+FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION
OF HIGHER EDUCATION
ITMO UNIVERSITY

Report
on the practical task No. 5
“Algorithms on graphs. Introduction to graphs and basic algorithms on graphs”

Performed by
Chernobrovkin Timofey (412642)
Academic group J4133c
Accepted by
Dr Petr Chunaev

St. Petersburg
2023

Goal

Algorithms on graphs. Introduction to graphs and basic algorithms on graphs.

Problem

I. Generate a random adjacency matrix for a simple undirected unweighted graph with 100 vertices and 200 edges (note that the matrix should be symmetric and contain only 0s and 1s as elements). Transfer the matrix into an adjacency list. Visualize the graph and print several rows of the adjacency matrix and the adjacency list. Which purposes is each representation more convenient for?

II. Use Depth-first search to find connected components of the graph and Breadth-first search to find a shortest path between two random vertices. Analyse the results obtained.

III. Describe the data structures and design techniques used within the algorithms.

Theory

Graph theory is actively used in modeling various complex systems, including social, transportation, communication and other networks. The tools of graph theory allow to carry out a comprehensive analysis of data represented as graphs. In the current laboratory work it is proposed to study in practice various graph representations, as well as standard algorithms of graph traversal.

An undirected graph is a pair G = (V, E), where V = $\{v_i\}$ is the set of vertices (or nodes), and E =$\{e_{ij}\}$ = $\{(v_i, v_j)\}$ is the set of pairs of vertices called edges (or links). The number of vertices is denoted by |V| and the number of edges by |E|. An oriented graph is a graph in which edges have directions (orientations). A weighted graph is a graph in which weights are assigned to each edge. A simple graph is a graph in which only one edge is possible between a pair of vertices. A multigraph is a generalization of a simple graph in which more than one edge between a pair of vertices is possible in the graph. A complete graph is a graph in which all vertices are connected by an edge. A path (chain) in a graph is a sequence of pairwise different edges connecting two different vertices. The length of a path (chain) is the number of edges (or the sum of edge weights) in the path (chain). Vertices v1 and v2 in a graph are called connected if there exists a path from v_1 to v_2. Otherwise, these vertices are called unconnected. A connected graph is a graph in which any pair of vertices is connected. Otherwise, the graph is called unconnected. The connectivity component of a graph is the maximum connected subgraph of the graph.

Let us turn to different types of graph representation. The first of them is the adjacency matrix, i.e., a matrix whose rows and columns are indexed by vertices and whose cells contain a Boolean value. are indexed by vertices and whose cells contain a Boolean value (0 or 1) indicating whether the corresponding vertices are adjacent (for weighted graphs the corresponding weights

are instead of 1). The adjacency matrix (as a 2D array) requires $O(|V|^2)$ memory. An example of the adjacency matrix is shown in Figure 1.

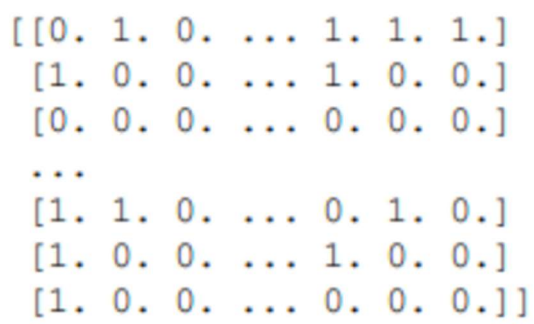

Figure 1 – An example of an adjacency matrix

Another type of representation is the adjacency list, i.e. collections of vertex lists, where for each vertex of the graph there is a list of adjacent vertices. list of vertices adjacent to it. An adjacency list (as a 1D array of lists) requires $O(|V| + |E|)$ memory. An example of an adjacency list is shown in Figure 2.

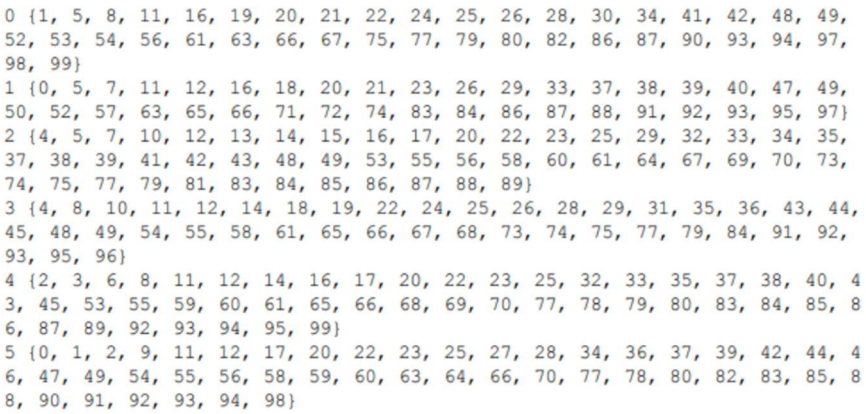

Figure 2 – An example of an adjacency list

For a sparse graph, i.e., a graph in which most pairs of vertices are not connected by edges, $|E| << |V|^2$, the adjacency list is significantly more efficient to store than the adjacency matrix.

For visualization purposes, graph images in the form as in Figure 3 are also used. In general, the task of informative visualization of graphs is very difficult.

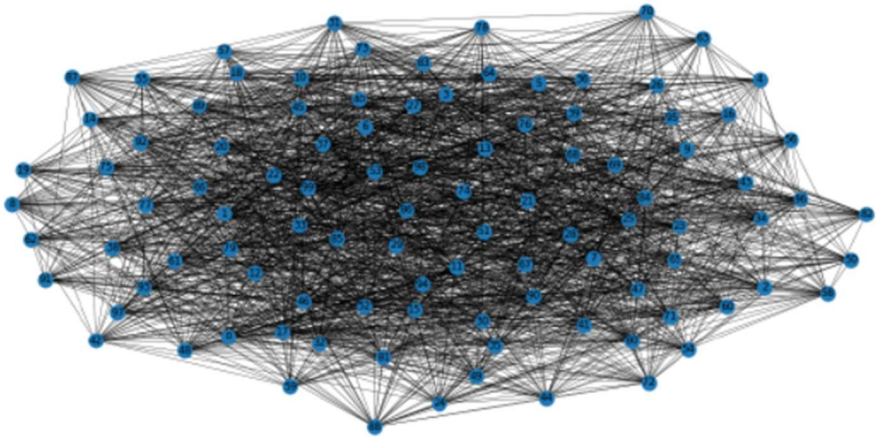

Figure 3 – Example of graph visualization

Let us turn to classical graph traversal algorithms. Depth-first search (DFS) is a graph traversal algorithm that starts from a selected root vertex and goes "deep" into the graph as far as possible before traversing from a new vertex. Its time complexity is O(|V| + |E|). Breadth-first search (BFS) is a graph traversal algorithm in which the traversal starts at a selected root vertex and explores all adjacent vertices at the current "depth" before moving on to vertices at the next "depth". This traversal algorithm uses a strategy that is in some ways the opposite of DFS. Its time complexity is also O(|V| + |E|).

Both algorithms can be applied to find the shortest path between vertices and to find the connectivity components of a graph.

Materials and methods

In this task, all calculations were performed on the student's personal laptop. The work was performed in the Python programming language.

Results

I. A random adjacency matrix was generated for a simple undirected unweighted graph with 100 vertices and 200 edges. The generated graph was visualized (Figure 4). Several rows of the adjacency matrix and the adjacency list are shown in Figure 5 and Figure 6, respectively.

The adjacency matrix is convenient for representing dense graphs in which the number of edges is close to the maximum possible number of edges.

The adjacency list is more suitable for more sparse graphs, where the number of edges is much smaller than the maximum possible number of edges.

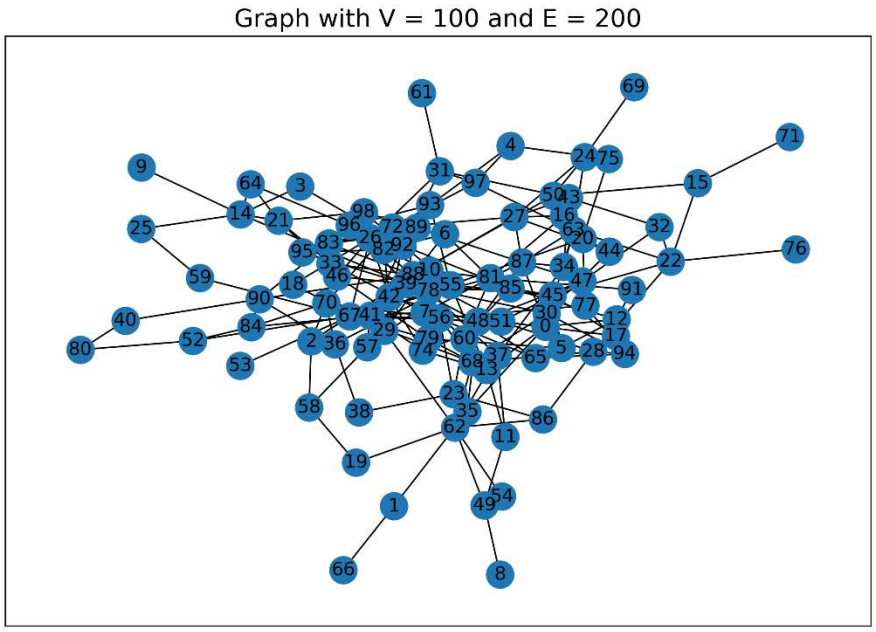

Figure 4 – Graph visualisation

```
several rows of the matrix adjacency:
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 1. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.
  0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
  0. 0. 0. 0.]]
```

Figure 5 – Several rows of the matrix adjacency

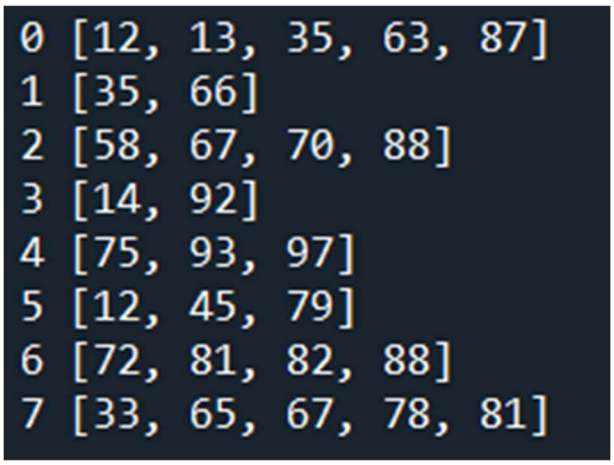

Figure 6 – Several rows of the adjacency list

II. Using depth-first search find the connected components of the graph and width-first search find the shortest path between two arbitrary vertices.

The number of connected components is 98 and the number of unconnected components is 2.

The result of finding the shortest path between two random points is shown in Figure 7.

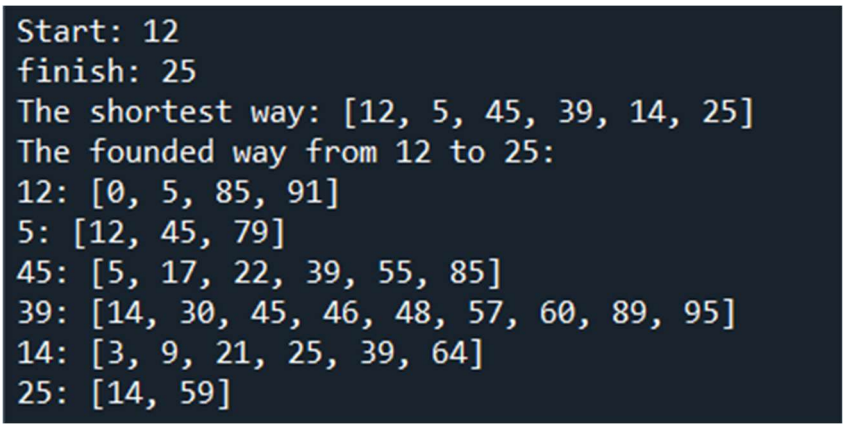

Figure 7 – the shortest way between two random points

III. An adjacency matrix is a V x V array of vertices. Each row and column represent a vertex.

A contiguity list is a representation of a graph as an array of linked list.

BFS algorithm uses queue data structure and DFS uses stack data structure. The execution time of both is O(V + E).

Conclusion

This assignment explored the use of different representations of graphs and basic algorithms on graphs (depth-first and breadth-first search).

A graph was generated. Then it was visualized, and its adjacency matrix and adjacency list were found. Using depth-first search found the connected components of the graph and width-first search the shortest path between two arbitrary vertices. Also looked at data structures and design techniques used in algorithms.

Appendix

GitHub link:

https://github.com/LesostepnoyGnom/Homework/blob/main/Task_5_03.10.23/Task_5_Chernobrovkin_J4133c.py