

shippingRoutes_SSSP_BFS.java

```
1
2
3 /* BFS SSSP Single Source Shortest Path on an un-weighted
   graph
4
5
6
7 import java.util.*;
8
9 class shippingRoutes_SSSP_BFS{
10     static Node[] G;
11     static int E;
12     static int N;
13     static int P;
14
15     static class Node {
16         List<Edge> adj;
17         String name;
18         int n;
19         public boolean visited;
20         int layer;
21         public Node(int N) {
22             adj = new ArrayList<Edge>();
23             n=N;
24             layer = -1;
25             visited = false;
26             name = null;
27         }
28     }
29     static class Edge{
30         int to, weight;
31         public Edge(int t, int w) {
32             to=t;
33             weight = w;
34         }
35     }
36     public static void makeGraph(int n) {
37         G = new Node[n];
38         for(int i =0; i<n; i++){
39             G[i]=new Node(i);
40         }
41     }
42     public static void addEdge(int u,int v, int w) {
```

shippingRoutes_SSSP_BFS.java

```
43         G[u].adj.add(new Edge(v,w));
44         G[v].adj.add(new Edge(u,w));
45     }
46     public static int charN(char c){
47         return c;
48     }
49     public static void init_bfs(){
50         for(int i = 0; i<G.length; i++){
51             G[i].layer = -1;
52             G[i].visited = false;
53         }
54     }
55     public static int bfs(int s, int t){
56         G[s].layer = 0;
57         Queue<Integer> q = new LinkedList<Integer>();
58         q.add(s);
59         int cur = -1;
60         while (!q.isEmpty()){
61             cur = q.peek();
62             G[cur].visited = true;
63             for (Edge e : G[cur].adj){
64                 if(!(G[e.to].visited)){
65                     q.add(G[e.to].n);
66                     G[e.to].visited = true;
67                     G[e.to].layer = G[cur].layer + 1;
68                 }
69             }
70             q.poll();
71         }
72         // for(Node node : G)
73         // System.out.print(node.layer+" ");
74         return G[t].layer;
75     }
76
77
78     public static void main(String[] args){
79         Scanner scan = new Scanner(System.in);
80         System.out.println("SHIPPING ROUTES OUTPUT");
81         System.out.println();
82         int K = scan.nextInt();
```

shippingRoutes_SSSP_BFS.java

```
83     String temp = "String";
84     String a = "String";
85     String b = "String";
86     int size = 0;
87     int length = 0;
88     for(int k =0; k<K; k++){
89         N = scan.nextInt();
90         E = scan.nextInt();
91         P = scan.nextInt();
92         makeGraph(N);
93         System.out.println("DATA SET  " + (k+1));
94         System.out.println();
95         for(int i = 0; i<N; i++){
96             temp = scan.next();
97             G[i].name = temp;
98         }
99         for(int i = 0; i<E; i++){
100             a = scan.next();
101             b = scan.next();
102             for(int x = 0; x<N; x++){
103                 for(int y = 0; y<N; y++){
104                     if(G[x].name.equalsIgnoreCase(a) &&
G[y].name.equalsIgnoreCase(b))
105                         {
106                             addEdge(x,y,1);
107                         }
108                 }
109             }
110         }
111         for(int i = 0; i<P; i++){
112             size = scan.nextInt();
113             a = scan.next();
114             b = scan.next();
115             for(int x = 0; x<N; x++){
116                 for(int y = 0; y<N; y++){
117                     if(G[x].name.equalsIgnoreCase(a) &&
G[y].name.equalsIgnoreCase(b))
118                         {
119                             init_bfs();
120                             length = bfs(x,y);
```

shippingRoutes_SSSP_BFS.java

```
121         }
122     }
123 }
124     if(length>0)
125         System.out.println("$" + length*size*100);
126     else
127         System.out.println("NO SHIPMENT
POSSIBLE");
128     }
129     System.out.println();
130
131 }
132 System.out.println("END OF OUTPUT");
133 scan.close();
134 }
135 }
136
```