Assignment 9

Sayak Ghorai || BT21GCS004 || B2 || Design and Analysis of Algorithm

Q: Write a code to demonstrate Bellman-Ford Algorithm to find shortest path from source node to any other node.

Answer:

Approach: first make a graph with certain Edges and Vertecies and then add edges between Vertecies. That creates a Graph Structure. Now we can easily apply Bellman Ford's Algo in this structure.

CreateGraph.java | ihb.java

```
class CreateGraph {
    3 usages
    class CreateEdge {
        9 usages
        int source, destination, weight;

        1 usage
        CreateEdge() {
            source = destination = weight = 0;
        }
    }
    2 usages
    int Vertex, Edge;
    26 usages
    CreateEdge[] edge;
    1 usage
    CreateGraph(int vertex, int e) {
        Vertex = vertex;
        Edge = e;
        edge = new CreateEdge[e];
        for (int i = 0; i < e; ++i)
            edge[i] = new CreateEdge();
    }
```

```java
void BellmanFord(CreateGraph graph, int s) {
    int V = graph.Vertex, E = graph.Edge;
    int[] dist = new int[V];
    for (int i = 0; i < V; ++i)
        dist[i] = Integer.MAX_VALUE;
    dist[s] = 0;
    for (int i = 1; i < V; ++i) {
        for (int j = 0; j < E; ++j) {
            int u = graph.edge[j].source;
            int v = graph.edge[j].destination;
            int w = graph.edge[j].weight;
            if (dist[u] != Integer.MAX_VALUE && dist[u] + w < dist[v])
                dist[v] = dist[u] + w;
        }
    }
    for (int j = 0; j < E; ++j) {
        int u = graph.edge[j].source;
        int v = graph.edge[j].destination;
        int w = graph.edge[j].weight;
        if (dist[u] != Integer.MAX_VALUE && dist[u] + w < dist[v]) {
            System.out.println("CreateGraph contains negative w cycle");
            return;
        }
    }
    printSolution(dist, V,s);
}

1 usage
void printSolution(int[] dist, int V,int s) {
    System.out.println("Source\tVertex\tDistance");
    for (int i = 0; i < V; ++i)
        System.out.println("  "+s+" -----> "+i + "       " + dist[i]);
}
```

```java
    public static void main(String[] args) {
        int V = 5;
        int E = 8;
        CreateGraph graph = new CreateGraph(V, E);
        // edge 0 --> 1
        graph.edge[0].source = 0;
        graph.edge[0].destination = 1;
        graph.edge[0].weight = 4;
        // edge 0 --> 2
        graph.edge[1].source = 0;
        graph.edge[1].destination = 2;
        graph.edge[1].weight = 3;

        // edge 1 --> 3
        graph.edge[2].source = 1;
        graph.edge[2].destination = 3;
        graph.edge[2].weight = 6;

        // edge 2 --> 1
        graph.edge[3].source = 2;
        graph.edge[3].destination = 1;
        graph.edge[3].weight = 5;

        // edge 3 --> 2
        graph.edge[4].source = 3;
        graph.edge[4].destination = 2;
        graph.edge[4].weight = 2;

        graph.edge[5].source = 3;
        graph.edge[5].destination = 4;
        graph.edge[5].weight = 2;
        graph.BellmanFord(graph,  s: 0); // 0 is the source vertex
    }
}
```

Output:

```
 /Users/sayakghorai/Desktop/DAA_Assignments/Assignment9/out/production/Assignment9 CreateGraph
Source  Vertex  Distance
  0 -----> 0       0
  0 -----> 1       4
  0 -----> 2       3
  0 -----> 3       10
  0 -----> 4       12

Process finished with exit code 0
```