SCC0220 - Laboratório Introdução à Ciência da Computação II

Relatório de execução do trabalho prático 11

Alunos	**NUSP**
Alec Campos Aoki	15436800
Juan Henrique Passos	15464826

Trabalho 11 - Tour

Grafos

Comentário

O trabalho prático 11 consiste em um conjunto de locais (nós), em que cada local está ligado há apenas um outro local (aresta), o que representa caminhos a serem percorridos e seus determinados destinos. Para resolver o problema, deve-se encontrar para cada passeio (pelo grafo) o destino final, sendo que cada passeio possui um local inicial e o número de passos (transições de locais). Como entrada é dado um n, contido no intervalo $1 < n < 2.10^5$, que representa a quantidade de locais, e a quantidade de passeios é dada por um k contido no mesmo intervalo de n. Após isso, é informado os locais e seus destinos, sendo que é dado o nome de um local e o destino a qual ele está ligado de forma numérica. Os números dos locais correspondentes estão definidos pela sua ordem de inserção, ou seja, o primeiro local a ser inserido é representado por 1. Para modelar o problema, foi utilizado um vetor de nós, em que cada posição do vetor possui um ponteiro para a posição de destino e o nome da posição atual, além de que cada posição do vetor indica o seu número - 1 (usado para não desperdiçar o espaço zero de memória).

Código

main.c

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

typedef struct no_ NO;

struct no_
{
    NO *proximo;
    char *local;
};

// Modularização código
```

```c
NO *ler_dados(int quant_no);
void cidade_destino(int quant_no, NO *grafo, int passos, char *buffer);
void desalocar_memoria(int quant_no,NO **grafo);

// Função principal.
int main(void)
{
    int quant_no, quant_passeios;
    scanf("%d %d", &quant_no, &quant_passeios);

    NO *grafo = ler_dados(quant_no);

    // Verificar passeios
    for(int i = 0; i < quant_passeios; i++)
    {
        // Armazenar cidade de início.
        char buffer[100];
        // Passos a serem percorridos
        int passos;

        scanf(" %s %d", buffer, &passos);
        cidade_destino(quant_no, grafo, passos, buffer);
    }

    desalocar_memoria(quant_no, &grafo);

    return 0;
}

// Le e guarda a informações dos nós do grafo.
NO *ler_dados(int quant_no)
{
    NO *grafo = (NO*) malloc(quant_no*sizeof(NO));
    if(grafo != NULL)
    {
        for(int i = 0; i < quant_no; i++)
        {
            // Ler nome do teclado e destino do teclado.
```

```
            int destino;
            char buffer[100];
            scanf(" %s %d", buffer, &destino);
            destino--; // Ajuste para base 0.

            // Ajusta tamanho para guardar nome(+ 1 para o \0).
            grafo[i].local = (char*) malloc(sizeof(char)*(strlen(buffer) + 1));
            strcpy(grafo[i].local, buffer);

            // Conectar local com seu destino.
            grafo[i].proximo = &grafo[destino];
        }
    }

    return grafo;
}

// Encontra cidade de destino de imprimi ela.
void cidade_destino(int quant_no, NO *grafo, int passos, char *buffer)
{
    // Posição da cidade no vetor
    // que se inicia o percurso.
    int origem = -1;
    // Procura-se cidade no grafo.
    for(int i = 0; i < quant_no; i++)
    {
        // strcmp retorna 0 se encontrar cidade.
        if(strcmp(buffer, grafo[i].local) == 0)
        {
            // Salva ponto de partida.
            origem = i;
            break;
        }
    }
    // Caso não seja encontrado.
    if(origem == -1)
    {
        printf("Cidade não está no grafo\n");
```

```
        return;
    }

    // Realizar o percurso com ponteiro.
    NO *estado_atual = &grafo[origem];
    for(int i = 0; i < passos; i++)
    {
        estado_atual = estado_atual->proximo;
    }
    // Imprimi destino.
    printf("%s\n", estado_atual->local);
    return;
}

// Função para desalocação de memória.
void desalocar_memoria(int quant_no,NO **grafo)
{
    for(int i = 0; i < quant_no; i++)
    {
        // Desalocar memória dos locais.
        free((*grafo)[i].local);
        (*grafo)[i].local = NULL;
    }
    // Desalocar memoria alocada para o grafo.
    free(*grafo);
    *grafo = NULL;
}
```

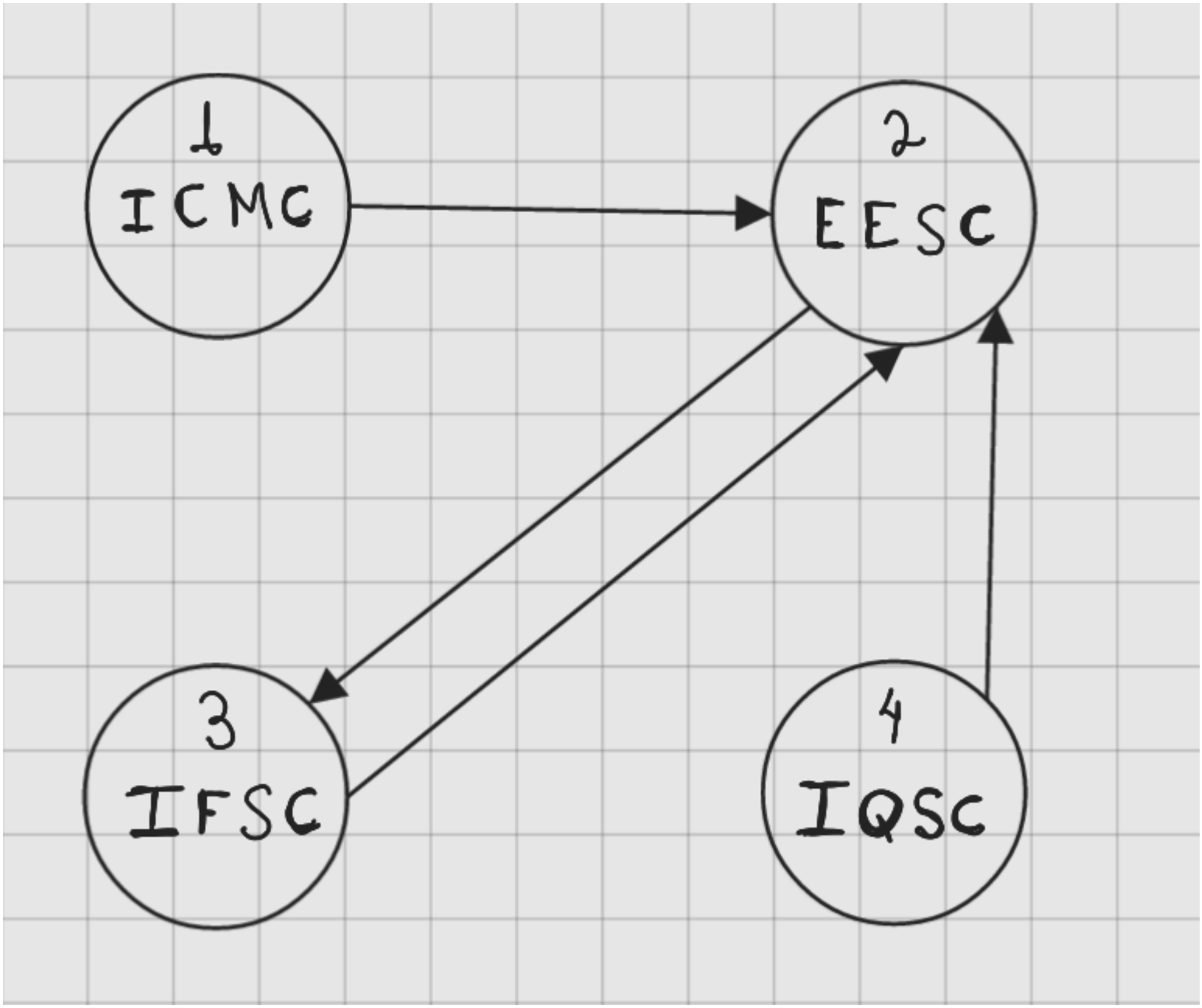
Saída

Caso teste 5 do run codes com 2 locais e 10000 passeios:

```
USP

Tempo de execucao: 12847.000000ms
```

Grafo (Caso teste 2)


1
ICMC
2
EESC
3
IFSC
4
IQSC