spoj.com Ada and trip

contributors: Andriy Korzhuk, David Zashkolnyi, Danylo Kovalenko

Problem statement - bla, bla, bla

- Ada the Ladybug loves trips. She travels around world taking photos and souvenirs. This week she went to Buganda. Common Tourist would surely travel around main city and some conurbations, but Ada has different politics. She wants to go as far as possible (because photos from outlying places are much more valuable).
- Problem is, that Buganda is very large so she can barely guess such places. Luckily, you are around so she asked you for help. Can you tell her, how far and how many cities are most distant (if the shortest path is used)?

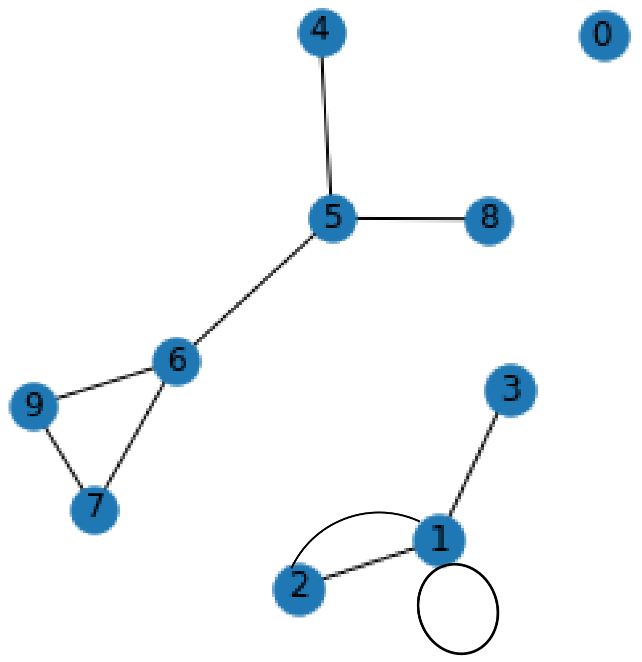
Problem statement - input

- The first line will contain three integers **$1 \leq N \leq 5*10^5$, $0 \leq M \leq 10^6$, Q**, the number of cities in Buganda, the number of roads and number of queries (possible arrival cities).
- Then **M** lines follow, with three integers **$0 \leq A, B < N$, $0 \leq L \leq 10$**, **A, B** are cities, which the (**<u>bidirectional</u>**) road connects and **L** is length of the road.
- Afterward, **Q** lines follow, each with number **$0 \leq q_i < N$**, meaning the city of arrival.
- You are assured that **max(N,M)*Q** will be always lesser/equal than **10^7**
- **<u>Gentle warning</u>**: Since we are in real world and not in some "graph theory", multiedges and self-edges are completely valid!

Problem statement - output

- For each query print two numbers: The distance of most distant place(s) and number of such places.

Example



```
# Input
// 10 cities, 10 edges and 10 queries
10 10 10

// next 10 lines – edges        // Queries,
// from, to, weight             // start point
1 1 1                           0
1 2 1                           1
1 2 3                           2
3 1 1                           3
5 4 10                          4
8 5 10                          5
5 6 5                           6
6 7 3                           7
6 9 3                           8
9 7 4                           9
```

```
# Output
// L, N
// L  – how far are the most
//      distant places from the start
// N – how many such places

0 1    // vertex 0 is alone
1 2    // 2, 3 connected with 1
2 1    // but no inbetween
2 1    // self-loop in 1 isn't matter
20 1
10 2
15 2
18 2
20 1
18 2
```

Problem statement – mat model

- According to the aforementioned task description we have an undirected graph with non-negative integer weights.
- The graph can contain cycles, self-loops and multiple edges.
- For each query we have to find vertices with the max shortest-path from the given start point.

Solution

- Let's build the bidirected orientation for the given graph.
- Since all the weights are non-negative integers the result digraph has no negative cycles
- We have a digraph with no negative cycles and we are required for shortest path from the source to the each vertex.
- Therefore, we can use Dijkstra algorithm.

Algorithm

- Build bidirected digraph as a list of adjacent vertices from the input (just add each edge twice)
- For each query run Dijkstra algorithm from the given start point
- Find value of the max shortest path.
- Find all the vertices with such shortest path and return.

Main code

- Input
- Dijkstra algo
- Find maximum

Full C implementation

```c
int main() {
    int N, M, Q;
    scanf("%d", &N);
    scanf("%d", &M);
    scanf("%d", &Q);

    // create Digraph with N vertices
    Digraph* digraph = createDigraph(N);
    for (int i = 0; i < M; i++){
        int tmp_from, tmp_to, tmp_len;
        scanf("%d", &tmp_from);
        scanf("%d", &tmp_to);
        scanf("%d", &tmp_len);

        // add each edge twice with opposite directions
        addEdge(digraph, tmp_from, tmp_to, tmp_len);
        addEdge(digraph, tmp_to, tmp_from, tmp_len);
    }

    for (int i = 0; i < Q; i++){
        int source;
        scanf("%d", &source);

        // run Dijkstra algo. Then find the biggest value in
        // the result distance table and how many times it appears
        DijkstraRet *res = DijkstraAlgo(digraph, source);
        maxDistance(digraph, res);
        printf("%d %d\n", res->maxDistance, mostDistantAmount(digraph, res));
    }
}
```

Complexity

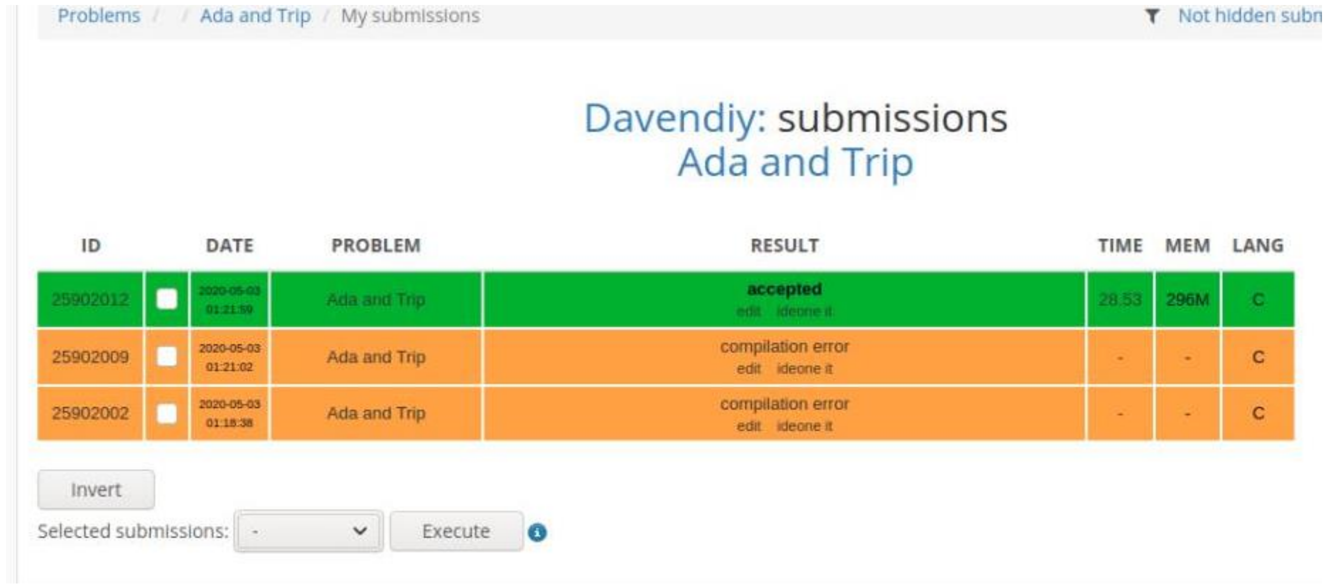
E – edges, V – vertices

- Preprocessing (read input + create digraph) - Θ(E) time and Θ(E+V) space
- Dijkstra with binary heap - Θ(E logV) time and Θ(V) extra space (not including edge-weighted digraph)
- Finding max - Θ(V) time and no extra space (just traverse all the vertices)

Results

- We have an algorithm that takes Θ(E logV + V) time and Θ(E + V) space in the worst case

- The real time and memory consumption could be decreased using less flexible and good-style coding



Problems / / Ada and Trip / My submissions

Not hidden subn

Davendiy: submissions
Ada and Trip

ID		DATE	PROBLEM	RESULT	TIME	MEM	LANG
25902012	☐	2020-05-03 01:21:59	Ada and Trip	**accepted** edit ideone it	28.53	296M	C
25902009	☐	2020-05-03 01:21:02	Ada and Trip	compilation error edit ideone it	-	-	C
25902002	☐	2020-05-03 01:18:38	Ada and Trip	compilation error edit ideone it	-	-	C

Invert

Selected submissions: - Execute

Thank you for attension.