Buscando os dados do grafo em um arquivo e armazenando em uma matriz de adjacência.

O programa utiliza uma estrutura de dados Grafo para representar o grafo.

A função criaGrafo aloca dinamicamente a memória para a matriz de adjacências com base no número de vértices fornecido em arquivo.

O arquivo deve ter o seguinte formato, conforme Figura 1: o número de vértices na primeira linha, o número de arestas na segunda linha e as arestas nas linhas seguintes, no formato vértice1 vértice2.

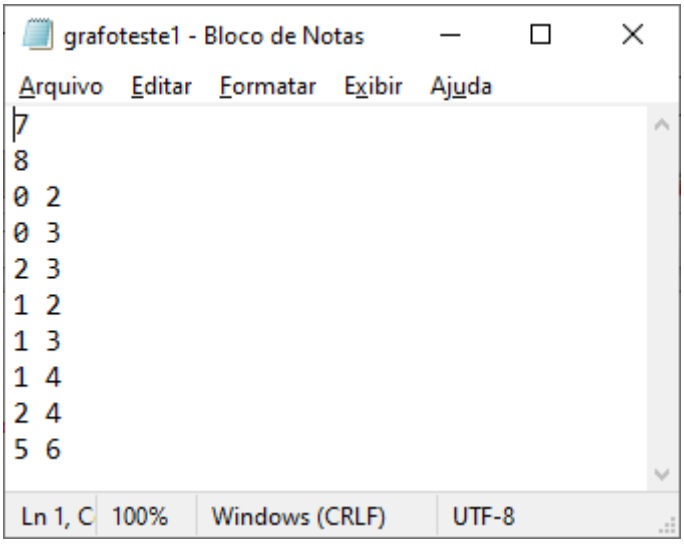

Figura 1 - Exemplo de arquivo grafoteste1.txt

Na Figura 2, temos a imagem do grafo utilizado como exemplo no arquivo grafoteste1.txt.

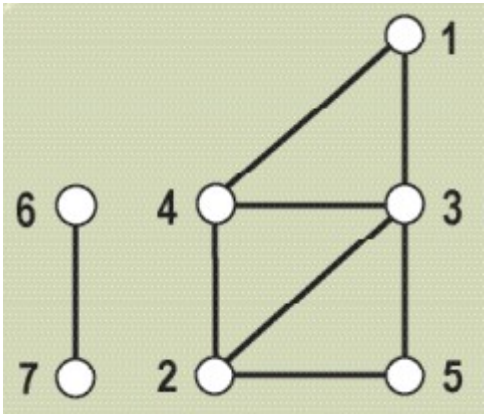

Figura 2 – Grafo do arquivo txt

A função adicionaAresta adiciona uma aresta na matriz de adjacências.

A função imprimeMatrizAdjacencias imprime a matriz de adjacências na tela (assim você verifica se ela está correta).

A função liberaMatriz libera a memória alocada para o grafo.

O arquivo está escrito em linguagem C. Você poderá usar também C++, Java, JavaScript.

Programa_Matriz_ de_ Adjacencia

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

 typedef struct {
     int** matriz;
     int numVertices;
 } Grafo;

 Grafo* criaMatriz(int numVertices) {
     Grafo* grafo = (Grafo*)malloc(sizeof(Grafo));
     grafo->numVertices = numVertices;

     // Aloca memória para a matriz de adjacências
     grafo->matriz = (int**)malloc(numVertices * sizeof(int*));
     for (int i = 0; i < numVertices; i++) {
         grafo->matriz[i] = (int*)malloc(numVertices * sizeof(int));
     }

     // Inicializa a matriz de adjacências
     for (int i = 0; i < numVertices; i++) {
         for (int j = 0; j < numVertices; j++) {
             grafo->matriz[i][j] = 0;
         }
     }

     return grafo;
 }

 void adicionaAresta(Grafo* grafo, int u, int v) {
     // Adiciona uma aresta na matriz de adjacências
     grafo->matriz[u][v] = 1;
     grafo->matriz[v][u] = 1; // se o grafo for não-direcionado
 }

 void imprimeMatrizAdjacencias(Grafo* grafo) {
     printf("Matriz de adjacencias:\n");
     for (int i = 0; i < grafo->numVertices; i++) {
         for (int j = 0; j < grafo->numVertices; j++) {
             printf("%d ", grafo->matriz[i][j]);
         }
         printf("\n");
     }
 }

 void liberaMatriz(Grafo* grafo) {
     // Libera a memória alocada para a matriz de adjacências
     for (int i = 0; i < grafo->numVertices; i++) {
         free(grafo->matriz[i]);
     }
     free(grafo->matriz);
```

```
51      free(grafo);
52  }
53
54  int main() {
55      char filename[100]; // nome do arquivo
56      FILE* file;
57
58      printf("Digite o nome do arquivo: ");
59      scanf("%s", filename);
60
61      // Abre o arquivo para leitura
62      file = fopen(filename, "r");
63      if (file == NULL) {
64          printf("Erro ao abrir o arquivo.\n");
65          return 1;
66      }
67
68      int numVertices, numArestas;
69      fscanf(file, "%d %d", &numVertices, &numArestas);
70
71      Grafo* grafo = criaMatriz(numVertices);
72
73      for (int i = 0; i < numArestas; i++) {
74          int u, v;
75          fscanf(file, "%d %d", &u, &v);
76          adicionaAresta(grafo, u, v);
77      }
78
79      fclose(file);
80
81      imprimeMatrizAdjacencias(grafo);
82
83      liberaMatriz(grafo);
84
85      return 0;
86  }
```

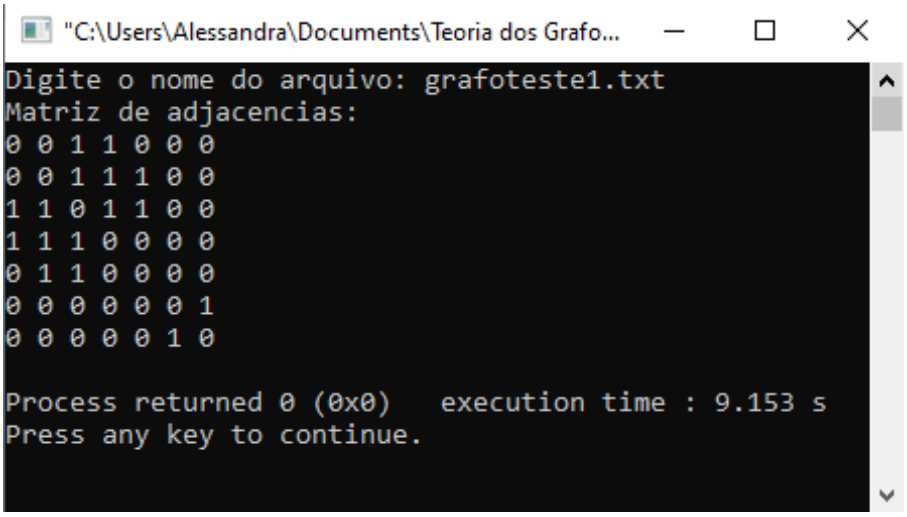

Figura 3 – Matriz de Adjacência