FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER EDUCATION ITMO UNIVERSITY

Report
on the practical task No. 6

“Algorithms on graphs.
Path search algorithms on weighted graphs”

Performed by
Kozlov Alexey
Dmitriy Koryakov
J4133c
Accepted by
Dr Petr Chunaev

St. Petersburg
2022

Оглавление

Goal

The use of path search algorithms on weighted graphs (Dijkstra's, A* and BellmanFord algorithms).

Problems

1. Generate a random adjacency matrix for a simple undirected weighted graph of 100 vertices and 500 edges with assigned random positive integer weights (note that the matrix should be symmetric and contain only 0s and weights as elements). Use Dijkstra's and Bellman-Ford algorithms to find shortest paths between a random starting vertex and other vertices. Measure the time required to find the paths for each algorithm. Repeat the experiment 10 times for the same starting vertex and calculate the average time required for the paths search of each algorithm. Analyze the results obtained.
2. Generate a 10x20 cell grid with 40 obstacle cells. Choose two random nonobstacle cells and find a shortest path between them using A* algorithm. Repeat the experiment 5 times with different random pair of cells. Analyze the results obtained.
3. Describe the data structures and design techniques used within the algorithms.

Brief theoretical part

Undirected weighted graph is G = (V, E, w), where V = {v1, v2, · · · , vn} is the set of vertices, E = {e1, e2, · · · , em} is the set of edges and each vertex vi ∈ V is associated with a wight w(vi)

The number of vertices is usually denoted by |V|.

The number of edges is usually denoted by |E|.

The adjacency matrix is a matrix whose rows and columns are indexed by vertices and whose cells contain a Boolean value that indicates whether the corresponding vertices are adjacent (for weighted graphs, it contains corresponding weights instead of 1s.). The matrix (stored as a 2D array) requires O(|V|2) of space.

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph. Time complexity of it is O(E * LogV).

Bellman-Ford algorithm is an algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted graph. Time complexity of it is O(E * V).

A* algorithm is an informed search algorithm that computes shortest paths from a single source vertex to another vertex. Time complexity of it is O(E).

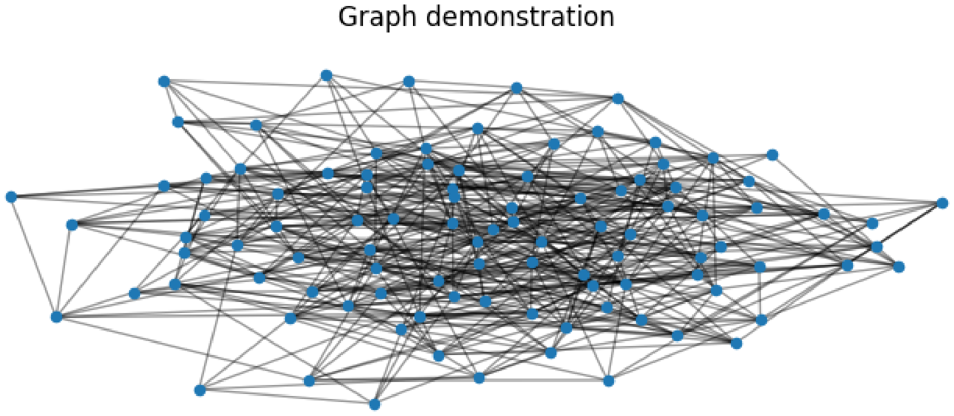
Solution

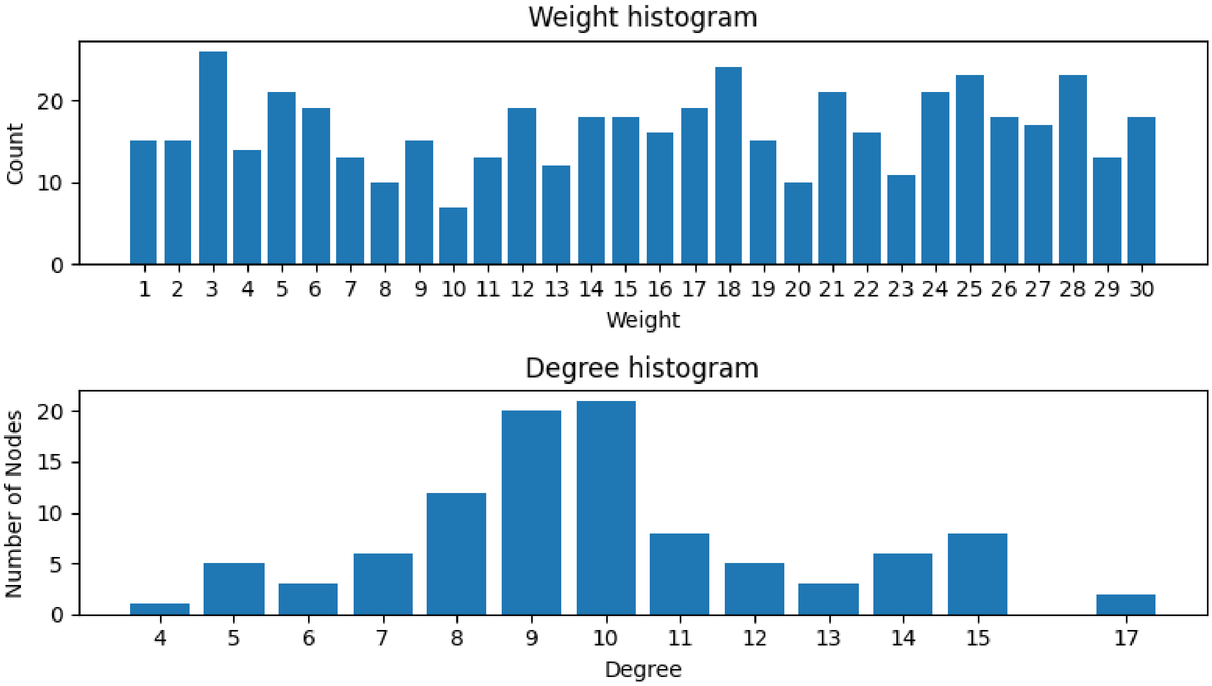
Undirected weighted graph

1. Random adjacency matrix for weighted undirected graph with 100 vertices and 500 edges.

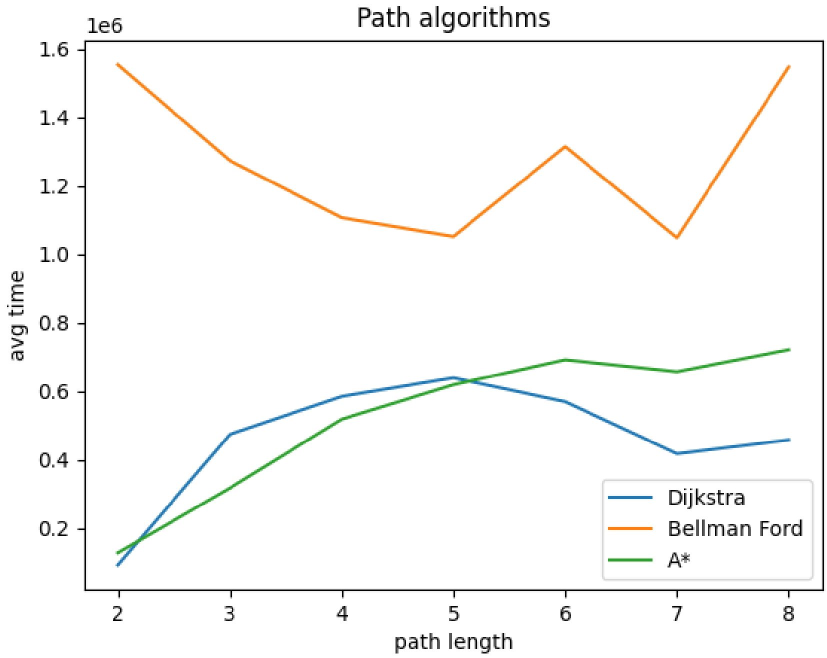
```
[[ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, 0,  0,  0,  0,  6,  0,  0,  0,  6,  0,
0,  0,  0,  0, 18,  0, 0 ,  0,  0,  0,  0,  0,  0,  0, 25,  0,  0,  0,  0,  0,  0,  0, 0,  0,  0,  0,
0,  0,  0,  0,  0,  0,  0, 17,  0,  0,  0,  0, 0,  0,  0,  0,  0,  0,  0,  0,  3,  0,  0,  0, 22, 14,
0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  4, 0,  0,  0,  0],

[ 0,  0,  0,  0, 11, 30,  0,  0, 28,  0,  0,  0,  0,  0,  0,  0, 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0, 25,  0,  0, 14,  0, 0,  7,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0, 0, 26,  0, 11,
0,  0,  0,  0,  0,  0,  1,  0,  0, 20,  0,  0, 0 ,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
0,  0, 0 ,  0,  0, 16,  0, 24,  0,  0,  0,  0,  0,  0,  0,  0,  0, 0,  0,  0,  0]]
```





2. Experiment (repeat 10 times)
    1. Choose start node
    2. Find path between start node and all others by different algorithm
    3. Store time of experiment
3. Results



4. Conclusion

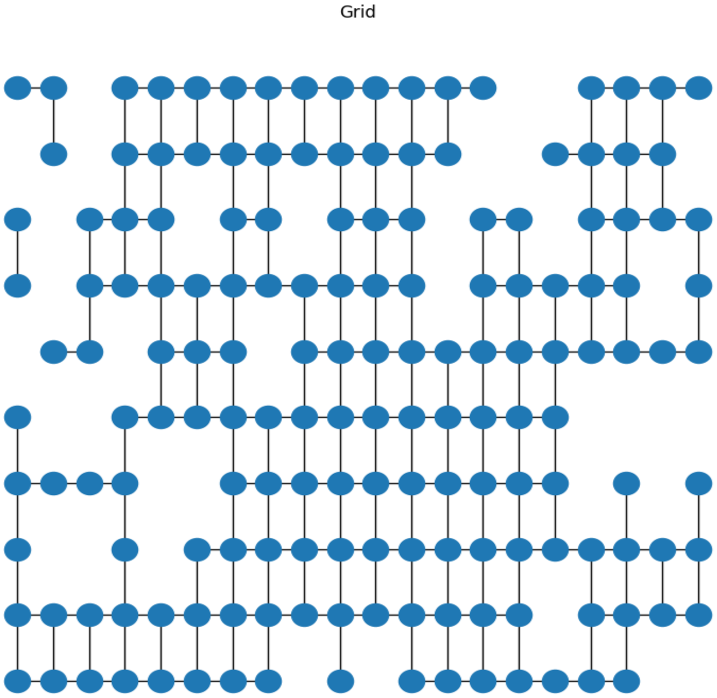
As a results, we can conclude that A* is used for problem to find path between pair of vertices and Dijkstra's and Bellman-Ford algorithms are used for problem to find path between source to all other vertices.

Comparison shows that Bellman-Ford algorithm is slower than Dijkstra`s algorithm, but first one can handle with graph, which contains negative weights of edges.

A* algorithm is similar with Dijkstra`s algorithm in time complexity during experiments.

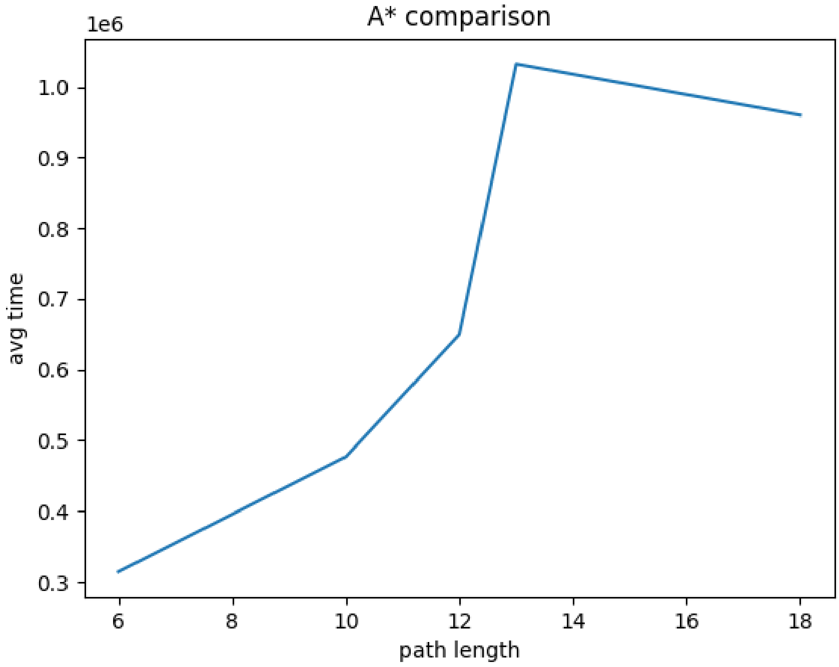
2-D grid

1) 10x20 cell grid with 40 obstacle cells

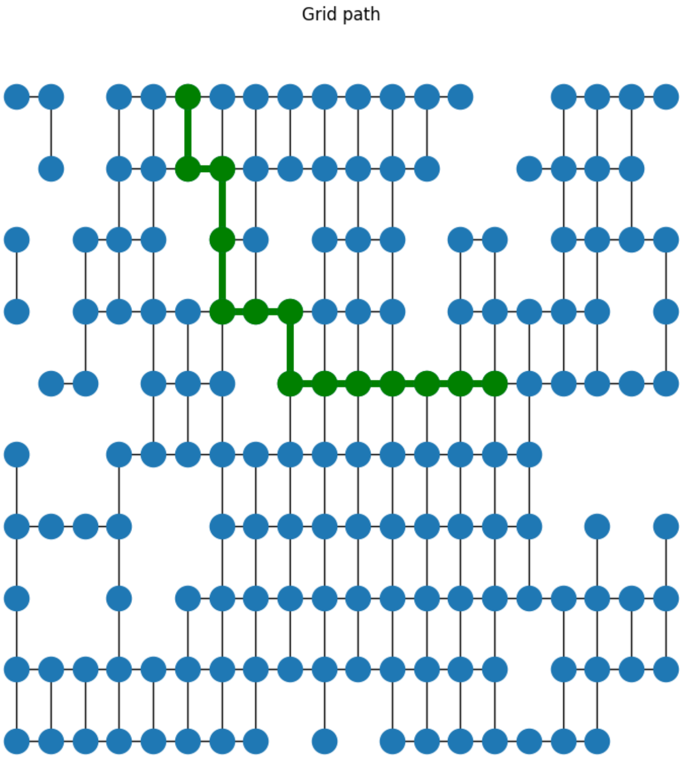


2) Experiment (repeat 5 times)
   1. Randomly choose start and target cells
   2. Find path between its cells using A* algorithm
   3. Store time of experiment

3) Results



4) Example of path



5) Conclusion

Dijkstra`s algorithm extension, A* algorithm, which find path using direction to target node, only used when finding paths between two vertices. A* algorithm guaranteed to expand fewer nodes than others.

Appendix

Link to github - https://github.com/OnlyOneUseAcc/algorithms-RD-practise