****

**DESIGN AND ANALYSIS OF ALGORITHMS**

**CS5592**

**PROJECT**

*(DUE DATE: 4/17/2017)*

**HARSHA SARANAM-16230289-hsgpd@mail.umkc.edu**

**CONTENTS**

***1. INTRODUCTION---------------------------------------------------------------------3***

***2. EXPERIMENTAL DESIGN----------------------------------------------------------4***

***3. EXPERIMENTAL IMPLEMENTATION-------------------------------------------6***

***4. DATA COLLECTION AND INTERPRETATION OF RESULTS-----------------9***

***5. CONCLUSION-----------------------------------------------------------------------12***

***6. EPILOGUE---------------------------------------------------------------------------13***

***7. APPENDIX--------------------------------------------------------------------------14***

***8. REFERENCES-----------------------------------------------------------------------17***

**1. INTRODUCTION**

Any sort of communications and transportations are often modeled using graphs. The communications such as message transfers, emails that can be wired, peer-to-peer or wireless over a network which reaches the destination using best way (i.e., shortest path). The transportation can be roadways, airways or other means. So, to establish a proper communication or transportation we need to predefine the source and destination for easy understanding. In this journey, the travelers may have different attributes i.e., to travel fast or to move in shortest path.

For a better understanding, we will explain with a possible example. Let us consider a person is travelling from Chicago to Kansas by road and we can state that Chicago is source and Kansas is destination. This journey has different scenarios:

* Can have multiple paths from source to destination.

*(Need to find the shortest path to reach the destination).*

* Trip can be disturbed with many vehicles along with congested roads which may delay the trip.

*(As we are dealing only with the shortest path, making the above statement as uncertain).*

**2.EXPERIMENTAL DESIGN**

The project that we developed for finding the shortest path based on different performances considered as different criterions.

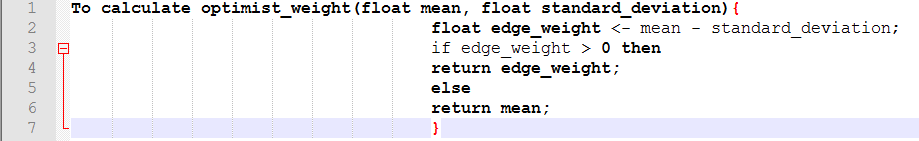
**Criteria-1(Mean Value):**

In this analysis, we are considering the ‘expected values’ as the edge lengths, based on these values, we found the shortest path to reach the destination.

*Mean value for edge****=****expected value*

**Criteria-2(Optimist):**

In this analysis, we are considering the optimist values ‘expected value-standard deviation’ as the edge lengths, later we found the shortest path from source to destination.

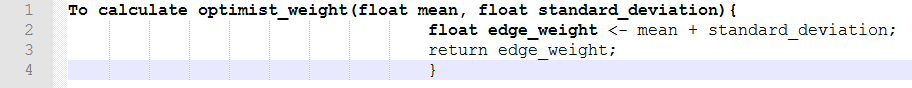


The above algorithm states the values of each edge at optimist criteria.

*Optimist value for edge=expected value-standard deviation.*

**Criteria-3(Pessimist):**

In this analysis, we are considering the pessimist values by ‘expected value + standard deviation’ as the edge lengths, later we found the shortest path from source to destination.



The above algorithm states the values of each edge at pessimist criteria.

*Pessimist value for edge=expected value + standard deviation.*

**Criteria-4(Double Pessimist):**

In this analysis, we are considering the double pessimist values by ‘expected value + 2\*standard deviation’ as the edge lengths, later we found the shortest path from source to destination.



The above algorithm states the values of each edge at double-pessimist criteria.

*Double Pessimist value for edge=expected value + (2\*standard deviation).*

**Criteria-5(Stable):**

In this analysis, we are considering the stable values by ‘squared coefficient of variation’ as the edge lengths, later we found the shortest path from source to destination.

c²= V [ X ] /E [ X ] ²

where E [X] is for the random variable on the link for edge.

  V[X] is for the random variable on the link for vertex.

*Stable value for edge= c²*

**3. EXPERIMENTAL IMPLEMENTATION**

Languages Used: JAVA

Data Structures included: Heap, Hash table.

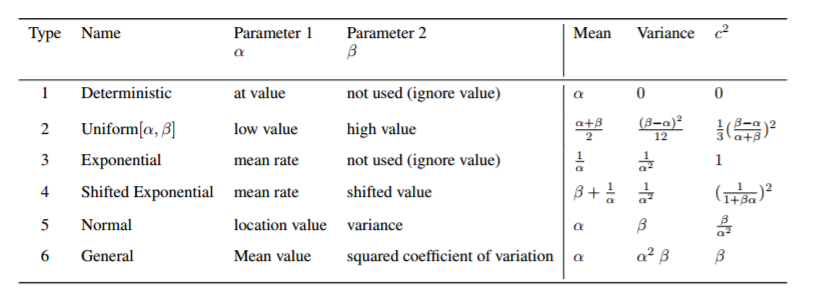
We implemented our entire project in JAVA language. As we are dealing with concepts like hash table and heap, it is easy for us to use those concepts as they are predefined. Whereas in other languages, we should implement the functions and use them.

The main intention with the project is to find the shortest path from start node to destination node along with different criteria that was given to perform. The implementation of the project goes like this:

1. Initially, we are accessing the input file where it contains the details regarding the edges (x, y, type, alpha, beta).
2. We are assigning each line into input [] array.

* If it is first line, the line is considered as no. of vertices, start node, destination node.
* If it is other than the first line, the attributes in the line are x, y, type, alpha, beta.

1. Next need to find the mean, variance and c² for every edge based on the type it is having. The table below has the different formulas for mean, variance and c²:



1. Now, after calculating the mean, variance and c² for each edge, we are assigning each edge with vertex1, vertex2, mean, variance, c². These values for each edge was pointed in the form of linked weight values.
2. After each edge assigned with (mean, variance, c²), we need to find the shortest path from start node to destination node.
3. But as per the given conditions, we developed our project for different criteria i.e., mean, optimist, pessimist, double-pessimist and stable.
4. Based on the criteria, the edges values between the vertices changes accordingly.
5. The description to find the edge weights under different criteria is given below:

* Under Mean value criteria, for each edge,

edge weight = expected value.

* Under optimistic criteria, for each edge,

edge weight = expected value – standard deviation.

* Under pessimist criteria, for each edge,

edge weight = expected value + standard deviation.

* Under double-pessimist, for each edge,

edge weight = expected value + 2\*standard deviation.

* Under stable, for each edge,

edge weight = squared coefficient of variation.

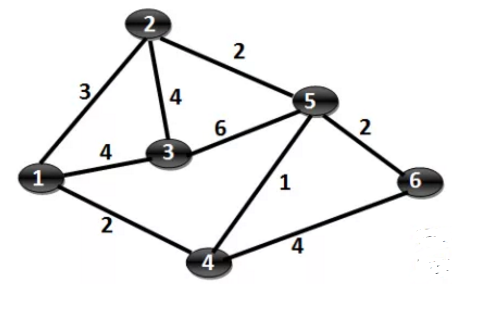
1. Once the edge values for different criteria was found, we need to find the shortest path from start node to destination node for every criterion.
2. How we implemented the shortest path is attached below:



1. For every criterion, we have printed the source, destination, shortest path length and path it is reaching the destination.
2. After these findings, we should find the hops. It can be found as the no. of edges present in the shortest path.

**4. DATA COLLECTION AND INTERPRETATION OF RESULTS**

The main theme here in the project is to find the shortest path among the set of nodes from start node to destination node. To understand in a better way, we are going to explain it through an example, that is demonstrated below:



Step-1: Mark Vertex 1 as the source vertex. Assign a cost zero to Vertex 1 and (infinite to all other vertices).

Step-2: For each of the unvisited neighbors (Vertex 2, Vertex 3 and Vertex 4) calculate the minimum cost as min(current cost of vertex under consideration, sum of cost of vertex 1 and connecting edge). Mark Vertex 1 as visited.

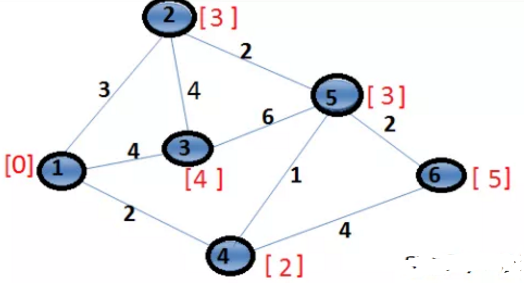
Step-3: Choose the unvisited vertex with minimum cost (vertex 4) and consider all its unvisited neighbors (Vertex 5 and Vertex 6) and calculate the minimum cost for both.

Step-4: Choose the unvisited vertex with minimum cost (vertex 2 or vertex 5, here we choose vertex 2) and consider all its unvisited neighbors (Vertex 3 and Vertex 5) and calculate the minimum cost for both. Now, the current cost of Vertex 3 is [4] and the sum of (cost of Vertex 2 + cost of edge (2,3) ) is 3 + 4 = [7]. Minimum of 4, 7 is 4. Hence the cost of vertex 3 won’t change. By the same argument, the cost of vertex 5 will not change. We just mark the vertex 2 as visited, all the costs remain same.

Step-5: Choose the unvisited vertex with minimum cost (vertex 5) and consider all its unvisited neighbors (Vertex 3 and Vertex 6) and calculate the minimum cost for both. Now, the current cost of Vertex 3 is [4] and the sum of (cost of Vertex 5 + cost of edge (5,3)) is 3 + 6 = [9]. Minimum of 4, 9 is 4. Hence the cost of vertex 3 won’t change. Now, the current cost of Vertex 6 is [6] and the sum of (cost of Vertex 5 + cost of edge (3,6)) is 3 + 2 = [5]. Minimum of 6, 5 is 45. Hence the cost of vertex 6 changes to 5.

Step-6: Choose the unvisited vertex with minimum cost (vertex 3) and consider all its unvisited neighbors (none). So, mark it visited.

Step-7: Choose the unvisited vertex with minimum cost (vertex 6) and consider all its unvisited neighbors (none). So, mark it visited.



The picture above gives the shortest path to every node from the source node i.e., 1.

1--->2 = 3

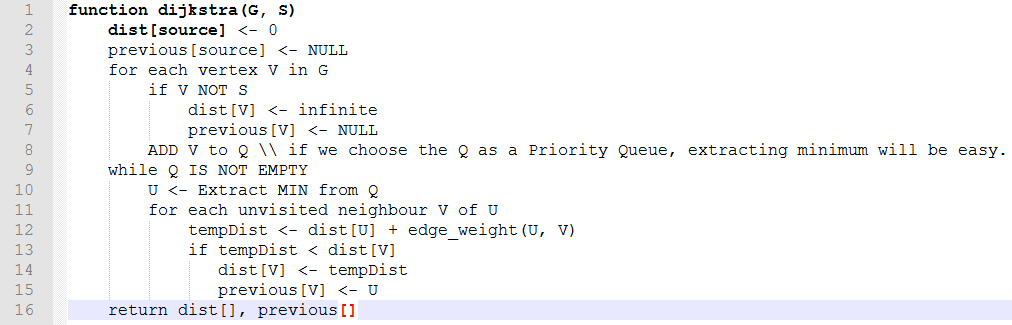
1--->3 = 4

1--->4 = 2

1--->5 = 3

1--->6 = 5

Pseudo code to find the shortest path:



**Using the functionalities that we developed, we are showing the results for each input data sets.**

**For input3.txt:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **µ − σ** | **µ** | **µ + σ** | **µ + 2σ** | **c²** | **hops** |
| **Mean** | 98.46 | 130.999 | 163.531 | 196.063 | 2.267 | 6 |
| **Optimistic** | 98.46 | 130.999 | 163.531 | 196.063 | 2.267 | 6 |
| **Pessimistic** | 140 | 141 | 142 | 143 | 0.002066 | 6 |
| **Double-Pessimistic** | 140 | 141 | 142 | 143 | 0.002066 | 6 |
| **Stable** | 140 | 141 | 142 | 143 | 0.002066 | 6 |

**For input4.txt:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **µ − σ** | **µ** | **µ + σ** | **µ + 2σ** | **c²** | **hops** |
| Mean | 88.64 | 136.999 | 185.35 | 233.705 | 5 | 6 |
| Optimistic | 88.64 | 136.999 | 185.35 | 233.705 | 5 | 6 |
| Pessimistic | 145.62 | 152.999 | 160.37 | 167.742 | 0.114 | 6 |
| Double-Pessimistic | 145.62 | 152.999 | 160.30 | 167.742 | 0.114 | 6 |
| Stable | 145.62 | 152.999 | 160.37 | 167.742 | 0.114 | 6 |

**5. CONCLUSION**

After implementing proper functionalities, we got the shortest paths for different criterions given along with the path that is carrying the shortest distance from start node to destination node.

The output values after the execution of the project having slight difference when compared with the values calculated manually, but better and consistent even after multiple executions.

In mean value criterion though the shortest path calculated is larger but the number of hops is like other nodes. The edge length formula used to calculate results in larger value of shortest distance. In optimistic value criterion, the shortest path criterion reached is less than mean value but is greater than other criterion, But the number of hops is same as previous criterion. Here all the edge lengths are lesser or equal when compared to mean value criterion. In Pessimistic value criterion, each edge length calculated is larger than mean value and optimistic value. So, the path calculated is larger value but it is due to larger edge lengths. Similarly, in Double pessimistic value also results in larger values of shortest paths than pessimistic. In stable value criterion edge lengths calculated are of shorter lengths than other values and similarly it results in shorter path value as low compared to other criterions.

The shortest paths differ for each criterion as they carry different weights from another criterion.

When it comes to the time-complexity for executing 12-nodes and 20 edges, it takes 21.5 milliseconds, which is very less. Even though the count for edges increased, the time-complexity remains in milliseconds.

**6. EPILOGUE**

The design of our approach that we implemented is totally structured. It is developed in a sequential manner. As it is sequentially developed, the output for one module becomes the input for the next module. It goes till end. If one module makes mistake, the project goes into vein. So, we dealt our project in very careful manner and making sure we are going good.

When it comes to the issues we faced, though finding distance between 2 nodes is easy, but implementing the code for finding the shortest path and getting the outputs made us difficult. After the shortest distance was found, again we fought to show the exact path that carries the shortest distance from source to destination.

We started focusing on our project from the very first day, but not so serious. As the deadline is coming, we started implementing the project which gave us mental pressure, sometimes it became messy. We came out of those situations, made proper decisions and at last we made it.

It is always better to start early, discussing with the group raises many ideas. So, we can have many options to go through and can implement effectively.

If we get an opportunity to work on this project again with additional requirements and time, we will try to implement it showing the flow in graphical view and try to do by including congestion between the nodes. That makes the project looks good by including some additional attributes.

Suggestions for future students from our experience: start early, go step-by-step, try hard, don’t lose hope.

**7. APPENDIX**

**Program Listing:**

The input is given in the format:

First Line: <<no. of vertices, start node, end node>>

All other lines:<<Edge, x, y, alpha, beta>>

12, 1, 12

E,1,2,1,24.000000,22.000000,

E,1,3,4,0.100000,13.000000,

E,2,4,1,25.000000,13.000000,

E,2,5,1,30.000000,13.000000,

E,3,4,4,0.111111,11.000000,

E,3,5,2,16.000000,28.000000,

E,4,6,4,0.100000,15.000000,

E,4,7,1,25.000000,15.000000,

E,5,6,2,19.000000,37.000000,

E,5,7,3,0.047619,9.000000,

E,6,8,5,22.000000,1.000000,

E,6,9,5,23.000000,1.000000,

E,7,8,3,0.045455,5.000000,

E,7,9,5,22.000000,1.000000,

E,8,10,2,12.000000,28.000000,

E,8,11,5,28.000000,1.000000,

E,9,10,1,22.000000,1.000000,

E,9,11,1,30.000000,1.000000,

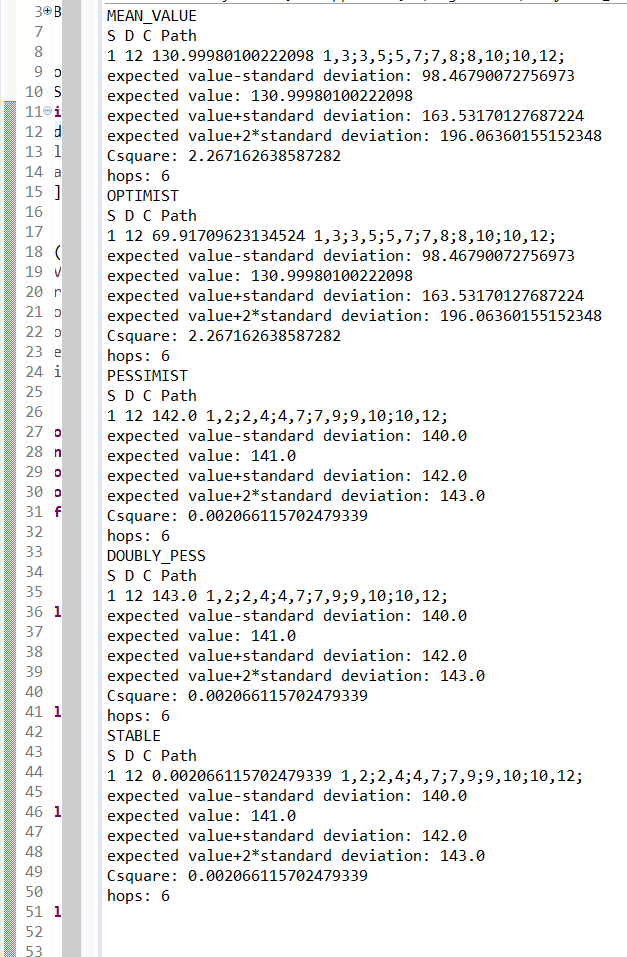
E,10,12,1,23.000000,1.000000,

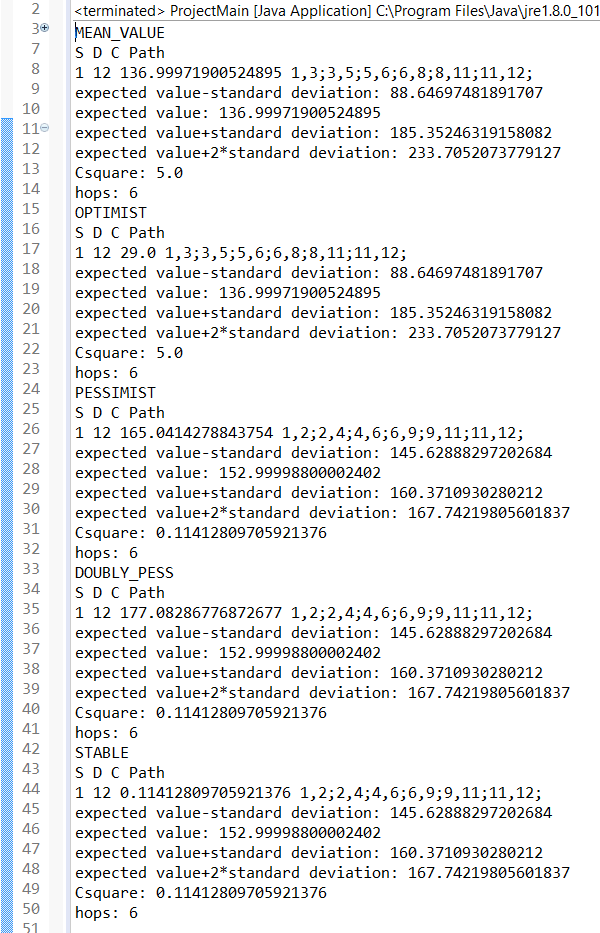
E,11,12,5,28.000000,1.000000,

Both input files are having this kind of representation. From these input files, we get the outputs for different criterions along with the required fields that we are asked to mention.

**Output:**

**Input3.txt:**



**Input4.txt:**

**8. REFERENCES**

1. <http://krishnalearnings.blogspot.com/2015/07/implementation-in-java-for-dijkstras.html>
2. <https://github.com/stewbob/dijkstra>