# CS5592 – Dijkstra's Shortest Uncertain Path Algorithm

CS5592, Spring Semester 2017: Dijkstra's Shortest Uncertain Path Algorithm

Team name: Dijkstra Coders

I understand and have adhered to the rules regarding student conduct. In particular, any and all material, including algorithms and programs, have been produced and written by myself. Any outside sources that I have consulted are free, publicly available, and have been appropriately cited. I understand that a violation of the code of conduct will result in a zero (0) for this assignment, and that the situation will be discussed and forwarded to the Academic Dean of the School for any follow up action. It could result in being expelled from the university.

| | First & Last Name | Signature | UMKC-Id |
|---|---|---|---|
| 1: | BALAJI NATARAJAN | Natarajan | 16231992 |
| 2: | Aishwarya Iyer | N.Aishwarya | 16227781 |
| 3: | Gayathree Iyer | N.Gayathree | 16227784 |

## TABLE OF CONTENTS:

## Introduction:

Our main goal in this project is to design a variation of Dijkstra's Shortest Path algorithm to find the shortest path where shortest is now defined in a stochastic setting- Uncertain Shortest Path.

Dijkstra's algorithm is an algorithm for finding the shortest path between nodes in a graph which may represent, for example, road networks. This can be explained in terms of road network between cities. In such a network, nodes are the cities, or intersections, or server nodes, and the links connect them. If there are more than one cars on the road, it may cause interference and extra delay. The delay fluctuates according to a random variable for each edge which is uncertain. Such a path is an Uncertain Shortest Path.

We implemented Dijkstra's algorithm with MIN-HEAP in order to efficiently generate the uncertain paths needed for comparison. We generate the shortest path according to several criteria and then compare these paths.

## Experimental Design:

We have used the following criteria:

### 1. Mean Value

In this criterion, we pick that path whose total "expected value" is smallest from among all paths. The average weight per step of the path should be lower than in every other path.

We use this criterion because mean value of the given distribution represents the overall expected value of the delay index.

The cost of executing this criterion: The mean path length changes proportionally to log n where n is the number of nodes in the network.

### 2. Optimist

In this criterion, we pick a path such that its "expected value – standard deviation" is smallest from among all paths.

We use this criterion because the expected value represents the mean and standard deviation shows how the values are spread out. The resulting value shows how much the expected delay index differs from standard spread out of the distribution.

$$\bar{x} \pm 1SD$$

### 3. Pessimist

In this criterion, we pick a path such that its "expected value + standard deviation" is smallest from among all paths.

We use this criterion because the expected value of a distribution along how spread out the values are can show the truth resulting value of the distribution. What I mean is that, if the expected value is say 50 and S.D (Standard Deviation) is 0, expected value will be the actual value and the error in judgement will obviously be 0. Depending upon the S.D it will represent legitimate delay index.

$$\bar{x} \pm 1SD$$

### 4. Double Pessimist

In this criterion, we pick a path such that its "expected value + 2 standard deviation" is smallest from among all paths.

We use this criterion, because the mean value along with magnitude of deviation of values in the distribution magnified (* 2) will help us get a better result.

$$\bar{x} \pm 2SD$$

### 5. Stable

In this criterion, we pick a path such that its "squared coefficient of variation" is smallest from among all paths.

The squared coefficient of variation is defined as:

$$c^2(X) = \frac{V[X]}{E[X]^2}$$

We use this criterion, because: Squared coefficient of variation is a normalized variance which plays a crucial role in modelling delays in communication networks. If $c^2$ is less than 1, RV (density) is more concentrated around its mean. On the other hand, if $c^2$ is greater than 1, RV is less concentrated around its mean and the possibility of getting a delay index value will get better in the distribution. Also, the squared coefficient of variation becomes smaller as the path gets longer.

### 6. Dispersion Index

In this criterion, we pick a path such that the ratio of "variance" and "mean" is smallest from among all paths.

Dispersion Index is a normalized measure of the dispersion of probability distribution. It is the ratio of variance to mean.

$$D = \frac{\sigma^2}{\mu}.$$

We use this criterion, because dispersion index is a measure used to determine whether a set of observed occurrences are dispersed or clustered when compared to a standard statistical model. Squared mean value in denominator shouldn't make the delay index probable value accuracy any better. The spread is captured by the numerator i.e., variance and squaring or cubing it can potentially make the probability that the resulting value from the distribution is any of the values in that.

### Execution Cost:

EXTRACT_MIN Heap operation takes O(log V). There will be |V| such operations. The time to build the binary heap is O(V). Each DECREASE_KEY operation takes O(log V) time. There are at most |E| such operations. Therefore, the total running time is: O((V+E)log V) which is O(E* Log V).

## Experimental Implementation:

We used Python programming language to implement the algorithm. Python is the most compelling language for this algorithm because of the following reasons:

- Coding the algorithm becomes easier because of its indentation based syntax.
- Python provides the fundamental data structures that can be used directly by the algorithm.
- Complex data structures such as trees and graphs can be expressed in python in more concise, human-readable form without having to reinvent the data structures.

The data structures we used to support an efficient implementation of this algorithm are:

- Graph
- List
- Heap
- Hash Table
- Tuple

# Data Collection and Interpretation of Results:

## Pseudocode:

```
algorithm SimplifiedDijkstra( graph myGraph, node a, node b)
////Color all nodes to WHITE and set all Distances to ∞
MyMinheap ← minheap.create()
Distance(a) ← 0
Color(a) ← GRAY (i.e. "in-the-heap")
MyMinheap.insert(a)
while NOT MyMinheap.IsEmpty do
    do Next ← MyMinheap.delete() until Color(Next) == GRAY end-do      /* Smallest distance GRAY node
    Color(Next) ← BLACK
    if Next == b then RETURN end if
    for every NON-BLACK Node adjacent to Next do
        // Try to RELAX the Distance
        if Distance(Next) + Cost(Next, Node) < Distance(Node)
            then do
            Distance(Node) ← Distance(Next) + Cost(Next, Node)
            Color(node) ← GRAY (i.e. "in-the-heap")
            MyMinheap.insert(Node)
            end do; endif
    end-for-every
end while
end algorithm SimplifiedDijkstra
```

# Pseudo Code

Algorithm CalculateDelayIndex ($\alpha$, $\beta$, rule)

Switch rule :

case 1 // Deterministic

mean = $\alpha$ , variance = 0 , $c^2 = 0$ ;

case 2 // Uniform [$\alpha$, $\beta$]

mean = $(\alpha+\beta)/2$ , variance = $(\beta-\alpha)^2/12$ , $c^2 = \left(\frac{\beta-\alpha}{\alpha+\beta}\right)^2 \cdot \frac{1}{3}$ ;

case 3 // Exponential

mean = $1/\alpha$ , variance = $1/\alpha^2$ , $c^2 = 1$ ;

Case 4 // Shifted Exponential

mean = $\beta + 1/\alpha$ , variance = $1/\alpha^2$ , $c^2 = (1/1+\beta\alpha)^2$

case 5 // Normal

mean = $\alpha$ , variance = $\beta$ , $c^2 = \beta/\alpha^2$

case 6 // General

mean = $\alpha$ , variance = $\alpha^2\beta$ , $c^2 = \beta$

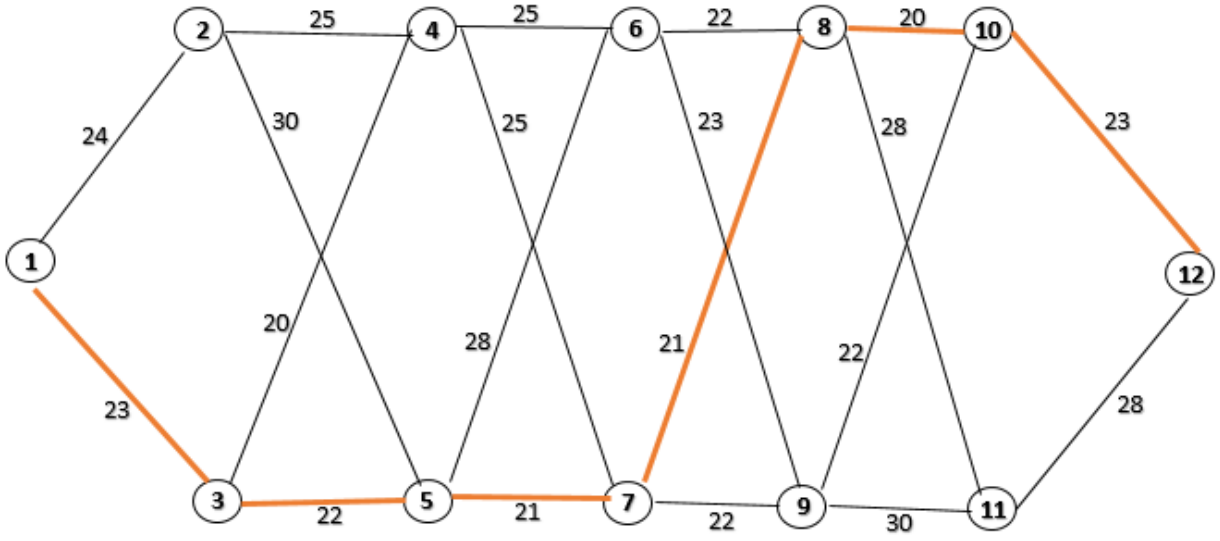
return list ( mean, variance , $c^2$ )

```
Algorithm UncertainPath
~~Create_graph_from_Input~~
For each Input
e = CalculateDelay_Index (input[α], input[β], input[rule])
create edge E in graph G
Add edge to graph
end loop
SimplifiedDijkstra_of(G, start_node, end_node)
```

**Input-1:**

```
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,
```

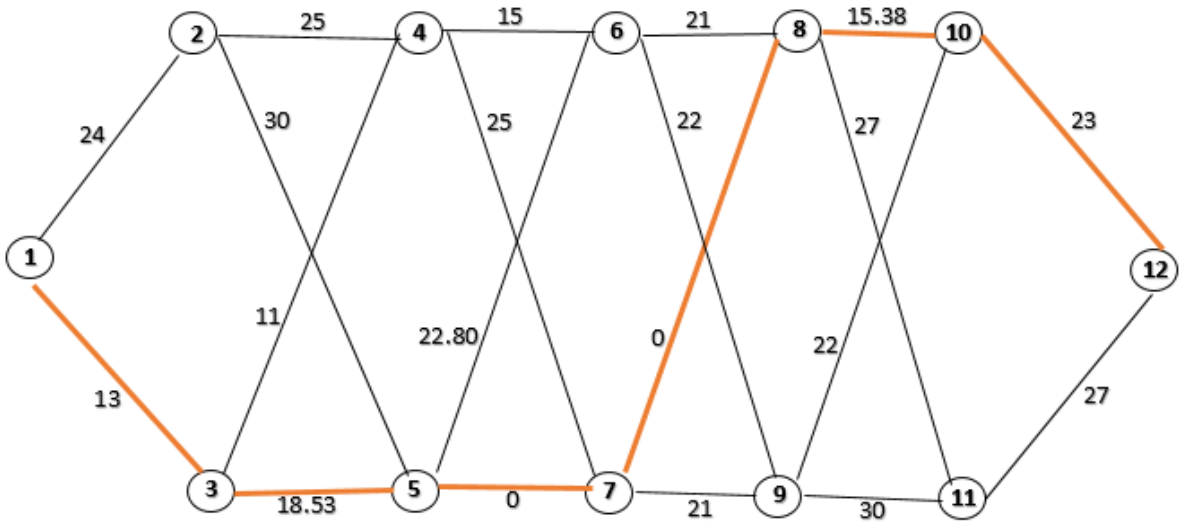
## 1) Mean




Shortest Path: **1-3-5-7-8-10-12**

Weight: 130

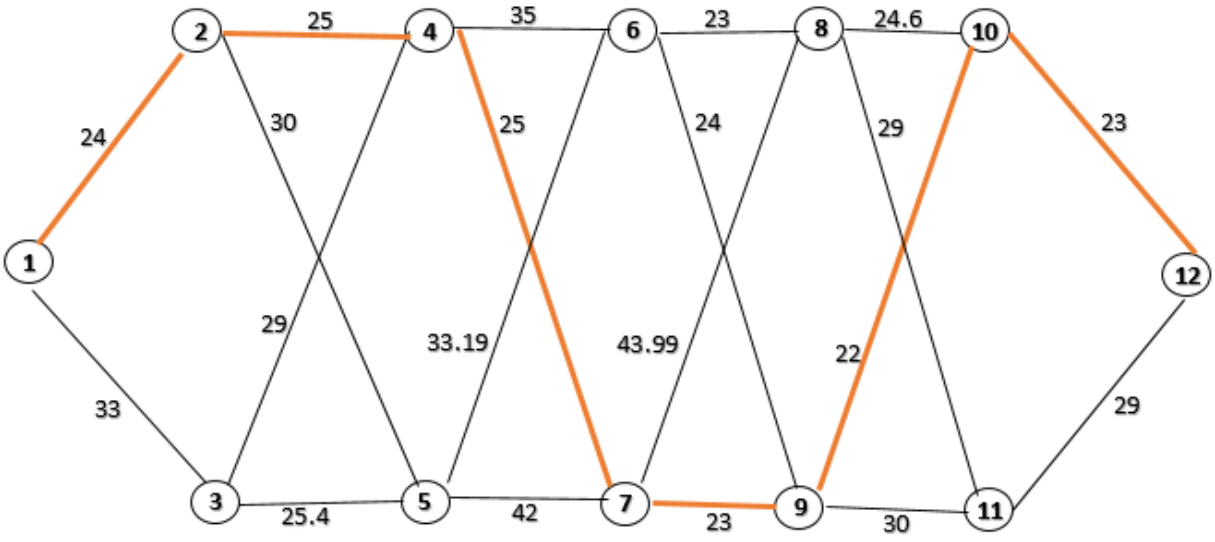
## 2) Optimist




Shortest Path: **1-3-5-7-8-10-12**

Weight: 69.91

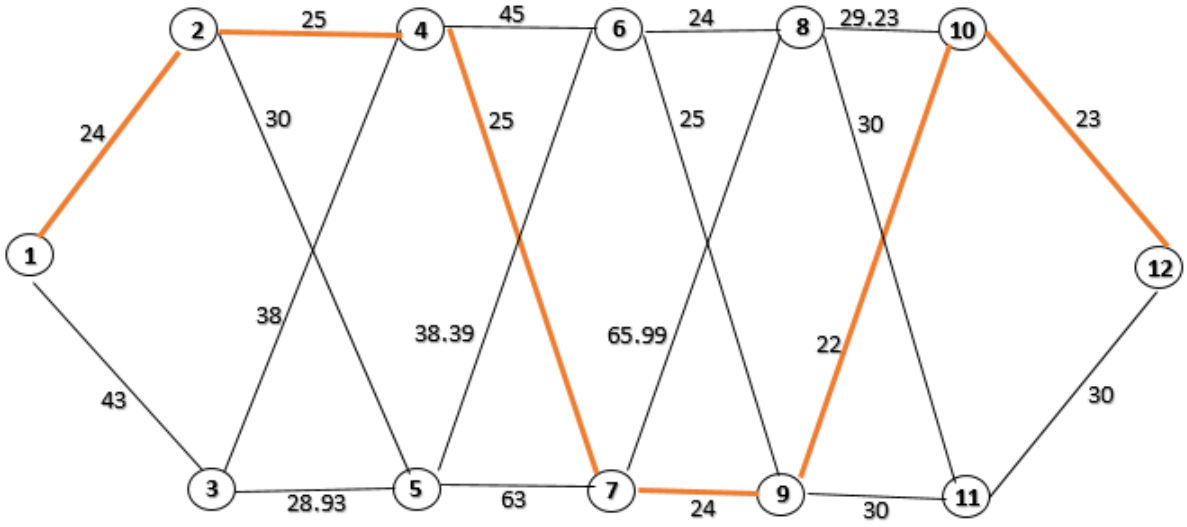
## 3) Pessimist




Shortest Path: **1-2-4-7-9-10-12**

Weight: 142

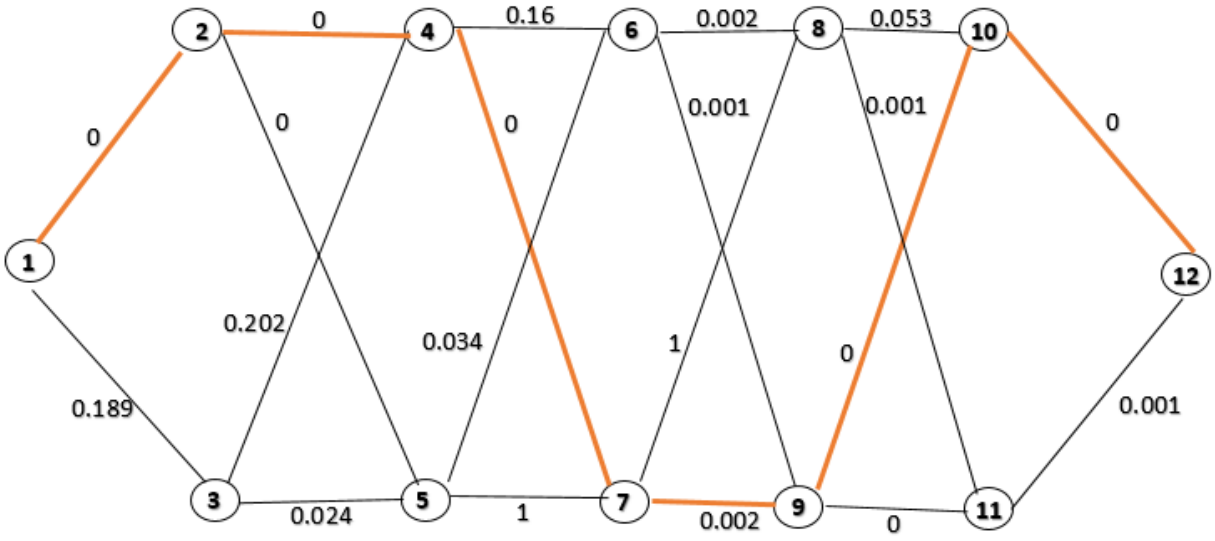
## 4) Double Pessimist




Shortest Path: **1-2-4-7-9-10-12**

Weight: 143

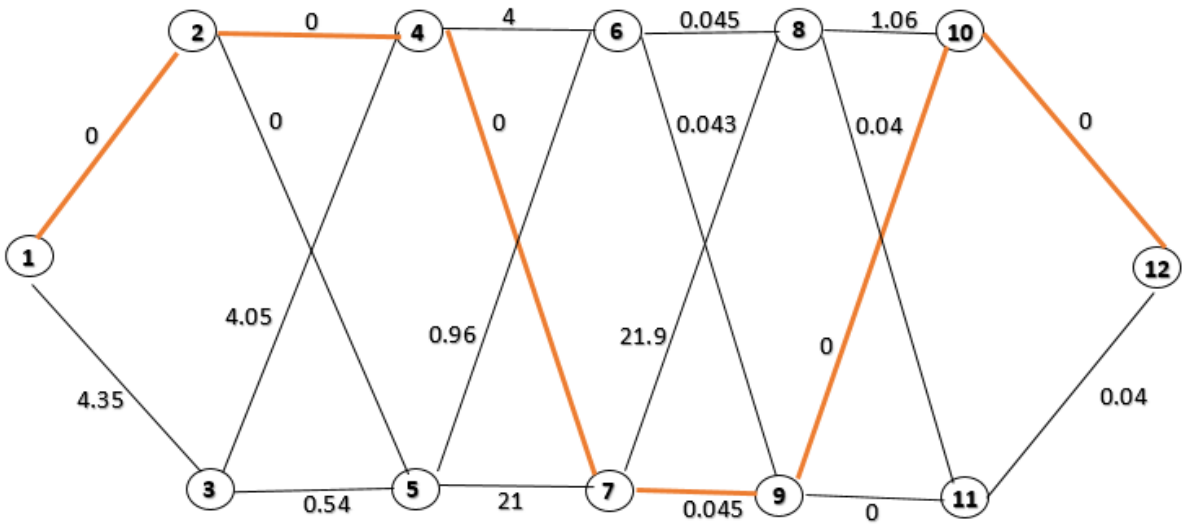
## 5) Stable




Shortest Path: **1-2-4-7-9-10-12**

Weight: 0.002

## 6) Dispersion Index




Shortest Path: **1-2-4-7-9-10-12**

Weight: 0.045

### Performance Measure:

| | μ - σ | μ | μ + σ | μ + 2σ | $C^2(Y)$ | V/μ | Hops |
|---|---|---|---|---|---|---|---|
| **Mean Value** | 69.917 | 130.99 | 192.08 | 253.16 | 2.267 | 48.96 | 6 |
| **Optimist** | 69.917 | 130.99 | 192.08 | 253.16 | 2.267 | 48.96 | 6 |
| **Pessimist** | 140 | 141 | 142 | 143 | 0.0020 | 0.045 | 6 |
| **Double Pessimist** | 140 | 141 | 142 | 143 | 0.0020 | 0.045 | 6 |
| **Stable** | 140 | 141 | 142 | 143 | 0.0020 | 0.045 | 6 |
| **Dispersion Index** | 140 | 141 | 142 | 143 | 0.0020 | 0.045 | 6 |

Here, the path-mean is indicated by μ, standard deviation is indicated by σ. The hop-length is indicated by Hops.

The shortest path for “Mean Value” and “Optimist” comes out to be same, which is, **1-3-5-7-8-10-12.**

The shortest path for “Pessimist”, “Double Pessimist”, “Stable”, “Dispersion Index” comes out to be same, which is, **1-2-4-7-9-10-12.**

Here, the edge (10,12) is common in all the criterion.

This table specifies whether a particular edge comes under each criterion.

| | Mean Value | Optimist | Pessimist | Double Pessimist | Stable | Dispersion Index |
|---|---|---|---|---|---|---|
| (1,2) | False | False | True | True | True | True |
| (1,3) | True | True | False | False | False | False |
| (2,1) | False | False | True | True | True | True |
| (2,4) | False | False | True | True | True | True |
| (2,5) | False | False | False | False | False | False |
| (3,1) | True | True | False | False | False | False |
| (3,4) | False | False | False | False | False | False |
| (3,5) | True | True | False | False | False | False |
| (4,2) | False | False | True | True | True | True |
| (4,3) | False | False | False | False | False | False |
| (4,6) | False | False | False | False | False | False |
| (4,7) | False | False | True | True | True | True |
| (5,2) | False | False | False | False | False | False |
| (5,3) | True | True | False | False | False | False |
| (5,6) | False | False | False | False | False | False |
| (5,7) | True | True | False | False | False | False |
| (6,4) | False | False | False | False | False | False |
| (6,5) | False | False | False | False | False | False |
| (6,8) | False | False | False | False | False | False |
| (6,9) | False | False | False | False | False | False |
| (7,4) | False | False | True | True | True | True |
| (7,5) | True | True | False | False | False | False |
| (7,8) | True | True | False | False | False | False |
| (7,9) | False | False | True | True | True | True |
| (8,6) | False | False | False | False | False | False |
| (8,7) | True | True | False | False | False | False |
| (8,10) | True | True | False | False | False | False |
| (8,11) | False | False | False | False | False | False |
| (9,6) | False | False | False | False | False | False |
| (9,7) | False | False | True | True | True | True |
| (9,10) | False | False | True | True | True | True |
| (9,11) | False | False | False | False | False | False |
| (10,8) | True | True | False | False | False | False |
| (10,9) | False | False | True | True | True | True |
| (10,12) | True | True | True | True | True | True |
| (11,8) | False | False | False | False | False | False |
| (11,9) | False | False | False | False | False | False |
| (11,12) | False | False | False | False | False | False |
| (12,10) | True | True | True | True | True | True |
| (12,11) | False | False | False | False | False | False |

The common edge that comes under all the criterion is edge (10,12).

## Edge Count:

```
Edges along with their counts in all the criteria

(('10', '12'), 6)
(('4', '7'), 4)
(('7', '9'), 4)
(('9', '10'), 4)
(('1', '2'), 4)
(('2', '4'), 4)
(('7', '8'), 2)
(('5', '7'), 2)
(('3', '5'), 2)
(('8', '10'), 2)
(('1', '3'), 2)
[cloudera@quickstart DAA]$
```

This output specifies the count of each edge in all the criteria. From this output, we can conclude that the edge (10,12) is present the most number of times. The edge (10,12) occurs 6 times. The edges (4,7), (7,9), (9,10), (1,2) and (2,4) have counts as 4. Other edges like (7,8), (5,7), (3,5), (8,10), (1,3) have count as 2.

## Conclusion:

The criteria mean and optimist result in the same path because optimist is mean - Standard Deviation (small quantity) ~ mean. Other criteria, pessimist (MEAN + S.D), and the double pessimist (MEAN + 2 * S.D) have the similar formula. The same reason can be applied to Stable and dispersion index. MEAN criterion is easy to understand because it involves only one statistical term existing in isolation. All other criteria, involves more than 1 statistical term and has to be interpreted carefully. All the criteria are easy to program and maintain as all follow the same algorithm.

## Epilogue:

The algorithm that we have used has proved to work the best through various test phases over different input sizes. However, the futuristic approaches may prove to work better. The programming language and the data structure that we have chosen has played a crucial role in improving the time complexities. However, big data platforms like Hadoop and Spark can be used to improve an important factor i.e. Scalability.

From this project we learnt that the uncertain shortest path can be calculated based on various criteria which can be used as a real world application like road networks.

The lasting lesson that we learnt from this project is that the algorithmic decisions play a vital role in improving the performance.

# Appendix:

## Output 1.1 :

```
[cloudera@quickstart DAA]$ python diji_mean.py
Graph data:
11 8 28.0
11 9 30.0
11 12 28.0
10 8 20.0
10 9 22.0
10 12 23.0
12 10 23.0
12 11 28.0
1 2 24.0
1 3 23.0
3 1 23.0
3 4 20.000009
3 5 22.0
2 1 24.0
2 4 25.0
2 5 30.0
5 2 30.0
5 3 22.0
5 6 28.0
5 7 21.000021
4 2 25.0
4 3 20.000009
4 6 25.0
4 7 25.0
7 4 25.0
7 5 21.000021
7 8 21.9997800022
7 9 22.0
```

When we run the program for the mean criterion, we get the graph data specifying all the start nodes, destination nodes and the mean for that corresponding edge.

## Output 1.2:

```
7 9 22.0
6 4 25.0
6 5 28.0
6 8 22.0
6 9 23.0
9 6 23.0
9 7 22.0
9 11 30.0
9 10 22.0
8 6 22.0
8 7 21.9997800022
8 11 28.0
8 10 20.0
Dijkstra's shortest path
updated : current = 1 next = 2 new_dist = 24.0
updated : current = 1 next = 3 new_dist = 23.0
updated : current = 3 next = 4 new_dist = 43.000009
updated : current = 3 next = 5 new_dist = 45.0
not updated : current = 2 next = 4 new_dist = 43.000009
not updated : current = 2 next = 5 new_dist = 45.0
updated : current = 4 next = 6 new_dist = 68.000009
updated : current = 4 next = 7 new_dist = 68.000009
not updated : current = 5 next = 6 new_dist = 68.000009
updated : current = 5 next = 7 new_dist = 66.000021
updated : current = 7 next = 8 new_dist = 87.9998010022
updated : current = 7 next = 9 new_dist = 88.000021
not updated : current = 6 next = 8 new_dist = 87.9998010022
not updated : current = 6 next = 9 new_dist = 88.000021
updated : current = 8 next = 11 new_dist = 115.999801002
```

Also, the output specifies what nodes are traversed to obtain the shortest path.

## Output 1.3 :

```
not updated : current = 9 next = 11 new_dist = 115.999801002
not updated : current = 9 next = 10 new_dist = 107.999801002
updated : current = 10 next = 12 new_dist = 130.999801002
not updated : current = 11 next = 12 new_dist = 130.999801002
The shortest path : ['1', '3', '5', '7', '8', '10', '12']
*****************************************
Mean: 130.999801002
Mean - SD: 69.9170962313
Mean + SD: 192.082505773
Mean + 2 * SD: 253.165210544
C Squared: 2.26716263859
Dispersion Index: 48.9597483013
Hop count: 6
Final Table
('12', '10')    True
('6', '9')      False
('8', '10')     True
('3', '4')      False
('9', '11')     False
('1', '2')      False
('7', '9')      False
('7', '5')      True
('10', '8')     True
('2', '5')      False
('11', '9')     False
('12', '11')    False
('8', '11')     False
('6', '4')      False
('9', '10')     False
('5', '3')      True
('7', '4')      False
```

This output specifies the shortest path obtained for the Mean criterion.

The shortest Path is [‘1’, ‘3’, ‘5’, ‘7’, ‘8’, ‘10’, ‘12’].

Also, we have calculated the mean, mean-SD, mean+SD, mean+2SD, C squared, Dispersion Index and Hop Count for the shortest path obtained in this criterion.

## Output 1.4 :

```
('4', '2')      False
('10', '9')     False
('10', '12')    True
('5', '7')      True
('11', '8')     False
('4', '6')      False
('7', '8')      True
('6', '5')      False
('5', '2')      False
('9', '7')      False
('11', '12')    False
('4', '3')      False
('6', '8')      False
('8', '6')      False
('5', '6')      False
('2', '1')      False
('4', '7')      False
('1', '3')      True
('3', '1')      True
('9', '6')      False
('2', '4')      False
('8', '7')      True
('3', '5')      True
[cloudera@quickstart DAA]$
```

This output represents whether a particular edge occurs in the shortest path for this criterion.

Similarly, we have executed for each and every criterion.

# References

1. **Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein.** Introduction to Algorithms.

2. ***BogoToBogo.*** **[Online] 04 17, 2017.**
**http://www.bogotobogo.com/python/python_Dijkstras_Shortest_Path_Algorithm.php.**