Universidade Federal do Maranhão
Departamento de Informática
Ciência da Computação
Linguagem de programação I
Prof.: Francisco Glaubos

Prova Final

2022.1

Suponha que um grupo de trabalhadores precise executar um conjunto de tarefas e, para cada trabalhador e tarefa, haja um custo para atribuir o trabalhador à tarefa. Crie um programa em C, que atribua a cada trabalhador no máximo uma tarefa, sem que dois trabalhadores executem a mesma tarefa, minimizando o custo total.

Exemplo:

Há cinco trabalhadores (numerados de 0 a 4) e quatro tarefas (numerados de 0 a 3). Os custos de atribuição de trabalhadores a tarefas são mostrados na tabela a seguir.

Trabalhador	Tarefa 0	Tarefa 1	Tarefa 2	Tarefa 3
0	90	80	75	70
1	35	85	55	65
2	125	95	90	95
3	45	110	95	115
4	50	100	90	100

```
// exemplo de entrada
custos[5][4] = [
    [90, 80, 75, 70],
    [35, 85, 55, 65],
    [125, 95, 90, 95],
    [45, 110, 95, 115],
    [50, 100, 90, 100],
]
```

Saída (em arquivo):

```
Custo total = 265,0

Trabalhador 0 atribuído à tarefa 3. Custo = 70
Trabalhador 1 atribuído à tarefa 2. Custo = 55
Trabalhador 2 atribuído à tarefa 1. Custo = 95
Trabalhador 3 atribuído à tarefa 0. Custo = 45
```

Obs: Cada trabalhador pode ser atribuído a no máximo uma tarefa, sem que dois trabalhadores executem a mesma tarefa. No exemplo acima, como há mais trabalhadores do que tarefas, um trabalhador não receberá uma tarefa.

Esta questão será pontuada da seguinte maneira:

- (4,0 pts) Solução do problema, com pontuação proporcional ao custo de produção obtido. Quanto menor o custo, mais próximo da pontuação máxima.
- (2,0 pts) Uso de alocação dinâmica
- (2,0 pts) Uso de structs

- (2,0 pts) Uso de arquivos para imprimir a solução do problema em um arquivo `solucao.txt`

Aluno: Luiz Eduardo Batalha

Matrícula:2021051963

Código Resposta

```
#include "stdio.h"

#include "stdlib.h"




// Configuração do programa

#define QUANT_TAREFAS 4

#define TRABALHADORES 5




// Struct principal

typedef struct acoes

{

    int utilizados[QUANT_TAREFAS];

    int escolhido[QUANT_TAREFAS];

}acoes_st;
```

```
int main(void) {

    int final = 0; // Contém o custo total

    int menor; // Para uso dentro do algoritmo

    acoes_st* acoes = (acoes_st*)malloc(sizeof(acoes_st));

    FILE* arquivo = fopen("saida.txt", "w");

    if (arquivo == NULL) {

        printf("Erro no arquivo");

        return 0;

    }




    int entrada[TRABALHADORES][QUANT_TAREFAS] = {{90 , 80 , 75 ,
   70 }, {35 , 85 , 55 , 65 }, {125, 95 , 90 , 95 }, {45 , 110,
   95 , 115}, {50, 100, 90, 100}};




    // O grande

    acoes->utilizados[0] = 1;

    for (int i = 0; i < QUANT_TAREFAS; i++) {

        menor = 1;
```

```
        if (i == QUANT_TAREFAS-1) {
            menor = 0;
        }
        // Procura o menor valor dentro da linha
        for (int j = 1; j < QUANT_TAREFAS; j++) {
            if (entrada[i][menor] > entrada[i][j]) {
                if (acoes->utilizados[j] == 0) {
                    menor = j;
                }
            }
        }

        // Adiciona logo o menor no custo final
        final += entrada[i][menor];
        // Bota o menor na lista
        acoes->escolhido[i] = menor;
        // Já foi escolhido - Essa coluna não pode mais ser
escolhida
        acoes->utilizados[menor] = 1;
    }
```

```
    fprintf(arquivo, "Custo total = %d\n\n", final);

    for (int i = 0; i < QUANT_TAREFAS; i++) {

        fprintf(arquivo, "Trabalhador %d atribuído à tarefa %d.
  Custo = %d\n", i, acoes->escolhido[i],
  entrada[i][acoes->escolhido[i]]);

    }

}
```