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
Daan Rodríguez

Academic group
J4132c
J4133c
Accepted by
Dr Petr Chunaev

St. Petersburg
2023

Goal

The goal of task 6 is to constrct a weighted graph and find the shortest distance with path search algorithms

Formulation of the problem

1. About Q1: generate undirected weigthted graph and its adjacency matrix of 100 vertices and 500 edges

```
: array([[0, 0, 0, ..., 0, 0, 0],
         [0, 0, 0, ..., 3, 0, 0],
         [0, 0, 0, ..., 0, 0, 0],
         ...,
         [0, 3, 0, ..., 0, 0, 0],
         [0, 0, 0, ..., 0, 0, 0],
         [0, 0, 0, ..., 0, 0, 0]], dtype=int32)
```

Figure 1 - adjacency matrix

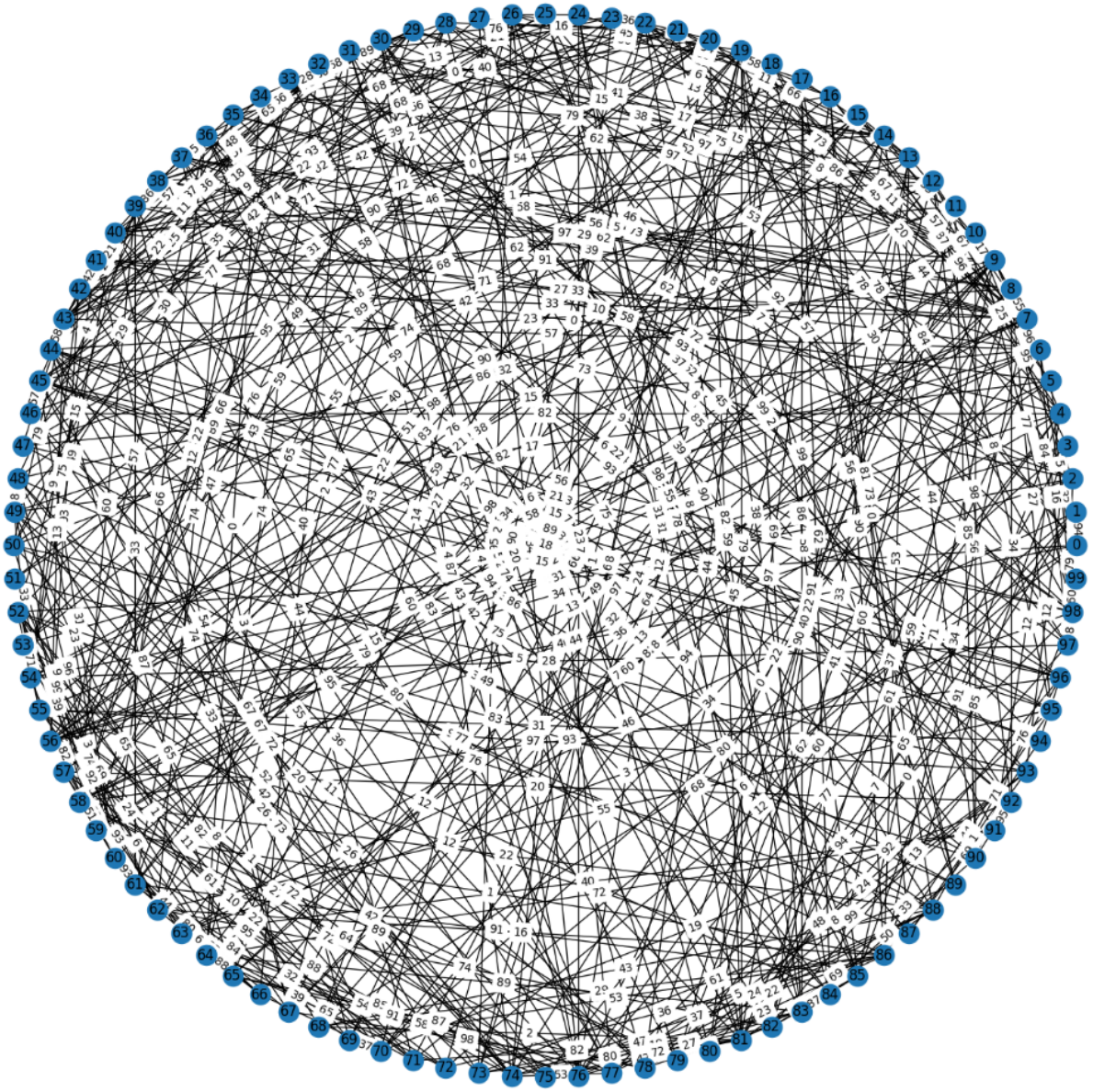


Figure 2 - weighted undirected graph

2. About Q2: randomly generate a 10x20 cell grid

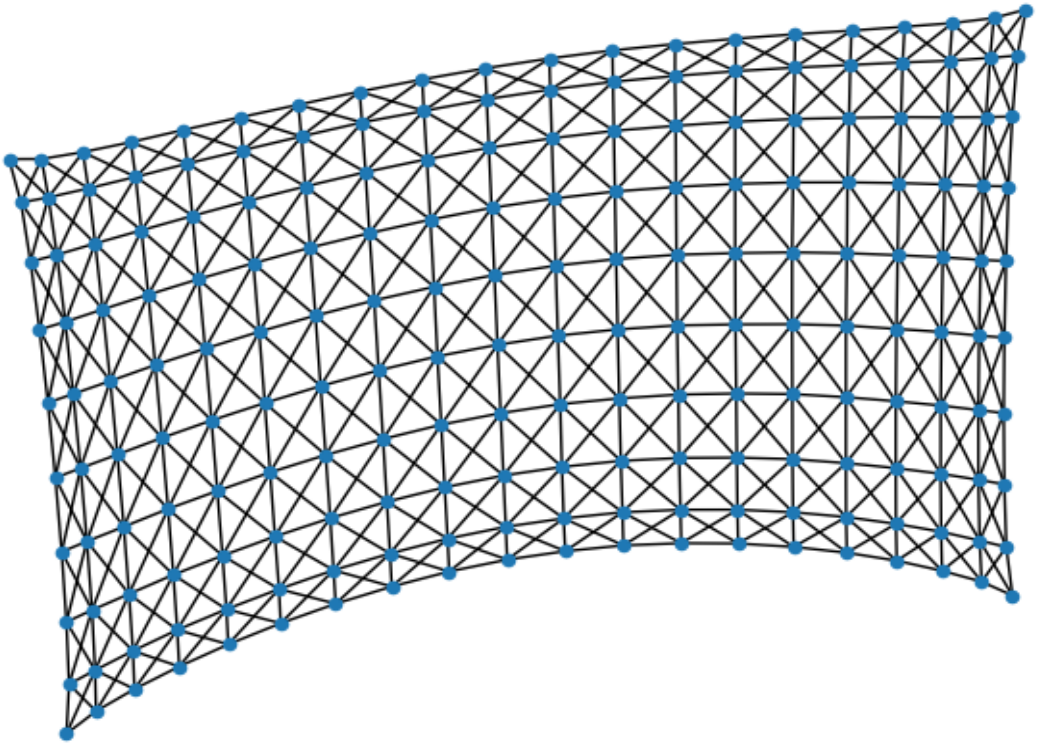


Figure 3 - Grid graph before generating obstacles

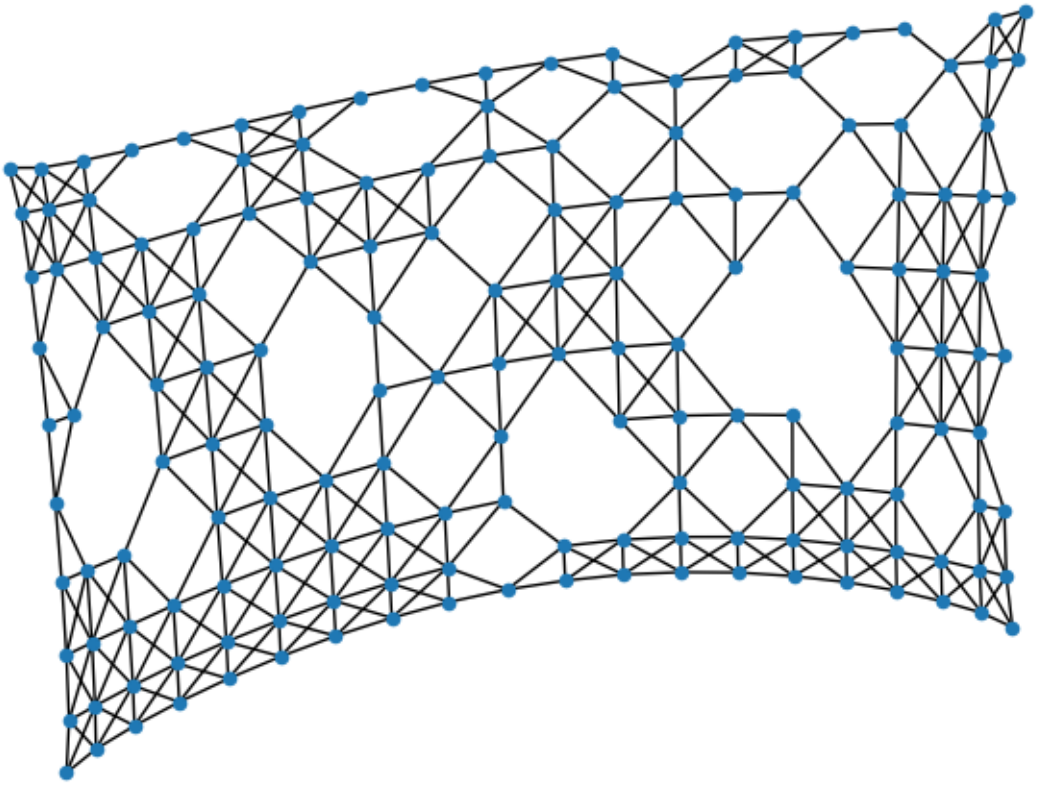


Figure 4 - Grid graph after generating obstacles

Brief theoretical part

1. Dijkstra's algorithm:

Dijkstra's algorithm is a graph traversal algorithm used to find the shortest paths from a source vertex to all other vertices in a weighted graph with positive edge weights. It works by maintaining two sets: one set keeps track of vertices included in the shortest path tree (SPT), and the other set includes vertices not yet included in SPT. At each step, it selects the vertex with the minimum distance from the source that is not in the SPT set and includes it in the SPT. It then updates the distances to adjacent vertices if a shorter path is found. The algorithm continues until all vertices are included in the SPT, resulting in the shortest paths from the source to all other vertices.

2. A* algorithm

A* algorithm is a graph search algorithm used to find the shortest path from a source vertex to a target vertex in a weighted graph. It employs a combination of real path cost and a heuristic estimate of the cost to the target to guide its search. The algorithm selects the most promising paths first based on their estimated total cost, ensuring efficient exploration of potential solutions.

3. Bellman-Ford algorithm

The Bellman-Ford algorithm is used to find the shortest paths from a source vertex to all other vertices in a weighted graph, which may include negative edge weights. It works by iteratively relaxing edges and updating the shortest path distances until the optimal solution is achieved or a negative weight cycle is detected.

Result:

I. Question 1

Find the shortest path from 13-55 using Dijkstra Algorithm and Bellman-Ford Algorithm:

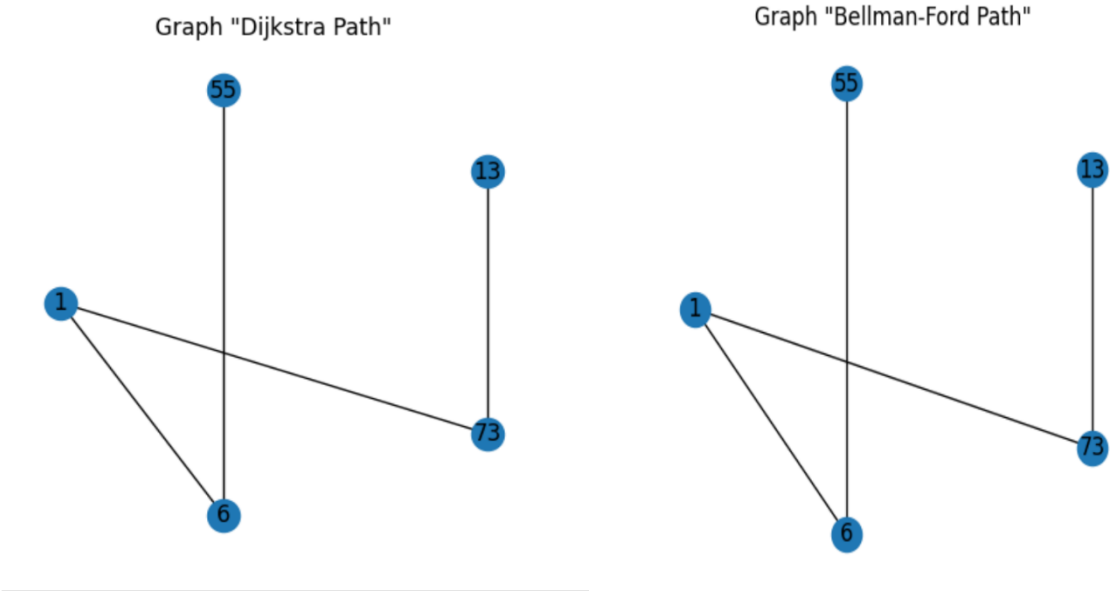


Figure 5 - First trial

Repeat the experiment 10 times for the same starting vertex and calculate the average time required for the paths search of each algorithm.

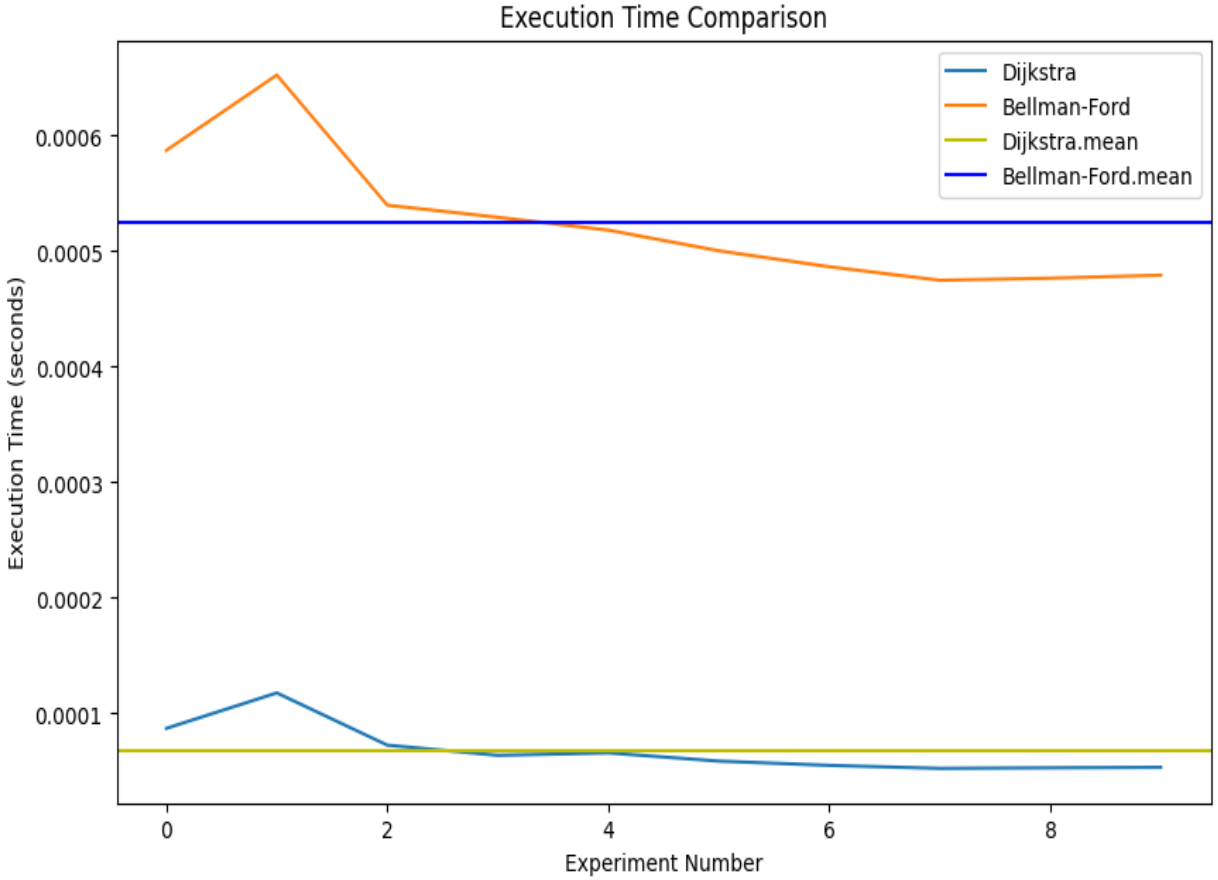


Figure 6 - Multiple trials plot

	Dijkstra_path	Bellman_Ford_path
0	0.000087	0.000587
1	0.000118	0.000652
2	0.000072	0.000540
3	0.000064	0.000529
4	0.000066	0.000518
5	0.000059	0.000500
6	0.000055	0.000486
7	0.000052	0.000475
8	0.000053	0.000477
9	0.000053	0.000479

Figure 7- Multiple trials table

```
Dijkstra_path        0.000068
Bellman_Ford_path    0.000524
```

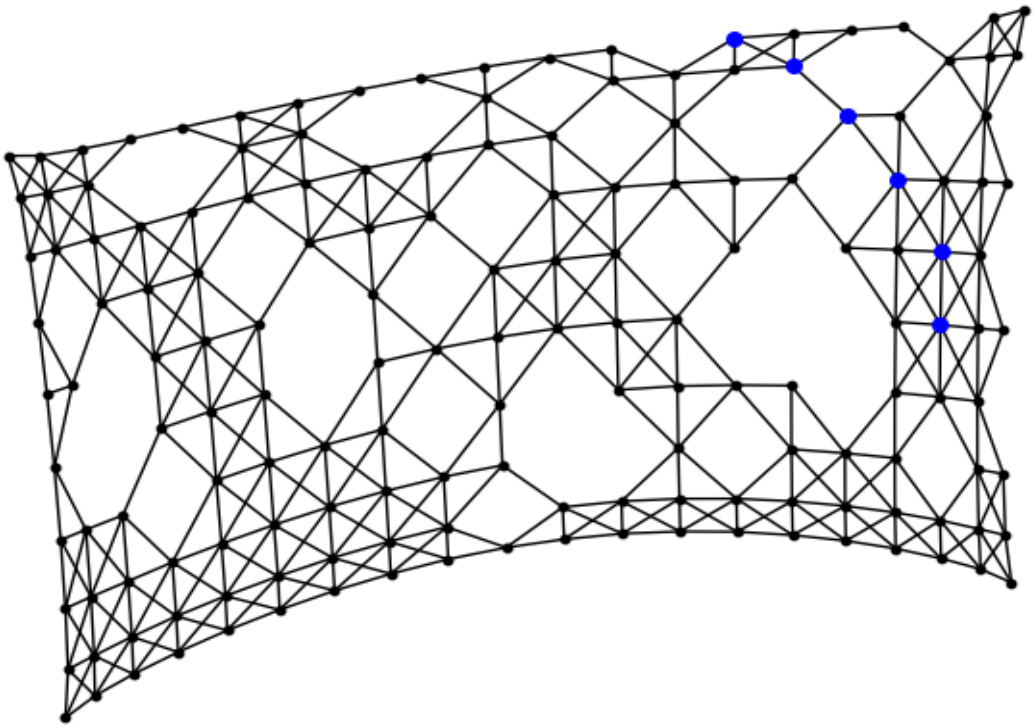
Figure 8- Multiple trials mean

Conclusion :

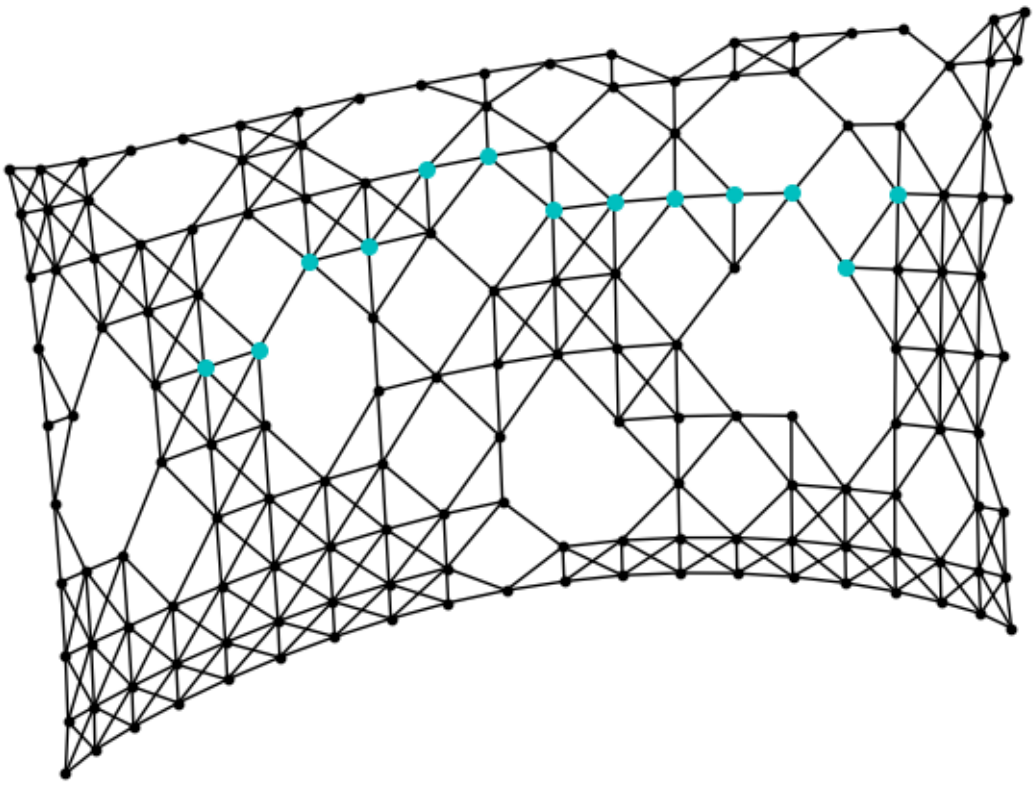
Both speed and average bellman-ford time are inferior than Dijkstras, but compared to Dijkstras algorithm bellman-ford can handle negative weights

II. 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.

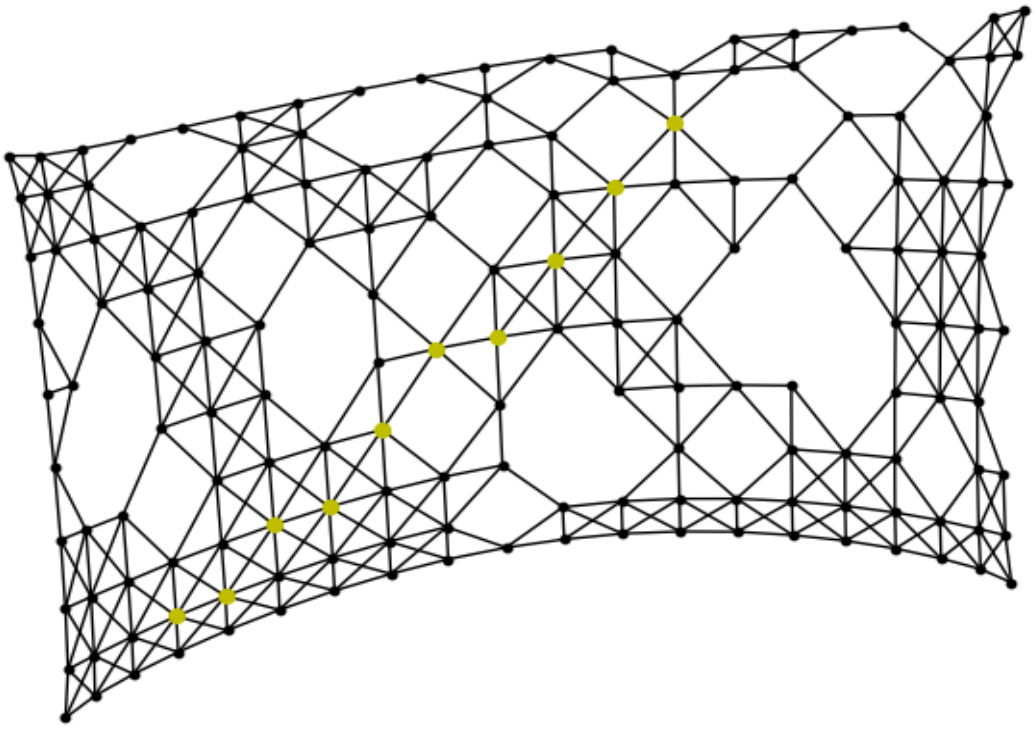
experiment1:



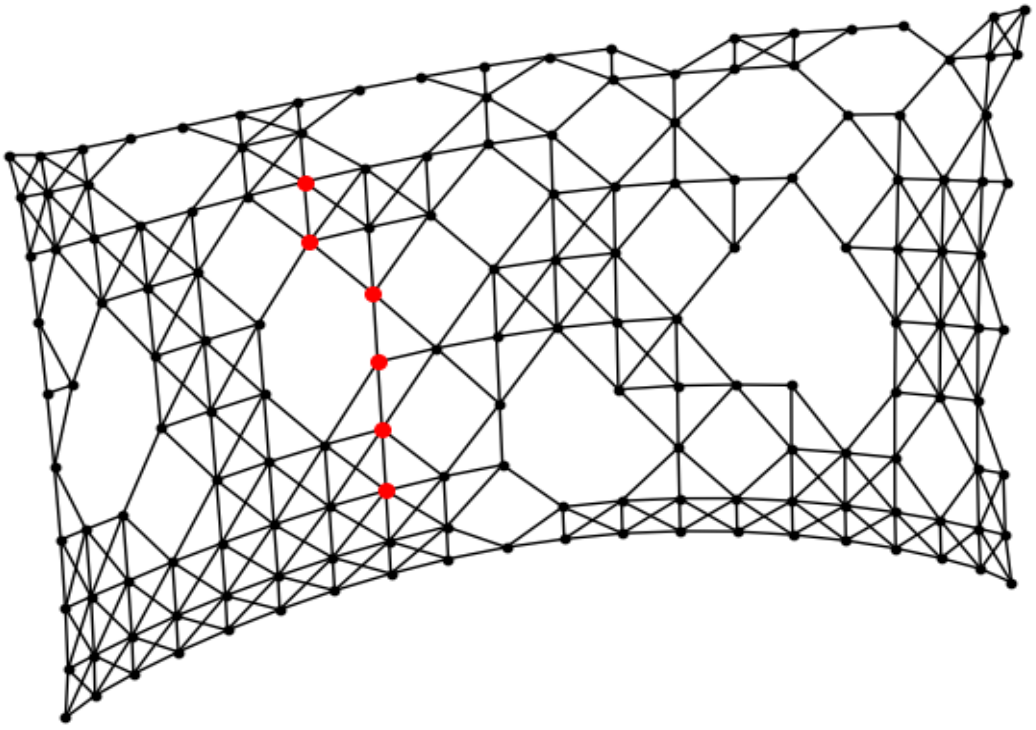
experiment2:



experiment3:



experiment4:



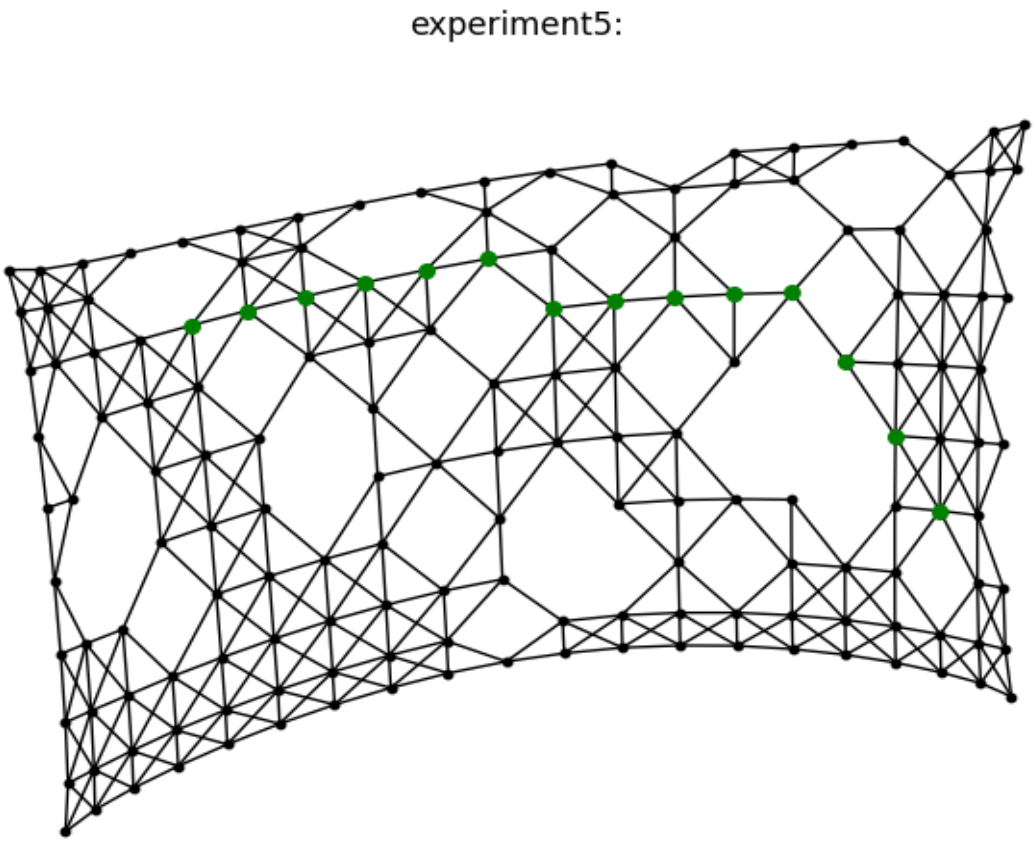


Figure 9-14 - Multiple trials figures

name	start node	goal node	shortest path	Chebyshev distance
experiment 1	(0, 9)	(2, 16)	[(0, 9), (1, 10), (1, 11), (1, 12), (1, 13), (2, 14), (3, 15), (2, 16)]	7
experiment 2	(1, 7)	(3, 9)	[(1, 7), (2, 8), (3, 9)]	2
experiment 3	(0, 18)	(1, 5)	[(0, 18), (0, 17), (0, 16), (0, 15), (0, 14), (0, 13), (0, 12), (0, 11), (1, 10), (1, 9), (1, 8), (1, 7), (1, 6), (1, 5)]	13
experiment 4	(8, 15)	(7, 18)	[(8, 15), (7, 16), (6, 17), (7, 18)]	3
experiment 5	(3, 11)	(1, 9)	[(3, 11), (2, 11), (1, 10), (1, 9)]	2

1. The average path is 6.6 and the average Chebyshev distance is 5.4
2. For each experiment: experiment 1: Detour 1 point
   experiment 1: Detour 1 point
   experiment 1: Detour 1 point
   experiment 1: Detour 1 point
   experiment 1: Detour 2 point

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

Algorithm name	Dijkstra's algorithm	A* algorithm	Bellman-Ford algorithm
data structure	priority queue and distance array	list and heuristic Function	distance array and edge list
design skills	greedy strategy	heuristic search	dynamic programming

Conclusions

In this task, we generate a random adjacency matrix for a simple undirected weighted graph 100 vertices and 500 edges. We then transform this matrix into a graph. Use Dijkstra's algorithm and Bellman-Ford algorithm to find the shortest distance Finally Analyze the running time of the two algorithms and compare them with drawings

Regarding question 2, make it possible to move the grid from 8 directions and set up 40 obstacles. Solve the minimum distance for 5 different groups of random variations, use Chebyshev distance to compare with the shortest path, and check the average detour point

In conclusion:1: 1. Dijkstra's algorithm runs faster than the Bellman-Ford algorithm; the Bellman-Ford algorithm has better applicability;

2. In the second question, a total of 6 points were detoured in 5 trials.

In summary, ask 6 aims to study the performance of different path search algorithms on different types of graphs and analyze the experimental results in order to understand the efficiency and applicability of the algorithms.