Samai Soares Pereira **TP01**

2020054811

1. Introdução

O problema proposto foi implementar um algoritmo para calcular o peso máximo suportado pela via para um caminhão com insumos transitar de forma que não excederia o peso da pista limitante. Para solucionar esse trabalho foi utilizado apenas o algoritmo semelhante com BFS para encontrar o vértice de destino a partir de um vértice inicial, no qual há um loop que retira os menores caminhos do grafo e repete a BFS até que o vértice de destino não seja mais encontrado, assim é retornado o último caminho retirado pois este é o caminho limitante.

2. Implementação

O programa foi implementado utilizando apenas 3 arquivos fontes main.cpp, inicializamatriz.cpp e pesquisa.cpp no qual há três funções principais do código no qual são:

```
void EncontraMaiorCaminho(int **matriz, int inicial, int final, int n_cidades, int n_caminhos)
```

Essa função tem como principal objetivo encontrar o maior caminho possivel entre as cidades, essa função utiliza de duas funções para que ela seja executada a Encontracaminho que é uma função booleana e a EncontraMin.

Nessa função realiza um loop enquanto a função booleana mencionada acima retorna 1, removendo o menor caminho do grafo auxiliar e realizando um novo loop até que o loop é cancelado, assim a função exibindo o valor máximo permitido na via.

A segunda função principal do código é a Encontracaminho

```
bool EncontraCaminho(int **matriz, int inicial, int final, int n_cidades, int n_caminhos)
```

Essa função é a base principal do código, nesta função é realizada uma busca pelo vértice de destino a partir de um vértice de início qualquer indicado previamente.

Nessa função de um array auxiliar no qual tem funcionalidade semelhante a um heap, nele é armazenado a ordem de encontro dos vértices que é utilizado para realizar a busca pelos caminhos tais como já mencionado, semelhante a uma BFS ou uma DFS. Dessa forma o código percorre todos os caminhos possíveis de um vértice, caso tenha passado por todos, o grafo volta uma posição no heap e procura novos caminhos até que seja expirado o número total de caminhos, ou arestas possíveis.

```
int EncontraMin(int **matriz, int n_cidades)
```

No qual ela retorna o menor valor do grafo e transformando o caminho em –1.

3. Complexidade

Para calcular a complexidade do algoritmo vamos analisar cada função separadamente.

Observaremos cada loop e observaremos a que tem maior complexidade pois ela sobressairá sobre as demais.

Consideremos para fazer o cálculo de complexidade a variável **V** como número de cidades e a variável **E** como sendo a quantidade de caminhos entre as cidades.

main.cpp:

No documento main.cpp temos na função **main** um loop no qual tem complexidade O(V), pois cria uma matriz quadrática com tamanho (numero de cidades) x (numero de cidades)

Dessa forma a complexidade final é

O(V)

inicializamatriz.cpp

Nesse arquivo há duas funções no qual vamos analisar separadamente cada uma destas.

A função **preencheMatriz** tem como complexidade O(E) pois há um loop no qual itera sobre cada caminho do grafo.

A função **inicializaMatriz** tem como complexidade $O(V^2)$ uma vez que este itera duplamente para cada vértice do grafo.

Dessa forma a complexidade final desse arquivo é de

$O(E+V^2)$;

pesquisa.cpp

Nesse arquivo há quatro funções, responsáveis para execução do programa no qual apenas 3 serão relevantes para o cálculo de complexidade.

A função **EncontraMaiorCaminho** há quatro loops sendo o primeiro de complexidade igual a O(V) e o segundo igual a $O(V^2)$ estas duas responsáveis por criar uma matriz auxiliar que sera modificada para o cálculo de encontrar o caminho que suporta o máximo peso. O terceiro loop terá um sua complexidade separadamente pois ela terá complexidade igual a função O(EncontraCaminho*EncontraMin), e por fim temos um loop responsável por deletar a matriz criada logo acima que tem complexidade $O(V^2)$.

A função **EncontraCaminho** tem dois loops principais um O(V) responsável de criar um heap para guardar os vértices já caminhados o segundo loop sendo O(E) pois a complexidade O(E) loop para uma vez que ele analisa todos os caminhos do grafo.

A função **EncontraMin** tem complexidade $O(V^2)$ pois precisa passar por todos os valores armazenados para encontrar o menor valor de caminho, para isso ele utiliza um loop dentro de um outro loop.

Assim a complexidade final do arquivo é

$O(V^2)*O(E)$

Assim a complexidade final do programa será

$O(V^2*E)$

4. Pseudocódigo

```
Void EncontraMaiorCaminho(int **matriz, int inicial, int final, int n_cidades, int n_caminhos)
    Cria matriz auxiliar;
    Copia Conteudo da matriz principal para matriz auxiliar;
    while (EncontraCaminho(matrizaux, inicial, final, n_cidades, n_caminhos))//loop executa enquanto o retorno for 1
      {
        minimo = EncontraMin(matrizaux, n_cidades);//pega o minimo valor do grafo
      }

bool EncontraCaminho(int **matriz, int inicial, int final, int n_cidades, int n_caminhos)
{
Bool usado[numero cidades]//cria vetor para saber se um numero foi visitado
Int sequecia [numero cidades]//heap de numero de cidades
    For(i=0; i < numero cidades; i++)
    {
    usado[i]=false; //inicializa vetor
    }
Sequencia[inicial]=true;
Sequencia[0]=inicial;

while(quantidade usada<numero cidades)
{
If(matriz[i][j] != -1 e j>= 0 e numero cidades > j)
{
If(usado[j] == false)//se j é um vértice não percorrido
```

```
{
contador++
                    Valor++
                    if(valor>= numero cidade)//se valor é superior ao numero de cidaes break
          }
          Break;
          }
If(j==final)//se j igual final chegou no destino
{
Return true;
}
Sequencia[valor]=i;
I=j;
Usado[j] = true;
Numero usado++;
J=-1;
}
Else if(matriz[i][j] == -1 e j + 1 >= numero cidades)//se matriz=-1 e j+1
)
          Valor--;
If(valor==-1)
          {
          Break;
          }
i = sequencia[valor+1];
          j = -1;

}
          J++
if (count == n_caminhos||j>=n_caminhos)//se percorreu todos caminhos possíveis
     {
```

```
            break;
        }

    return false;//finaliza a iteraçao retornando falso
}
```

5. Instruções para Compilação

Para compilar o código necessita apenas executar o código na pasta do arquivo:

g++ main.cpp -o main

E para executar o programa, após compilado na linha de comendo escrever

main.exe < arquivoentrada.txt