**PROJECT REPORT**

**DESIGN AND ANALYSIS OF ALGORITHM**

**SUBMITED TO:**

SIR WAHEED AHMED

**SUBMITED BY:**

FIZZA ZAKIR 19K-1106

AISHA ANWAR 19K-1086

**ABSTRACT:**

**“Implementing six algorithms to get the shortest path from starting node to final node on 10 sets of input files.”**

**ALGORITHMS:**

1. Prims
2. Kruskals
3. Dijkstra
4. Bellman ford
5. Floyd Warshall algorithm
6. Clustering coefficient in graph theory
7. Borůvka's algorithm

**INTRODUCTION**:

1. **PRIMS ALGORITHM:**

* Prim’s Algorithm is a famous greedy algorithm.
* It is used for finding the Minimum Spanning Tree (MST) of a given graph.
* To apply Prim’s algorithm, the given graph must be weighted, connected and undirected.

1. **KRUSKALS ALGORITHM**:

* Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach.
* This algorithm treats the graph as a forest and every node it has as an individual tree
* A tree connects to another only and only if, it has the least cost among all available options and does not violate MST properties.

1. **DIJKSTRA ALGORITHM:**

* Dijkstra Algorithm is a very famous greedy algorithm.
* It is used for solving the single source shortest path problem.
* It computes the shortest path from one particular source node to all other remaining nodes of the graph.

1. **BELLMAN FORD ALGORITHM:**

* Bellman-Ford Algorithm terminates upon finding a negative cycle and hence, can be used to detect it.
* It is used in cycle-cancelling techniques in network flow analysis.
* Bellman-Ford Algorithm works without the full network view/ knowledge and thus, can be used in distance vector algorithms.

1. **FLOYD WARSHALL:**

* Floyd-Warshall Algorithm is an algorithm for finding the shortest path between all the pairs of vertices in a weighted graph.
* This algorithm works for both the directed and undirected weighted graphs.
* It does not work for the graphs with negative cycles (where the sum of the edges in a cycle is negative)

**PROPOSED SYSTEM:**

we ask the user input, based on the input and choice of the user, the respective benchmark file is opened after the conditional check of switch case which is applied for 10 different input options. The respective Algorithm reads the file and starts executing to find the shortest path among different nodes and calculate the total cost from one node to another.

GUI representation shows the outcome of the algorithm to the user

**EXPERIMENTAL SETUP:**

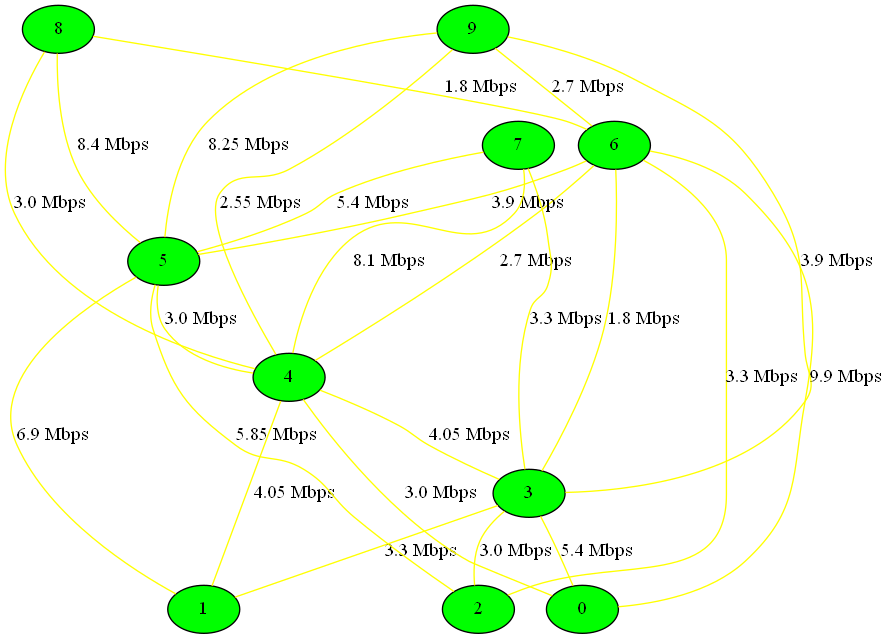
* **TIME COMPLEXITIES:**

1. PRIMS ALGORITHM-  **O ((V+E) logV)**
2. KRUSKAL ALGORITHM- **O(ElogV)**
3. DIJKSTRA ALGORITHM- **O (V2)**
4. BELLMON FORD- **O(V-E)**
5. FLOYD WARSHALL- **O(n3)**
6. CLUSTERING COEFFICIENT- **O (kn2)**

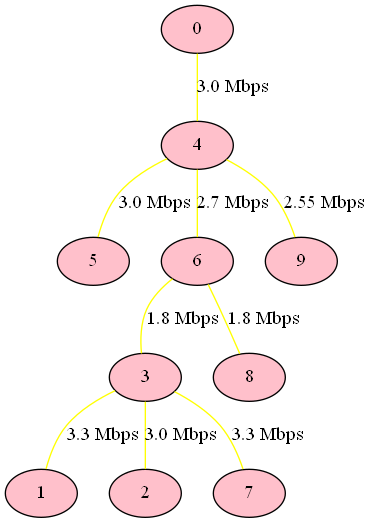
**GRAPHICAL REPRESENTATION:**

These graph will show the shortest path from above mentioned algorithms on 10 input nodes.

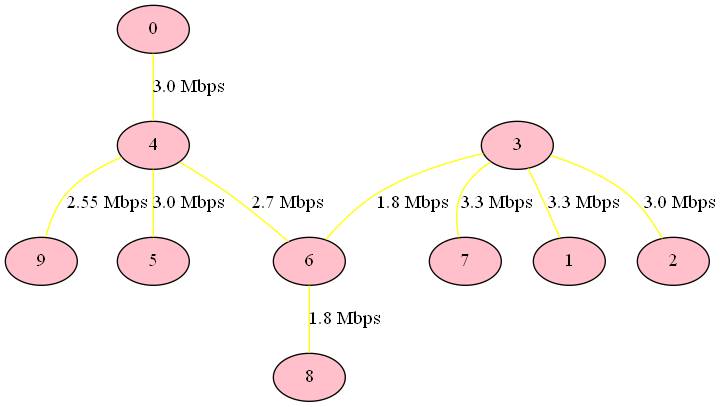
* **Sample Input**

****

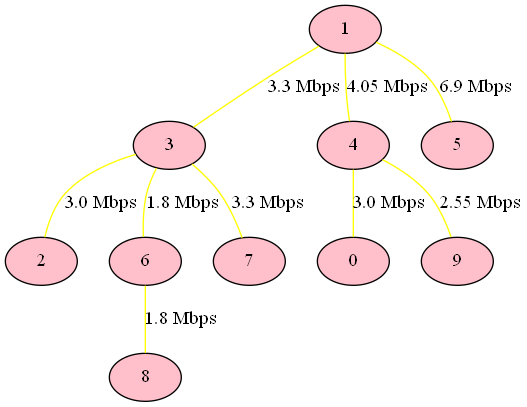
* **PRIMS ALGORITHM:**



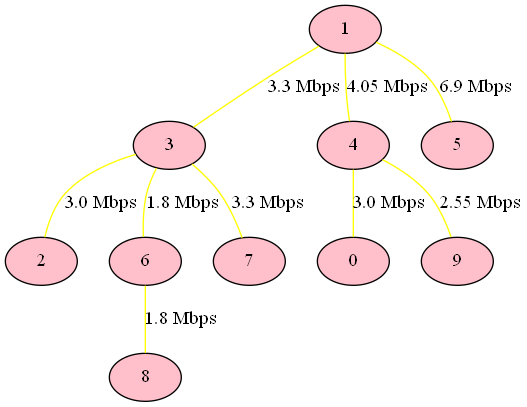
* **KRUSKAL ALGORITHM:**



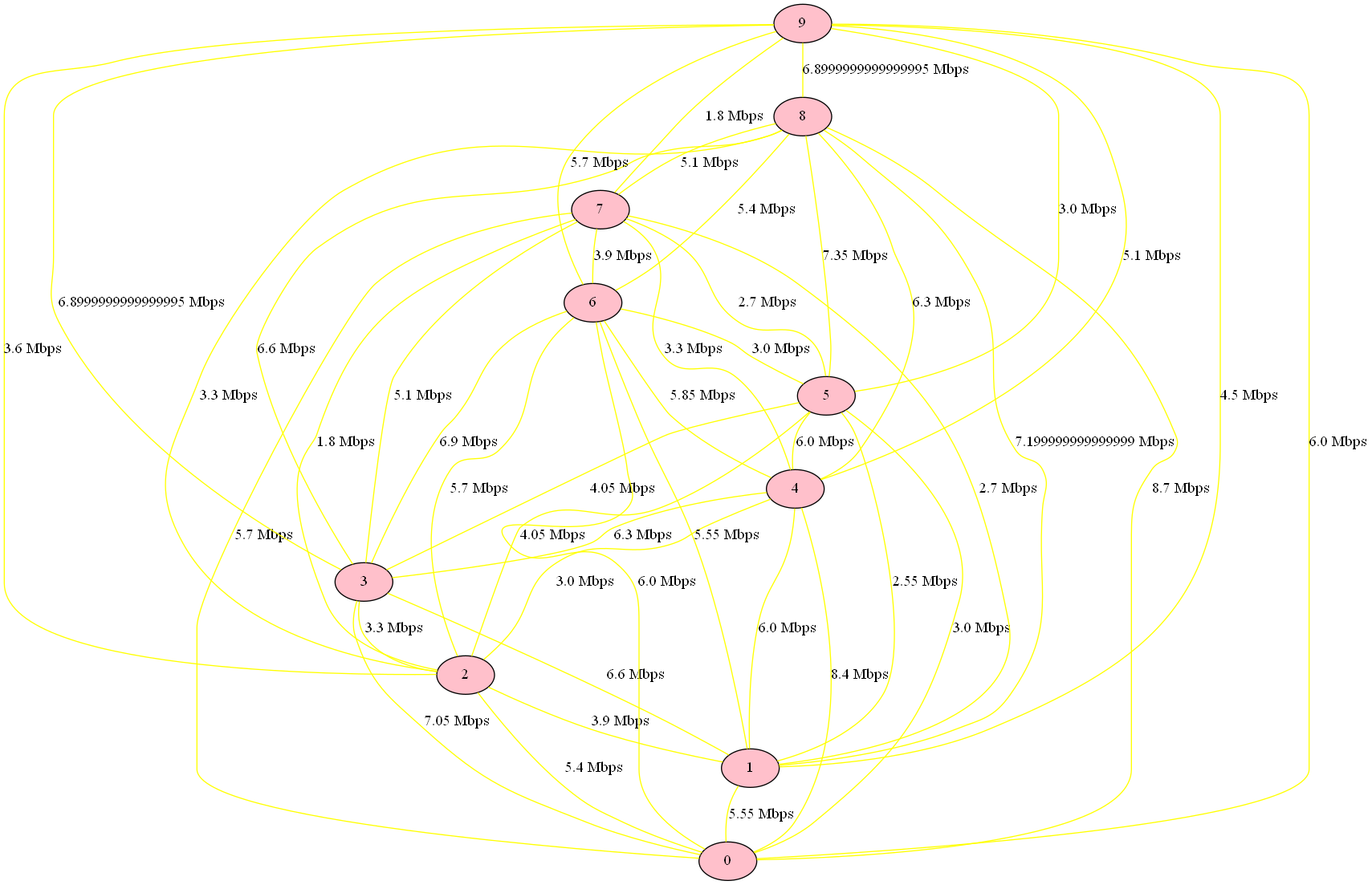
* **DIJKSTRA ALGORITHM:**

****

* **BELLMAN FORD ALGORITHM:**

****

* **FLOYD WARSHALL ALGORITHM:**



**RESULT:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Benchmark** | **Prims Total cost ing Mbps** | **Kruskal** | **Dijisktra (Choose any input node e.g. 5)** | **Bellman-For d (Choose any input node e.g. 5)** | **Floyd**  **Warshall Algorith m** | **Clusterin g**  **Coefficient (Local Clusterin g)** |
| **Input 10** | 24.45 | 24.45 | 52.8 | 52.8 | 453.6 | 0.6583 |
| **Input 20** | 51.45 | 51.45 | 122.25 | 122.25 | 2274.9 | 0.48284 |
| **Input 30** | 87.6 | 87.6 | 162.9 | 162.9 | 6271.79 | 0.69809 |
| **Input 40** | 137.1 | 137.1 | 344.55 | 344.55 | 13504.2 | 0.77097 |
| **Input 50** | 133.05 | 133.05 | 260.4 | 260.4 | 15542.09 | 0.6114 |
| **Input 60** | 201.15 | 201.15 | 557.249 | 557.249 | 28674.6 | 0.7103 |
| **Input 70** | 195.749 | 195.749 | 475.35 | 475.35 | 36364.5 | 0.68545 |
| **Input 80** | 249.3 | 249.3 | 427.35 | 427.35 | 50689.5 | 0.70086 |
| **Input 90** | 287.1 | 287.1 | 765.3 | 765.3 | 65643.0 | 0.78583 |
| **Input 100** | 304.5 | 304.5 | 693.149 | 693.149 | 80038.79 | 0.69942 |