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:
Roman Bezaev
J4133c

Accepted by:
Dr Petr Chunaev

St. Petersburg
2021

1 Goal

The use of path search algorithms on weighted graphs (Dijkstra's, A* and Bellman- Ford algorithms)

2 Formulation of the problem

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. Analyse the results obtained.
2. Generate a 10x20 cell grid with 40 obstacle cells. Choose two random non-obstacle cells and find a shortest path between them using A* algorithm. Repeat the experiment 5 times with different random pair of cells. Analyse the results obtained.
3. Describe the data structures and design techniques used within the algorithms

3 Brief theoretical part

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph. It generates a shortest path tree (SPT) with the source as a root, with maintaining two sets: one set contains vertices included in SPT, other set includes vertices not yet included in SPT. At every step, it finds a vertex which is in the other set and has a minimum distance from the source. Algorithm has time complexity $O(|V|^2)$. Dijkstra's algorithm is based on a greedy technique.

A* algorithm is another algorithm for finding the shortest paths in a graph. A* is an informed search algorithm, or a best-first search: starting from a specific starting node of a graph, it aims to find a path to the given goal node having the smallest cost. It does this by maintaining a tree of paths originating at the start node and extending those paths one edge at a time until its termination criterion is satisfied. At each iteration of its main loop, A* needs to determine which of its paths to extend. It does so based on the cost of the path and an estimate of the cost required to extend the path all the way to the goal. A* exploits heuristic technique which in general helps it to work faster that Dijkstra's algorithm. Its estimated time

complexity is $O(|E|)$.

Another algorithm for finding the shortest path is **Bellman-Ford algorithm**. The idea can be expressed as follows: at i-th iteration, Bellman-Ford calculates the shortest paths which has at most i edges. As there is maximum $|V| - 1$ edges in any simple path, $i = 1, ..., |V| - 1$. Assuming that there is no negative cycle, if we have calculated shortest paths with at most i edges, then an iteration over all edges guarantees to give shortest paths with at most $(i+1)$ edges. To check if there is a negative cycle, make $|V|$-th iteration. If at least one of the shortest paths becomes shorter, there is such a cycle. The time complexity of this algorithm is $O(|V||E|)$ (which is $O(|V|^3)$ in the worst-case scenario). Bellman-Ford uses dynamic programming technique.

4 Results

4.1 Task 1

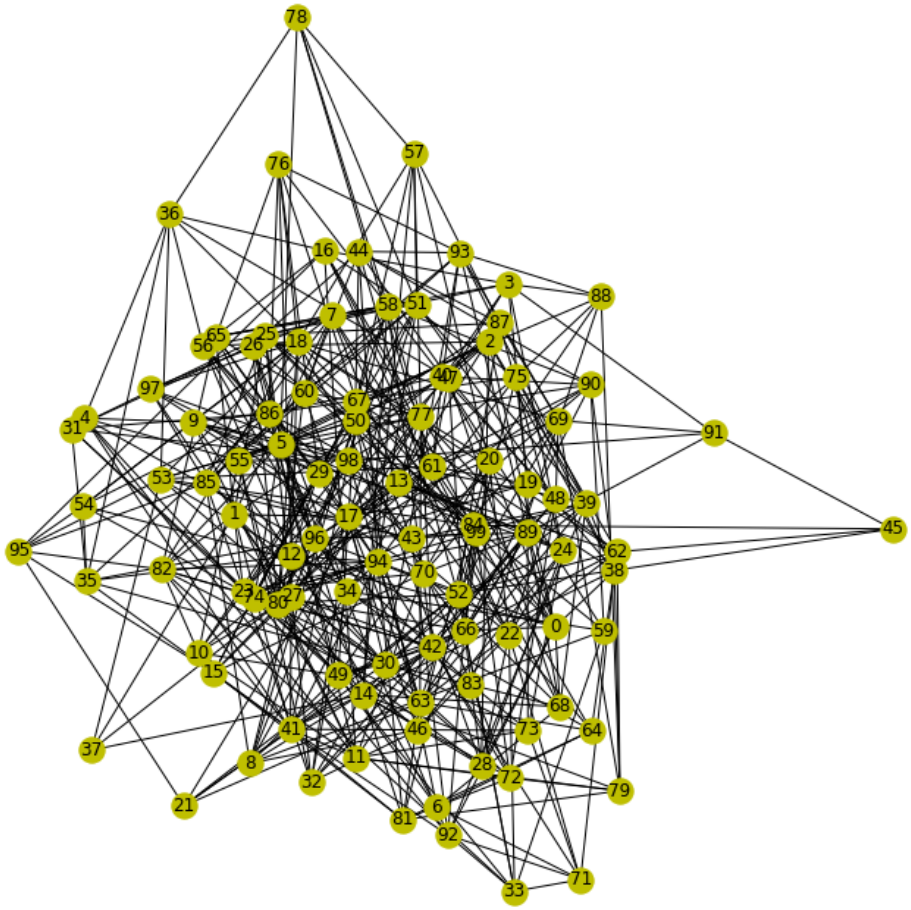

Figure 1: Generated graph

Weighted graph containing 100 nodes and 500 edges with positive weights lying in the range from 1 to 100 was created. The graph contained only one connected component.

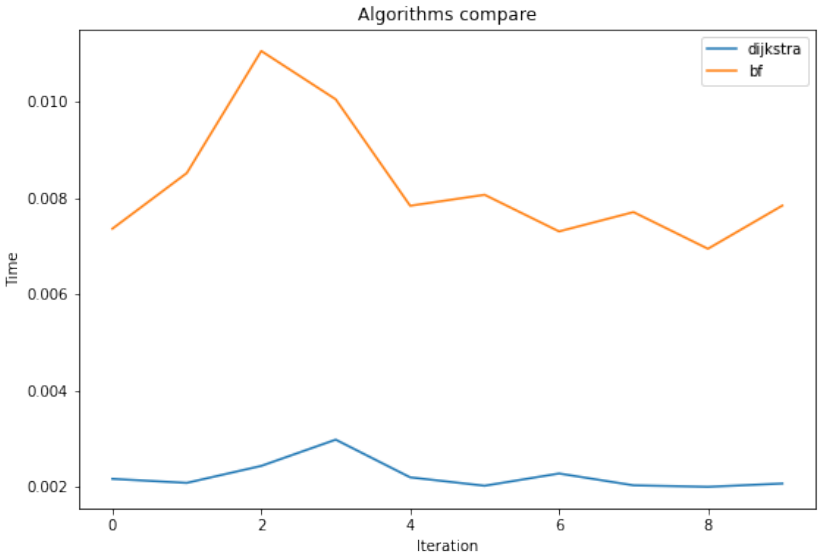

Figure 2: Time

The time required for Bellman-Ford algorithm (0.08 – 0.01 seconds) was much higher than the time required for Dijkstra's algorithm (approx 0.002 second).

4.2 Task 2

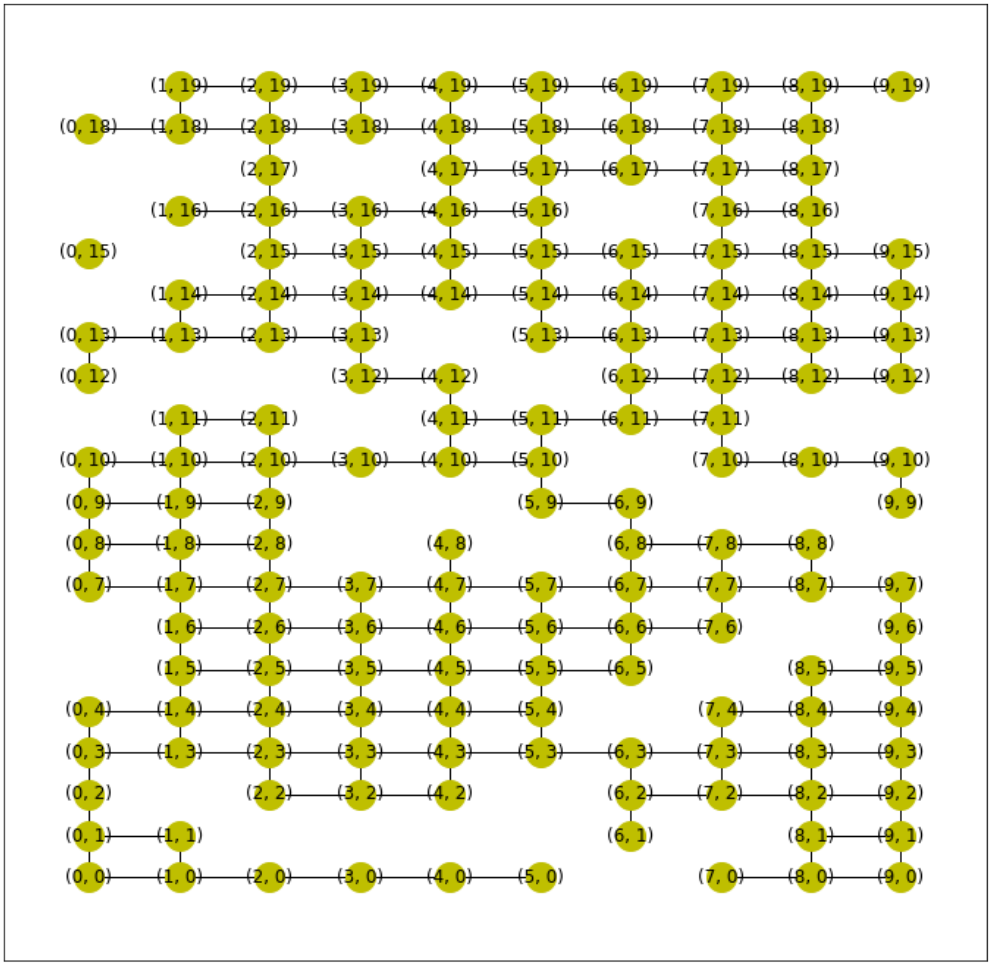

Figure 3: Grid with obstacles

For example, the path will be found between two random points: $(9, 19)$ and $(1, 0)$. Obtained path is 30 cells long.

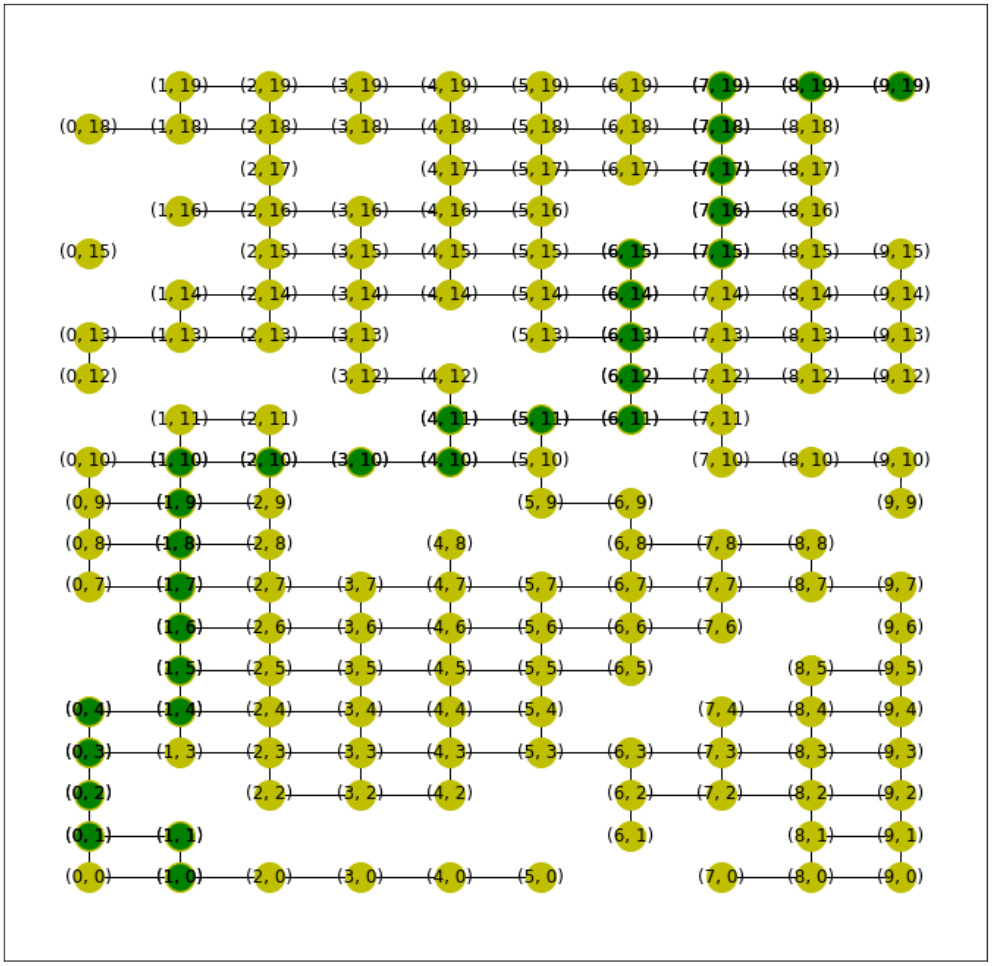

Figure 4: Shortest path according to A*

Obviously, algorithm can't find a path between points that are not connected.

I decided to make 50 experiments to make a more understandable plot. Paths between 50 different pairs of points were found. Time plot of this operation:

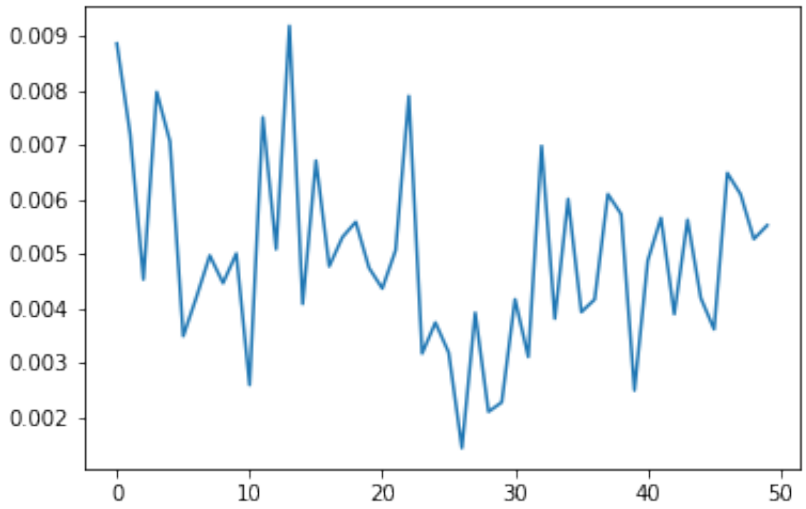

Figure 5: Path finding time

Average time of finding a path is 0.00496 seconds. Big values mean that in this experiment algorithm probably spent some time when it chose the wrong direction.

5 Conclusions

Three major algorithms for searching the shortest path in the graph were analyzed. Dijkstra's algorithm is good in the case of graphs with positive edge weights. Bellman-Ford algorithm is useful for the case if the graph contains negative edge weights, but is slower if used in graph with positive edge weights. A* algorithm is an interesting Dijkstra's algorithm extension that utilized special heuristic that allows it to select the direction and find shortest paths in that direction without spending time on the other directions.

6 Appendix

https://github.com/trixdade/ITMO_algorithms/blob/master/Lab6/Lab6.ipynb