Fall 2017
ECE 608 Assignment
11/21/17

Songrui Li

li884@purdue.edu

0025338817

Introduction

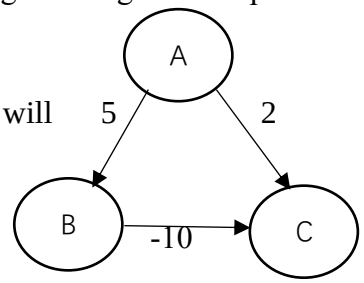
In this assignment, the single-source shortest path problem is selected and studied. The Dijkstra's algorithm and the Bellman-Ford algorithm are chosen to solve the selected problem. The following report is based on the selected problem and algorithms.

Part 1

When given a graph, two aspects should be considered to decide whether Dijkstra or Bellman-Ford applies.

1) Applicability

Dijkstra algorithm cannot deal with the graphs with negative edges. A simple example is given to illustrate this statement.



When Dijkstra is applied to the example graph, vertex will choose C first. The shortest path from A to C according to Dijkstra is A → C which is 2.

However, the correct shortest path should be A→B→C which is -5. So, Dijkstra is not applicable to graphs with negative edges. On the other hand, Bellman-ford can detect negative edges and find the correct shortest path no matter the graphs have negative edges or not.

In conclusion, the Bellman-ford algorithm should be selected as long as the given graphs have negative edges.

2) Efficiency

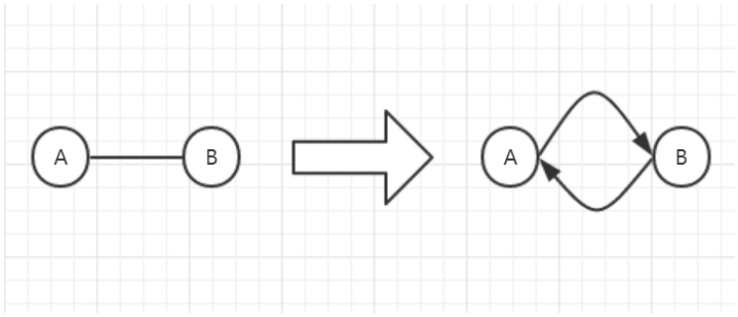
The time complexity of Dijkstra algorithm is $O(V^2)$, and Bellman-ford is O(EV), where E is the number of graph's edges, V is the number of graph's vertices.

According to the time complexity, we can tell that the factors which could affect the efficiency of Dijkstra or Bellman-ford algorithm are the graph's scale (number of

vertices) and the graph's density (number of edges). Directed or undirected graphs can also be considered. Therefore, an experiment is designed to test the performance of Dijkstra and Bellman-ford algorithms under different conditions mentioned above.

Experiment and Procedure

Actually, undirected graphs can be treated as special directed graphs.



As shown in this figure, the edge (A, B) in the undirected graphs is equivalent to the two edges (A, B) and (B, A) in the directed graphs. Therefore, the result we get from directed graphs can also apply to undirected graphs.

The procedures of the experiment design as the following:

a. Generate a random graph without negative edges.

First, determine the graph's scale which is the number of vertices.

Second, determine the graph's density. Use average degree of each vertex to indicate the density, and calculate edge number of the graph by average degree. Therefore, we have the following equation:

Edge number = vertex number * average degree / 2.

(We defined average degree = |E|*2/ |V|. Because in the actual algorithm, an undirected edge is equal to two directed edges. Therefore, my result and analysis is working properly no matter it is an undirected graph or a directed graph.)

With the vertex and edge number determined, repeatedly choose two vertices from the graph randomly and connect them. And then assign random numbers between 1 and 10 to the edges' weights.

b. Apply the Dijkstra and Bellman-ford algorithm to this randomly generated graph separately. Collect their run time separately.

c. Repeat procedure a) and b) n times. Calculate the average run time of Dijkstra algorithm and that of Bellman-ford algorithm after n times run. The algorithm which has a smaller average run time is more efficient when applied to this type of graphs.

After implementing a program according to the above experiment, the following result graphs are obtained:

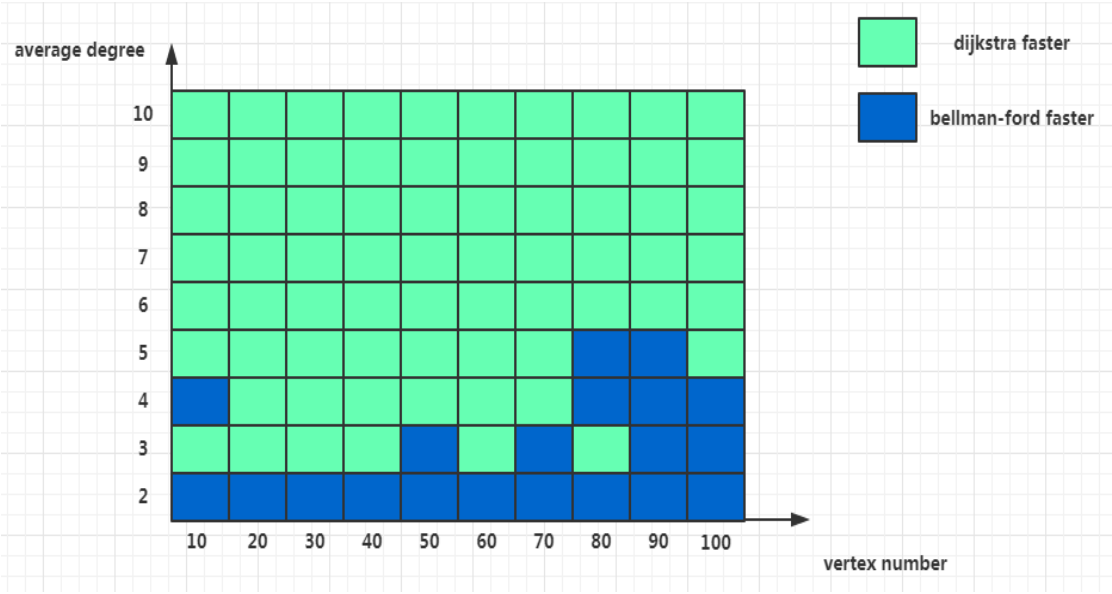


Figure 1

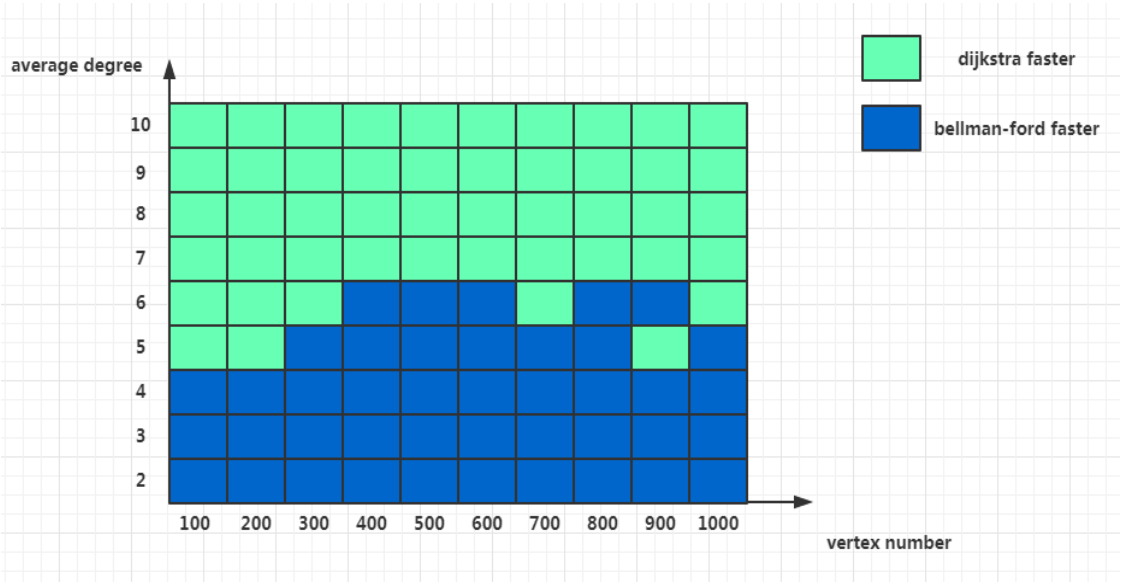
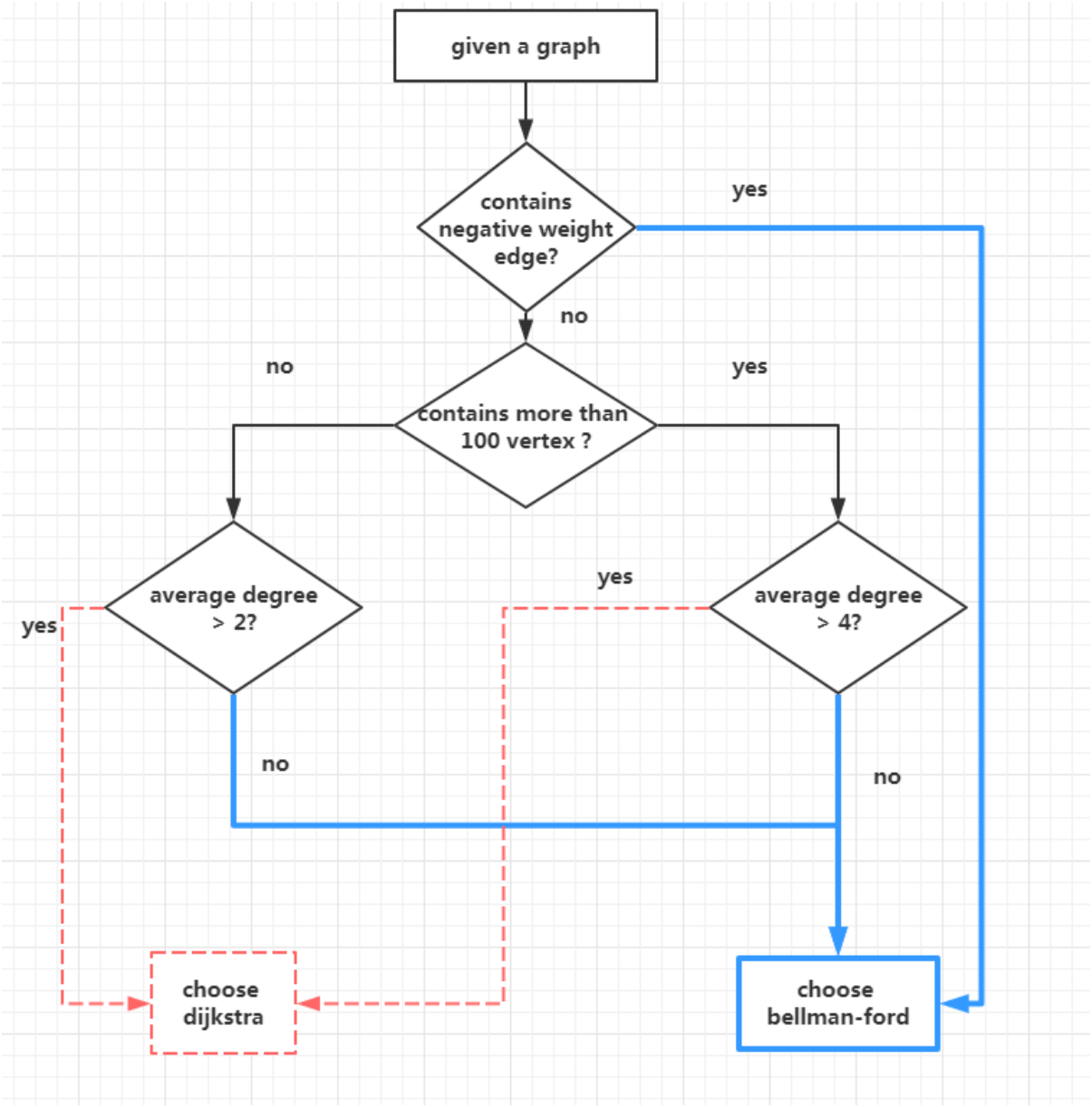


Figure 2

From the above two figures we can tell: When the average degree is low which means edge number is small, the Bellman-ford algorithm is better. When the average degree is high which means edge number is large, the Dijkstra algorithm is more efficient. Moreover, with the vertex number being larger, the effect of the average degree becomes more obvious.

Algorithm

According to the experimental results and analysis, an algorithm is designed to decide which shortest path algorithm should be applied when given a graph.



Pseudo code

Given a graph, with vertex set V and edge set E:

```
if contains negative weighted edge:
        use bellman-ford
else:
        average_degree = |E|*2/|V|
        if |V| < 100:
            if average_degree > 2:
                use dijkstra
                else:
                    use bellman-ford
             else:
                if average_degree > 4:
                    use dijkstra
                else:
                    use bellman-ford
```

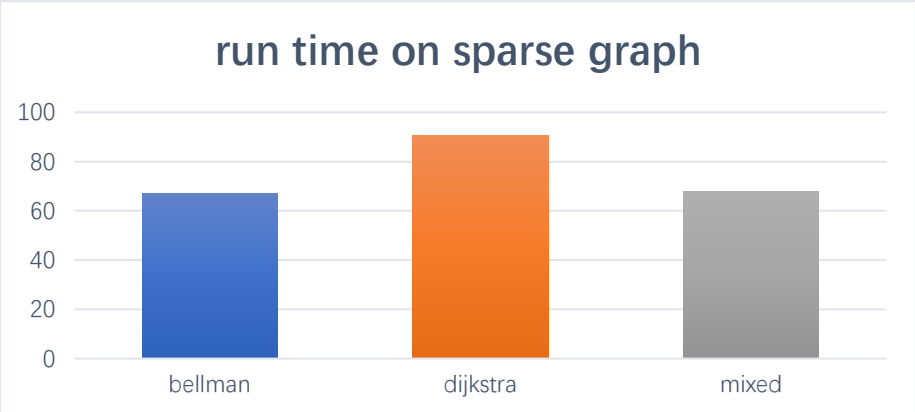
Part 2

To prove that the choosing algorithm which I designed is effective, several tests are done in this part to evaluate the results.

We only consider the non-negative edge undirected graphs. The following two tests are based on vertex number and average degree.

1) In 300 seconds, randomly generate a graph which vertex number is between 100 – 1000 and average degree is between 2 – 4. Apply those three algorithms and collect the number of tests and total run time. The results are tabulated below:

Number of tests in 300s	Bellman-ford run time	Dijkstra run time	Mixed algorithm run time
723304	67.19 seconds	90.742 seconds	67.816 seconds



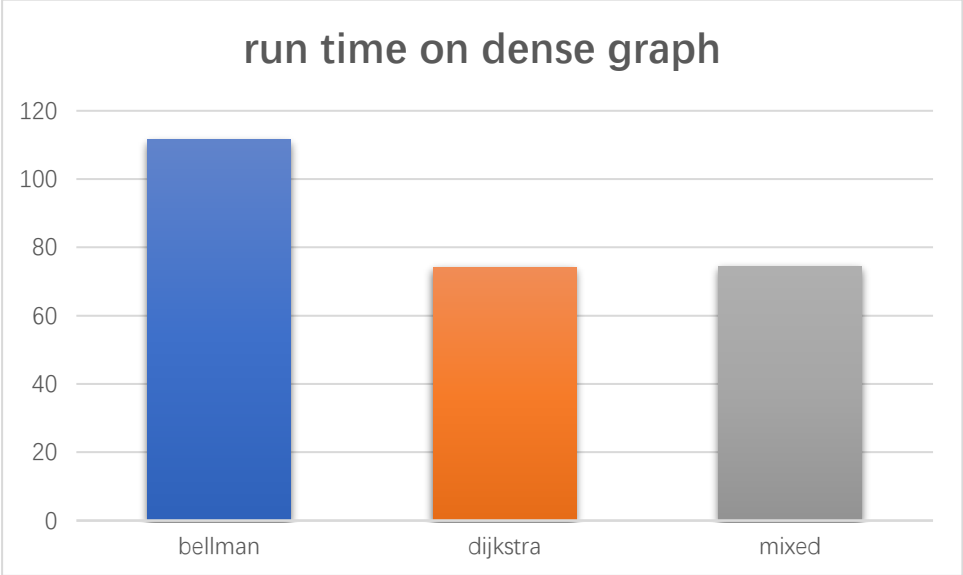
Furthermore, the result can be showed in the table form as following:

algorithm / vertex	Bellman-ford (average runtime)	Dijkstra (average runtime)	mixed (average runtime)
100	5.79e-05s	7.97e-05s	5.85e-05s
300	5.25e-04s	7.55e-04s	5.13e-04s
500	1.68e-03s	3.62e-03s	1.54e-03s
1000	5.94e-03s	1.09e-02s	5.43e-03s

From the above table we can clearly tell that when average degree is between 2-4, the average runtime of Bellman-ford and mixed algorithm is smaller than Dijkstra. It means Bellman-ford algorithm is better when average degree is low and the choosing algorithm I designed is effective.

2) In 300 seconds, randomly generate a graph which vertex number is between 100-1000 and average degree is between 8 – 10. Apply those three algorithms and collect the number of tests and total run time. The results are tabulated below:

Number of tests in 300s	Bellman-ford run time	Dijkstra run time	mixed algorithm run time
20573	111.483 seconds	74.159 seconds	74.128 seconds



Furthermore, the result can be showed in the table form as following:

algorithm / vertex	Bellman-ford (average runtime)	Dijkstra (average runtime)	mixed (average runtime)
100	1.94e-04s	1.08e-04s	1.12e-04s
300	1.69e-03s	1.11e-03s	1.13e-03s
500	4.84e-03s	2.79e-03s	2.67e-03s
1000	1.83e-02s	1.24e-02s	1.25e-02s

From the above table we can clearly tell that when average degree is between 8-10, the average runtime of Dijkstra and mixed algorithm is smaller than Bellman-ford. It means Dijkstra algorithm is better when average degree is high and the choosing algorithm I designed is effective.

Conclusion

The experiments, analysis, and tests in this report show ample evidence that given a graph whose vertex number is fixed, the Bellman-ford algorithm is more efficient if the graph's edge number is low. While Dijkstra algorithm is more advantageous in the case that the graph's edge number is large. The tables and graphs obtained in the tests in part 2 show the robustness of the choosing algorithm designed in this study which has the ability to correctly select a proper algorithm based on the features of a graph.