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
Carlos Andres Daza Rachen

Academic group
Accepted by
Dr Petr Chunaev

St. Petersburg
2022

Goal

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

Formulation of the problem

Generate a random adjacency matrix, apply Dijkstra's and Bellman-Ford algorithms on it

Brief theoretical part

Dijkstra's algorithm (named after its discover, E.W. Dijkstra) solves the problem of finding the shortest path from a point in a graph (the source) to a destination. It turns out that one can find the shortest paths from a given source to all points in a graph in the same time, hence this problem is sometimes called the single-source shortest paths.

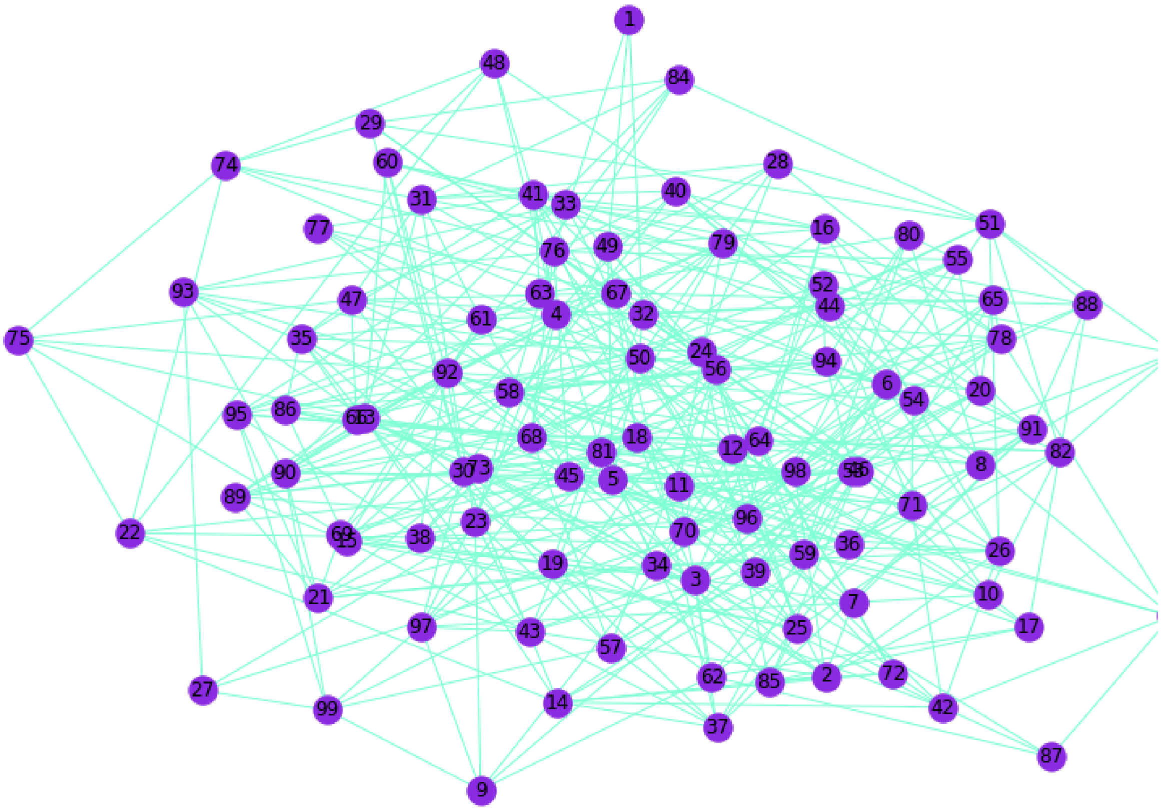
The **Bellman–Ford algorithm** is an algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted digraph[1].

A* is a graph traversal and path search algorithm, One major practical drawback is its space complexity, as it stores all generated nodes in memory. However, A* is still the best solution in many cases[4].

problem

Results

Graph



Adjacency matrix

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	11	0	0	72	0	0	0	2
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	57	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	11	0	0	57	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	72	0	0	0	0	0	0	0	0	0	60	0	0
11	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	60	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	44
14	0	0	2	0	0	0	0	0	0	0	0	0	0	44	0

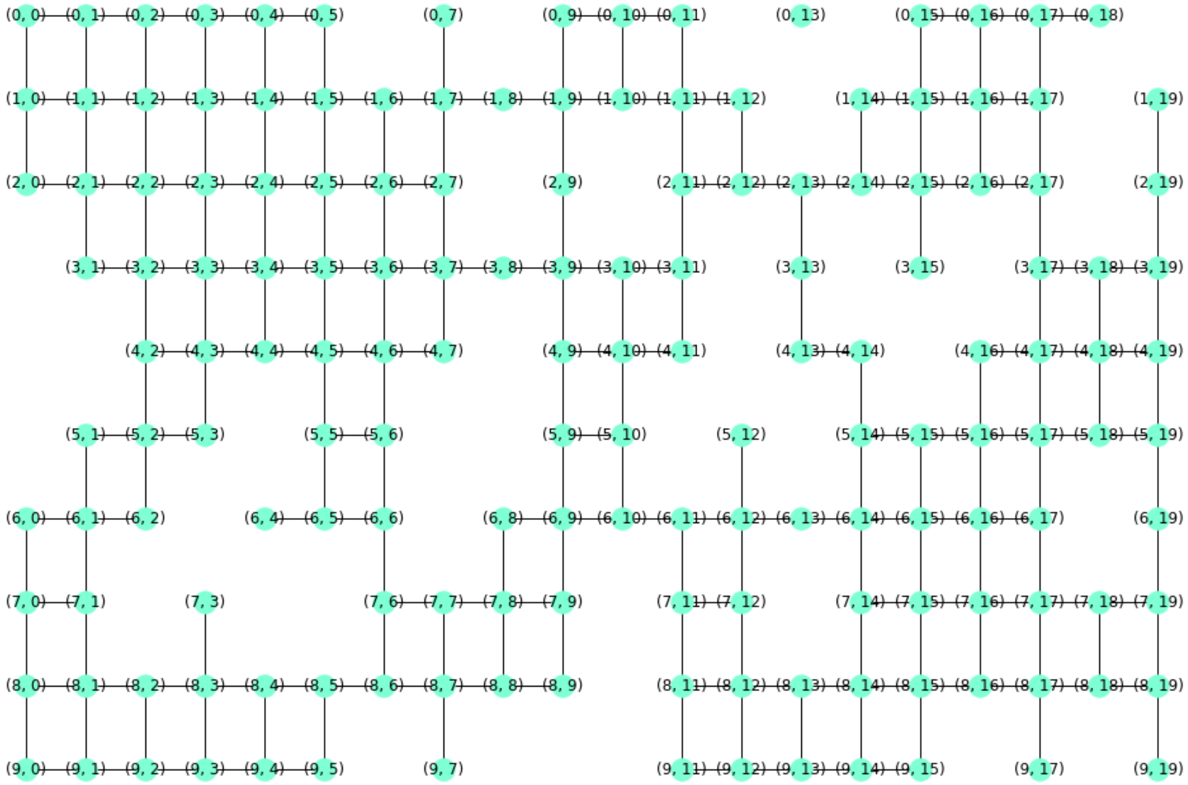
Dijkstra

```
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
Dijkstra's avg time: 0.0005995988845825196
```

Bellman - Ford

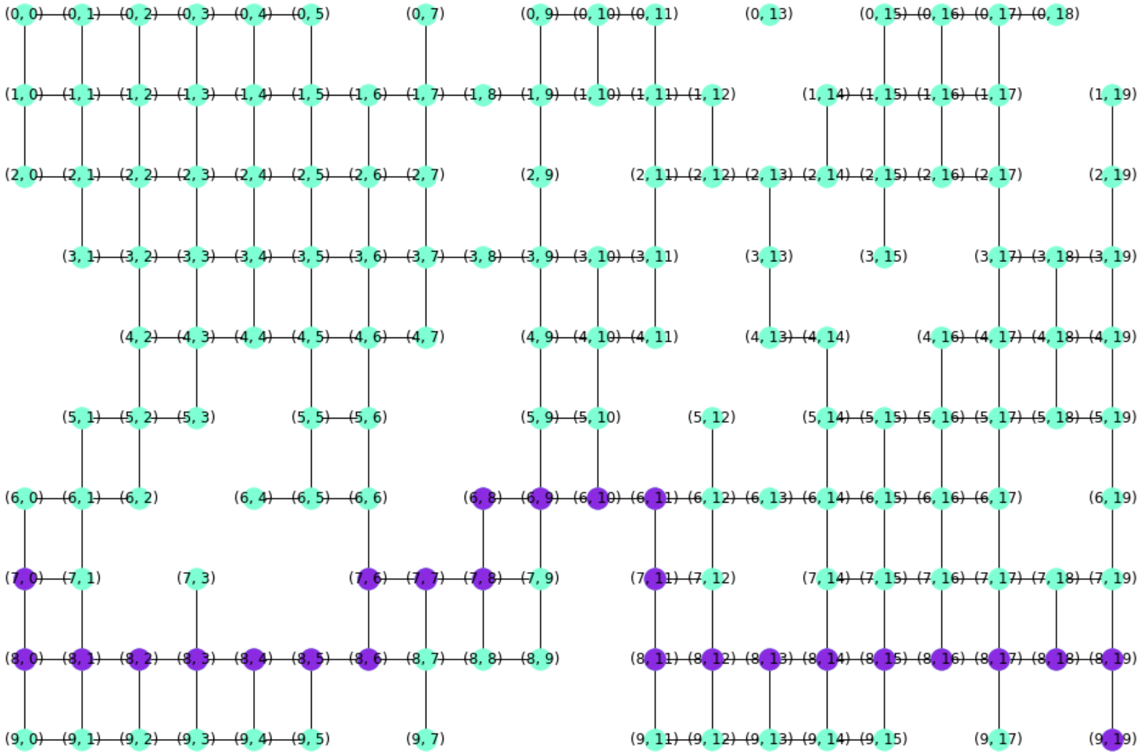
```
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
[5, 38, 26, 89, 50, 95, 8, 58]
Bellman Ford's avg time: 0.002998089790344238
```

Nodes

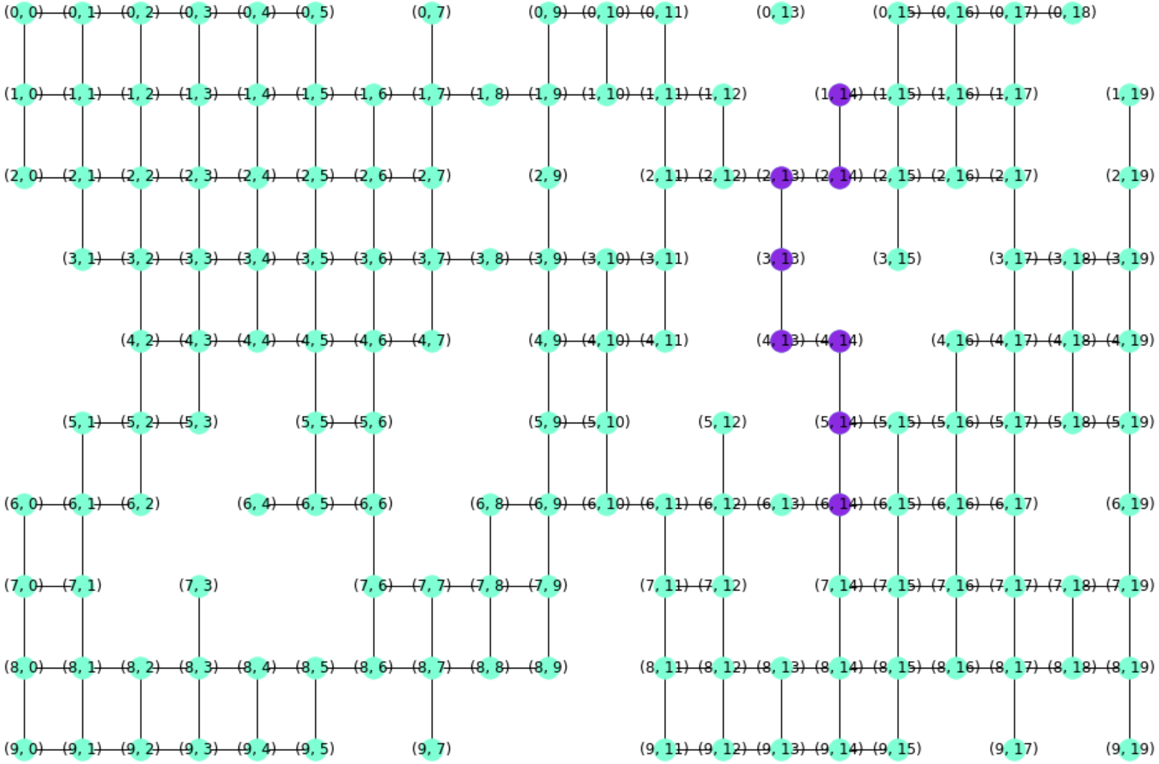


A*

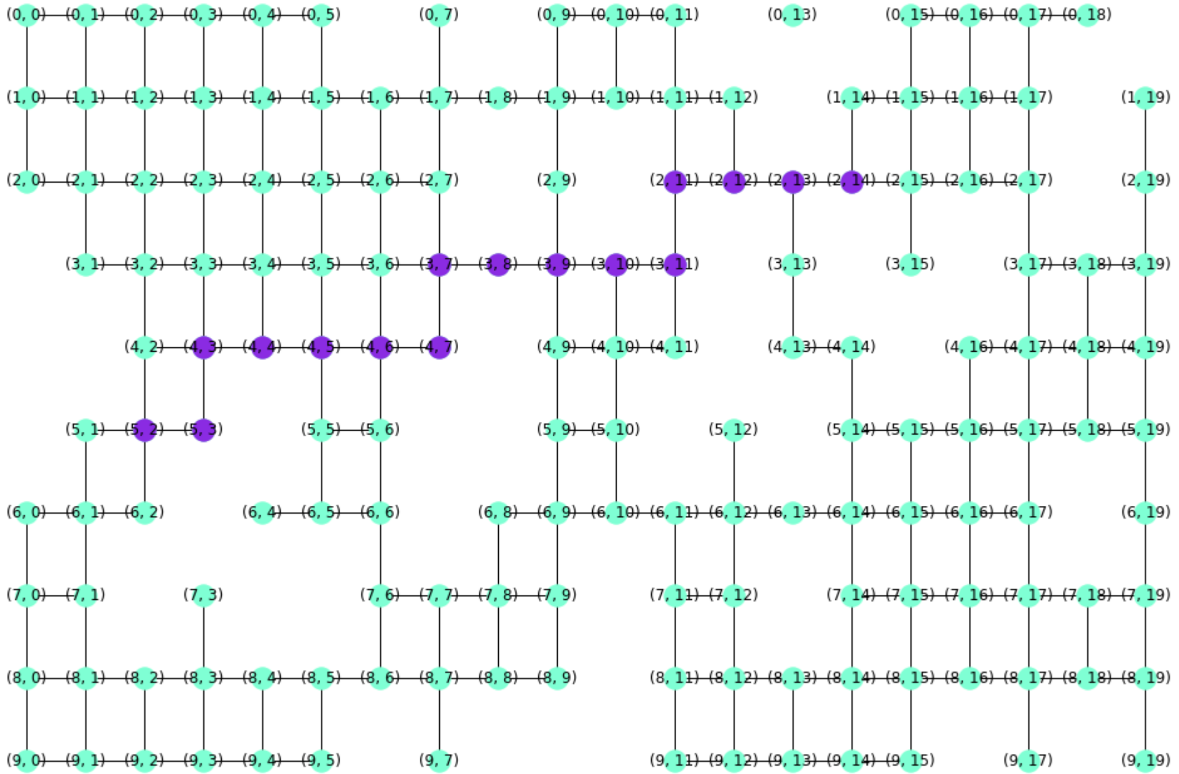
7,0 - 9,19



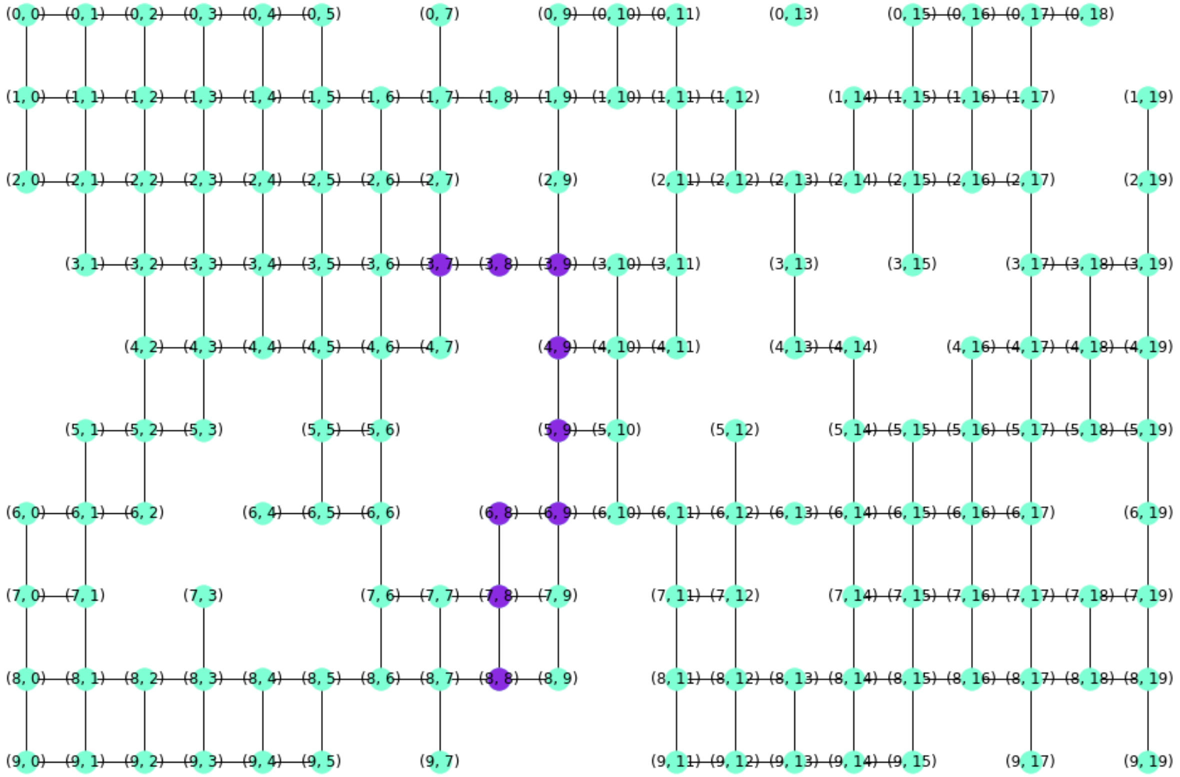
1,14 - 6,14



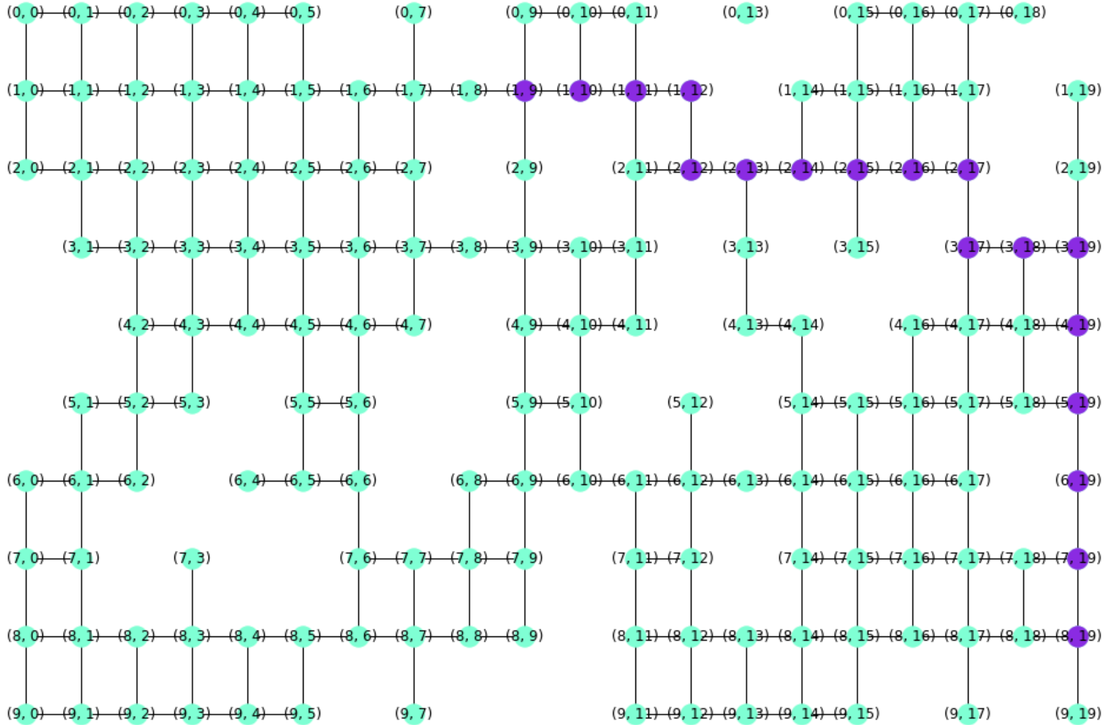
5,2 - 2,14


(0, 0)
(0, 1)
(0, 2)
(0, 3)
(0, 4)
(0, 5)
(0, 7)
(0, 9)
(0, 10)
(0, 11)
(0, 13)
(0, 15)
(0, 16)
(0, 17)
(0, 18)
(1, 0)
(1, 1)
(1, 2)
(1, 3)
(1, 4)
(1, 5)
(1, 6)
(1, 7)
(1, 8)
(1, 9)
(1, 10)
(1, 11)
(1, 12)
(1, 14)
(1, 15)
(1, 16)
(1, 17)
(1, 19)
(2, 0)
(2, 1)
(2, 2)
(2, 3)
(2, 4)
(2, 5)
(2, 6)
(2, 7)
(2, 9)
(2, 11)
(2, 12)
(2, 13)
(2, 14)
(2, 15)
(2, 16)
(2, 17)
(2, 19)
(3, 1)
(3, 2)
(3, 3)
(3, 4)
(3, 5)
(3, 6)
(3, 7)
(3, 8)
(3, 9)
(3, 10)
(3, 11)
(3, 13)
(3, 15)
(3, 17)
(3, 18)
(3, 19)
(4, 2)
(4, 3)
(4, 4)
(4, 5)
(4, 6)
(4, 7)
(4, 9)
(4, 10)
(4, 11)
(4, 13)
(4, 14)
(4, 16)
(4, 17)
(4, 18)
(4, 19)
(5, 1)
(5, 2)
(5, 3)
(5, 5)
(5, 6)
(5, 9)
(5, 10)
(5, 12)
(5, 14)
(5, 15)
(5, 16)
(5, 17)
(5, 18)
(5, 19)
(6, 0)
(6, 1)
(6, 2)
(6, 4)
(6, 5)
(6, 6)
(6, 8)
(6, 9)
(6, 10)
(6, 11)
(6, 12)
(6, 13)
(6, 14)
(6, 15)
(6, 16)
(6, 17)
(6, 19)
(7, 0)
(7, 1)
(7, 3)
(7, 6)
(7, 7)
(7, 8)
(7, 9)
(7, 11)
(7, 12)
(7, 14)
(7, 15)
(7, 16)
(7, 17)
(7, 18)
(7, 19)
(8, 0)
(8, 1)
(8, 2)
(8, 3)
(8, 4)
(8, 5)
(8, 6)
(8, 7)
(8, 8)
(8, 9)
(8, 11)
(8, 12)
(8, 13)
(8, 14)
(8, 15)
(8, 16)
(8, 17)
(8, 18)
(8, 19)
(9, 0)
(9, 1)
(9, 2)
(9, 3)
(9, 4)
(9, 5)
(9, 7)
(9, 11)
(9, 12)
(9, 13)
(9, 14)
(9, 15)
(9, 17)
(9, 19)

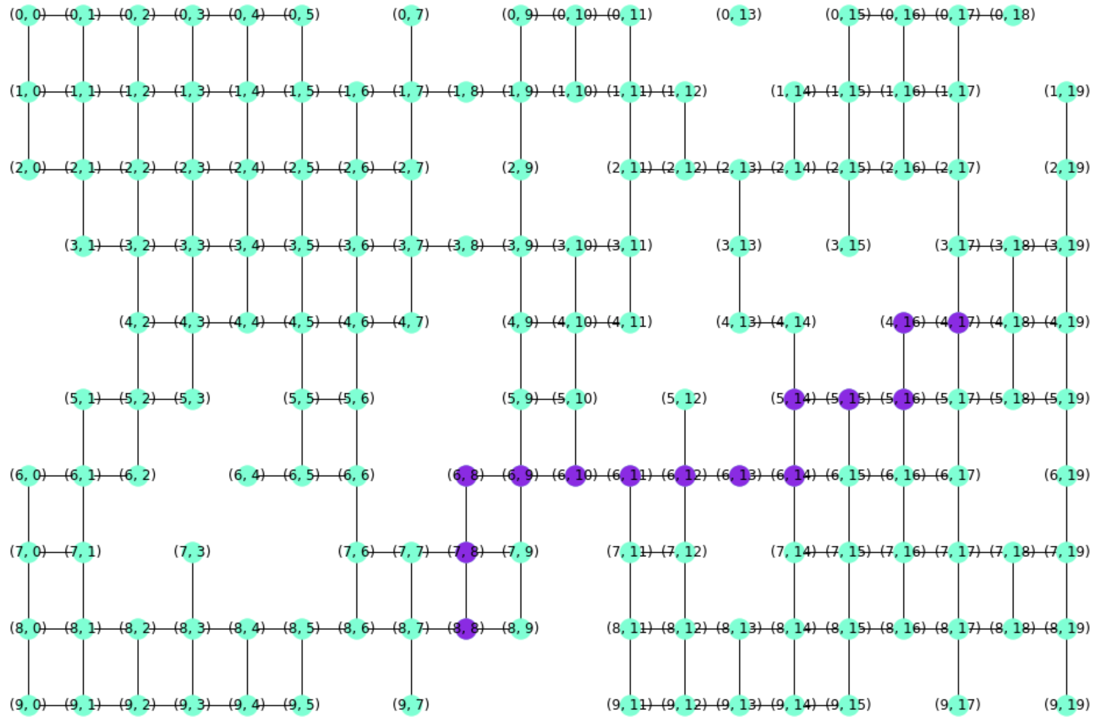
3,7 - 8,8


(0, 0)
(0, 1)
(0, 2)
(0, 3)
(0, 4)
(0, 5)
(0, 7)
(0, 9)
(0, 10)
(0, 11)
(0, 13)
(0, 15)
(0, 16)
(0, 17)
(0, 18)
(1, 0)
(1, 1)
(1, 2)
(1, 3)
(1, 4)
(1, 5)
(1, 6)
(1, 7)
(1, 8)
(1, 9)
(1, 10)
(1, 11)
(1, 12)
(1, 14)
(1, 15)
(1, 16)
(1, 17)
(1, 19)
(2, 0)
(2, 1)
(2, 2)
(2, 3)
(2, 4)
(2, 5)
(2, 6)
(2, 7)
(2, 9)
(2, 11)
(2, 12)
(2, 13)
(2, 14)
(2, 15)
(2, 16)
(2, 17)
(2, 19)
(3, 1)
(3, 2)
(3, 3)
(3, 4)
(3, 5)
(3, 6)
(3, 7)
(3, 8)
(3, 9)
(3, 10)
(3, 11)
(3, 13)
(3, 15)
(3, 17)
(3, 18)
(3, 19)
(4, 2)
(4, 3)
(4, 4)
(4, 5)
(4, 6)
(4, 7)
(4, 9)
(4, 10)
(4, 11)
(4, 13)
(4, 14)
(4, 16)
(4, 17)
(4, 18)
(4, 19)
(5, 1)
(5, 2)
(5, 3)
(5, 5)
(5, 6)
(5, 9)
(5, 10)
(5, 12)
(5, 14)
(5, 15)
(5, 16)
(5, 17)
(5, 18)
(5, 19)
(6, 0)
(6, 1)
(6, 2)
(6, 4)
(6, 5)
(6, 6)
(6, 8)
(6, 9)
(6, 10)
(6, 11)
(6, 12)
(6, 13)
(6, 14)
(6, 15)
(6, 16)
(6, 17)
(6, 19)
(7, 0)
(7, 1)
(7, 3)
(7, 6)
(7, 7)
(7, 8)
(7, 9)
(7, 11)
(7, 12)
(7, 14)
(7, 15)
(7, 16)
(7, 17)
(7, 18)
(7, 19)
(8, 0)
(8, 1)
(8, 2)
(8, 3)
(8, 4)
(8, 5)
(8, 6)
(8, 7)
(8, 8)
(8, 9)
(8, 11)
(8, 12)
(8, 13)
(8, 14)
(8, 15)
(8, 16)
(8, 17)
(8, 18)
(8, 19)
(9, 0)
(9, 1)
(9, 2)
(9, 3)
(9, 4)
(9, 5)
(9, 7)
(9, 11)
(9, 12)
(9, 13)
(9, 14)
(9, 15)
(9, 17)
(9, 19)

2,9 - 8,19



8,8 - 4,17



```
A*'s avg time: 0.000595855712890625
```

Data Structures [2]

Array: Is a structure of fixed-size, which can hold items of the same data type. It can be an array of integers, an array of floating-point numbers, an array of strings or even an array of arrays (such as 2-dimensional arrays).

Matrix: Matrices are also known as multidimensional arrays. The dimension of a matrix is m-by-n, where m is the number of rows and n is the number of columns.

Graph: A graph consists of a finite set of vertices or nodes and a set of edges connecting these vertices. The order of a graph is the number of vertices in the graph. The size of a graph is the number of edges in the graph. Two nodes are said to be adjacent if they are connected to each other by the same edge.

Grid: Is an array that can have any shape. Grid elements need not be contiguous i.e. grids can have holes in them. For example, grids can be trapezoidal, parabolic, rectangular with a hole, or pyramid-like[3].

Priority queue is an abstract data-type similar to a regular queue or stack data structure in which each element additionally has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority.

Design techniques

Loop: A loop in a computer program is an instruction that repeats until a specified condition is reached. In a loop structure, the loop asks a question. If the answer requires action, it is executed. The same question is asked again and again until no further action is required. Each time the question is asked is called an iteration [3].

Inheritance: is the mechanism of basing an object or class upon another object (prototype-based inheritance) or class (class-based inheritance), retaining similar implementation

Combinatorial optimization is a subfield of mathematical optimization that consists of finding an optimal object from a finite set of objects, where the set of feasible solutions is discrete or can be reduced to a discrete set.

Conclusions

- One of the main difference between Dijkstra's algorithm and Bellman–Ford is that the first is not suitable for graph with negative edges, for that Bellman-Ford is the best option
- One disadvantage of Bellman-Ford is that it doesn't scale well so in perfomance Dijsktra works better, but it has a lot benneftis, that makes it so widelly used
- A* deals with obstacles between nodes eficiently, but it's complexity makes it demanding

Bibliography

[1] Bang-Jensen, Jørgen; Gutin, Gregory (2000).Theory, Algorithms and Applications
[2] 8 Common Data Structures every Programmer must know(2021), https://towardsdatascience.com/
[3]Narain Gehani(1997), A new data structure — the grid
[4]Liang Zhao(2000), A* Algorithm for the time-dependent shortest path problem