)
2 {
3
4         int i,k;
5         double soma;
6         double consumo;
7
8         sort(coffee, coffee+n, compare); //n*log(n)
9
10        for(i=0; i<a; i++) //varia de 0 a quantidade de alunos (a)
11        {
12
13                soma = 0;
14                k = n-1;
15
16                while (e[i] != 0 && k>=0) //varia de quantidade de consumo do aluno (e[i])
17                {
18
19                        if (e[i] >= coffee[k].v)
20                                consumo = coffee[k].v;
21                        else
22                                consumo = e[i];
23
24                        e[i]=e[i]-consumo;
25                        soma=soma+(consumo*coffee[k].cv);
26                        k--;
27                }
28                cout << setiosflags (ios::fixed) << setprecision (4) <<  soma <<endl;
29        }
30 }
```

Para ordenar utilizou-se a função *sort()* da biblioteca padrão do C++, só sendo necessário definir qual é a condição para essa ordenação. Nesse caso a função *compare* que define isso, já que ela verifica qual a menor variável *cv* de duas *struct*.

A função *sort()* tem um custo de tempo e espaço de **n*log(n)**, onde n no nosso caso é a quantidade de porções definida pelo usuário.

Logo depois começamos a selecionar os alunos e determinar as porções que ele pode comer. O for da linha 9 percorrerá todos os máximos informados pelos alunos e no *while* da linha 15 a 26 calcular o valor do custo máximo desse aluno.

Esse *while* é controlado pela quantidade que o aluno ainda pode comer, que está armazenado em *e[i]*, a condição *k>=0* existe só para que não dê *loop* caso aconteça algum erro quando o usuário informa esse valor. A lógica consiste em pegar um porção do mais caro, ou pegar um pedaço dessa porção (quando o volume da porção é maior que o permitido pelo estomago do aluno). O *if* e *else* das linhas 18 e 20 permite que aconteça isso.

Por fim, vai subtraindo o consumo das porções do volume do estomâdo e salvando o valor do custo total dessas porções. Assim que ele comer todo o permitido sai do *while* e informa o valor com 4 casas decimais.

Analisando essa função, percebe-se que as comparações dentro dos loops de *for* e *while* irão repetir *a* (variável de quantidade de alunos) vezes *e[i]* (variável de volume do estomâdo) dependendo do tamanho do consumo. Sendo o *consumo = e[i]*, temos o melhor caso sendo constante dentro do *while* e com o *for* sendo linear (*a*), já o pior caso seria ele comer todas as porções disponíveis, sendo então *n* vezes (variável da quantidade de tipos do coffee), ou seja, temos *a*n* no pior caso.

Como o *a* só pode ir até 100, concluímos que essa função possui uma ordem de crescimento, em relação ao tempo, no pior e melhor caso n*log n, correspondente a função de ordenação.

Já em relação a memória podemos concluir que essa é constante, já que trata-se de uma função interativa que não requer memória proporcional as entradas.

Na função *main* está ocorrendo a leitura e a chamada da função até que seja informado 0 0 na leitura na quantidade das diferentes comidas e dos alunos.

Foi realizado o teste com a entrada do arquivo *coffee* e depois comparado com o arquivo de saída e não houve diferenças, confirmando assim que o algoritmo implementado passou pelos casos de testes.