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
Denis Zakharov,
Maxim Shitilov
Academic group J4132c
Accepted by
Dr Petr Chunaev

St. Petersburg
2022

Goal

The use of different representations of graphs and basic algorithms on graphs (Depth-first search and Breadth-first search).

Formulation of the problem

The use of different representations of graphs.

Brief theoretical part

Depth First Search (DFS):

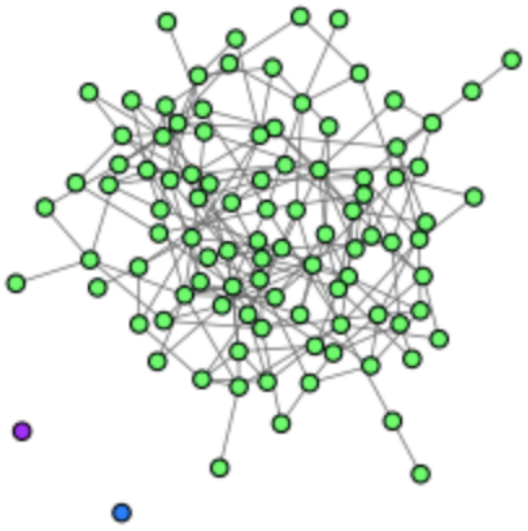
It is a way to traverse the graph. In this, we visit all the vertices and edges and traverse the graph. In depth-first search, the stack data structure is used that follows the LIFO(Last In and First Out) principle. Depth-first search produces a non-optimal solution. In depth-first search, we keep on exploring or visiting vertices till we reach a dead-end where there are no more edges to traverse from that vertex and then we backtrack.

Breadth-First Search (BFS):

Breadth-first search is also called a level order traversal. Breadth-first search is an algorithm to traverse the graph level by level. In the traversing process, we have to visit all vertices and edges. In this, we can take any node as a root node during traversal starting. For BFS, a queue data structure would be used that follows FIFO(First In First Out) principle. We visit nodes level wise. First, we complete the top level and then move on to lower levels.

Results

1) Explain differences (if any) in the results obtained.



Pic. 1 — Visualization of generated graph

```
Shortest path between vert 19 and vert 37 = [[19, 71, 37]]
```

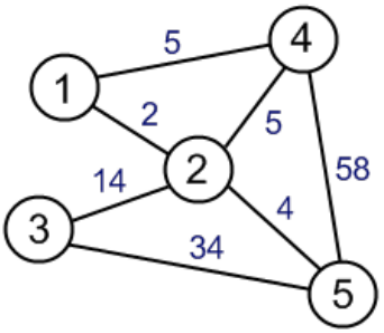
Pic. 2 — Shortest path

Conclusions

To sum up, we have used different representations of graphs (adjacency matrix, adjacency list), and it's better to store graph as an adjacency list, cause it Space Complexity $O(number\ of\ vertices\ *\ longest\ path\ from\ vertices)$ and basic algorithms on graphs (DFS, BFS). As can be seen from the graphs and console output above, it's really hard to visualize a graph. For these purposes we've tried to use plotly, but it was no good. So why we decided to use built in igraph/networkx visualization.

Questions for self-monitoring from Russian educational materials

1. Describe the graph below: oriented/undirected; weighted/unweighted; connected/disconnected.



The graph presented is *non-oriented, weighted, connected*.

2. Is the time complexity of graph traversal algorithms the same in width and depth (with the same graph storage methods)?

BFS:

Time complexity is $O(|V|)$, where $|V|$ is the number of nodes. You need to traverse all nodes.

Space complexity is $O(|V|)$ as well — since at worst case you need to hold all vertices in the queue.

DFS:

Time complexity is again $O(|V|)$, you need to traverse all nodes.

Space complexity - depends on the implementation, a recursive implementation can have a $O(h)$ space complexity [worst case], where h is the maximal depth of your tree.

Using an iterative solution with a stack is actually the same as BFS, just using a stack instead of a queue — so you get both $O(|V|)$ time and space complexity.

(*) Note that the space complexity and time complexity is a bit different for a tree than for a general graphs because you do not need to maintain a visited set for a tree, and $|E| = O(|V|)$*, so the* $|E|$ *factor is actually redundant.*

Appendix

DataLore: site. – URL: https://datalore.jetbrains.com/notebook/RemqSkuJwmr1PM4Gc3cBqB/DkF3GncdKl719NdXTKb9WJ (circulation date: 29.09.2022)