CSN-261: Data Structures Laboratory

Lab Assignment 12 (L12)

---

1. Write a Java program to implement Dijkstra's algorithm to find all shortest paths between all pair of vertices in a weighted graph. Modify this algorithm to find all shortest paths between two nodes, if more than one occurs. Following this, compute the betweenness centrality measure of each node.

   Reference: 'Betweenness Centrality' of a node/vertex, w is given as, $BC(w) = \sum_{u,v \in V} \frac{\sigma_{uv}(w)}{\sigma uv}$ where σ_{uv} is the number of all shortest paths between u and v through w.

   Input:

   The Shortest path can be computed based on the cost (sum of weights along the path). Low cost indicates the shortest path and more than one shortest paths are possible. Betweenness Centrality can be calculated based on the total number of shortest paths from the source node to target node, passing through particular vertex/node $\sigma_{uv}(w)$ and the total number of shortest paths from the source node to target node (σ_{uv}).

   First line contains two integers: number of Nodes n and number of edges e.
   Followed by e lines each containing three integers: vertex u, vertex v and edge weight w between u and v

   Output:

   Betweenness Centrality of each node, separated by space.

   Sample Input

   - 10 20
   - 0 1 4
   - 0 4 8
   - 0 6 7

- 0 7 5
- 0 8 1
- 0 9 9
- 1 3 8
- 1 5 6
- 1 6 8
- 1 7 2
- 1 8 4
- 2 3 5
- 2 8 3
- 3 9 5
- 4 5 8
- 4 9 8
- 5 7 5
- 5 8 5
- 7 9 3
- 5 9 5

Sample Output

- 7.8333 1.3333 6.7500 2.6667 0.0000 9.0000 0.0000 1.7500 7.0000 6.2500

2. For a given weighted graph G, its vertices show the routers, and edge weights show the bandwidth between the routers. Write a program in Java to find the bandwidth between two designated routers, i.e., vertices. This can be achieved by maximizing the weight of the minimum-weight edge in the path between the given routers.

   Input

   - Adjacency matrix of the weighted graph G.
   - Vertex 1, v_1.
   - Vertex 2, v_2.

Output

Value of the bandwidth between v_1 and v_2.

Sample Input

$$\begin{bmatrix} 0 & 30 & 11 & 2 & (\infty) \\ 30 & 0 & 8 & \infty & \infty \\ 11 & 8 & 0 & 3 & 14 \\ 2 & \infty & 3 & 0 & 43 \\ \infty & \infty & 14 & 43 & 0 \end{bmatrix}$$

A
D

Sample Output

11

3. Mr. Alex is planning for a tour with his family during winter vacation. He lives in a country with N states (1 to N) with R bidirectional interstate roads connecting them. We know the amount of fuel required by Mr. Alex to traverse each road, moving within a state consumes no fuel. The cost of petrol is fixed at F_i units of money per liter for each state i. Mr. Alex is a very clever person, so he carries big barrels in his car to store extra petrol and avoids purchasing petrol in costly states. Therefore, he can carry any amount of petrol he needs during the journey, and he may visit the same state more than once if he saves his money. Mr. Alex starts the trip without any fuel.

   How could you help Mr. Alex find the minimum possible amount of money he should spend on petrol to travel from state P to Q?

   Input

   - The first row of the input has a single integer T indicates the number of test cases. The details of T test cases are as follows:
   - The first row of each test case has two space-separated integers N and R.
   - The next R rows have three space-separated integers u, v, and w. It indicates that a road links the state u and state v, and it requires w liter of petrol to traverse that road.

- The next row has N space-separated integers F_1, F_2, F_3.....F_N.
- The next row has two space-separated integers P and Q.

Output

Print the minimum amount of money spent by Mr. Alex, in a single line, having one integer value, for each test case.

Constraints

- $1 \leq T \leq 5$
- $1 \leq N \leq 300$
- $1 \leq R \leq N(N-1)/2$
- $1 \leq u, v, P, Q \leq N$
- Each road connects two different states.
- No two roads connect the same pair of states.
- $1 \leq F_i \leq 10^9$ for each valid i

Sample Input

1
4 3
1 2 3
2 3 1
2 4 7
3 6 2 2
1 4

Sample Output

28

Explanation

Mr. Alex can buy 10 liters of petrol in state 1, travel to state 2 and then state 4. This way, he spends 30 units of money. However, the optimal solution is to buy 4 liters of petrol in state 1, travel to state 2, then to state 3, buy 8 liters of petrol in state 3 and then travel to state 4, since this costs only 28 units of money.

4. Assume there is an infection spread in the country. There are many cities in which few are connected to other cities by some connections or roads. You are supposed to stop this spread by destroying some cities such that any destruction will remove all its connections with other cities. Your aim is to isolate all the cities (i.e., no more connection among them). Assume that each destruction of a city requires 1 unit of explosive. You are following the simple approach of destroying the city with the most number of connections in it at that moment. Note that there is no multi-connection among the cities (i.e. connected by only a single path). Calculate the total units of explosives needed. Implement the code in Java.
Note: In the case of multiple cities with the highest connections, destroy the city with the lowest index.

Input

- The first line will contain 2 space-separated integers N and M, where N is the number of cities in town, and M is the number of connections that the cities have.
- Next M lines follow each line consists of two integers A and B, specifying that city A has a connection with city B and B has a connection with city A.

Output

- For each test case, output the minimum explosive units required to achieve the goal.

Constraints

- $1 \leq N \leq$100000
- $1 \leq M \leq$300000
- $1 \leq A, B \leq$N

Sample Input
4 2
1 2

3 4

Sample Output
2