Out For a Walk, v2015 (Subtask A)

Released: Saturday, 03 October 2015, 08.00am

Due: Saturday, 17 October 2015, 07.59am

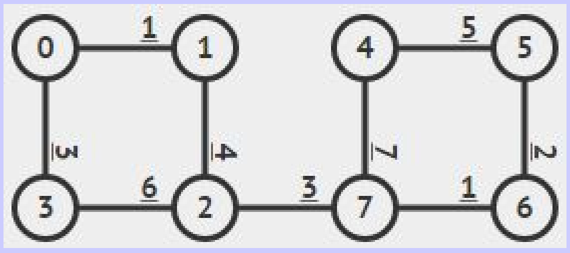
The Actual Problem Description

Given a layout of a building (as a connected undirected weighted graph of course, stored in an Adjacency List data structure), Grace's effort rating to traverse the corridors of that building (as integer weights between [0..1000] of the corresponding edges: lower weight means easier corridor for Grace, higher weight means harder corridor for Grace), Grace's source vertex, Grace's destination vertex, determine the maximum effort that Grace has to endure in order for her to go from the source vertex to the destination vertex (the edge with maximum weight along Grace's easiest path).

Grace is not in rush. She can take a longer path (detour, etc) as long as her maximum effort that she has to endure along that path is minimized.

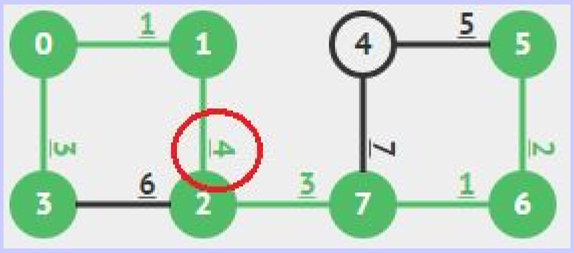
There will be **Q** queries with varying source and destination vertices. For this PS4, we restrict that the source vertices in the query can only range from [0..9] while the destination vertices in the query can range from [0..**V**-1].

For example, suppose that the building is a connected undirected weighted graph as shown below:



A Sample Building; For your convenience, view this connected undirected weighted graph in VisuAlgo

If Grace wants to go from point 3 to point 5, she will choose this path: 3 → 0 → 1 → 2 → 7 → 6 → 5. This is *not* the shortest path, but it is the easiest path for her as she only needs to endure maximum effort rating of 4 when she goes through corridor 1-2. The other corridors along this easiest path have effort ratings ≤ 4.



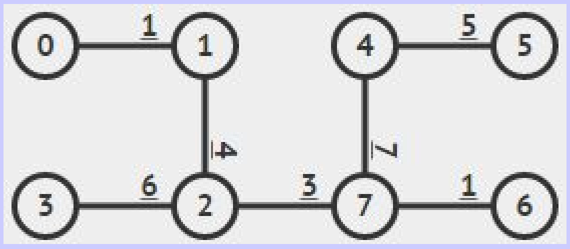
If Grace choose the shortest path (in terms of number of edges traversed): 3 → 2 → 7 → 6 → 5, she has to endure a tougher corridor 3-2 (with an effort rating of 6) compared to her easiest path above.

Your task is to implement one, two, or more method(s)/function(s):

- void PreProcess()
- This is an optional method that you may choose to use to speed up your queries.
- You can leave this method blank if you do not need it.
- int Query(int source, int destination)
- Query your chosen data structure and return the weight of a corridor (an edge) which has the highest effort rating along Grace's easiest path from source vertex to destination vertex.
- We guarantee that source is different than destination.

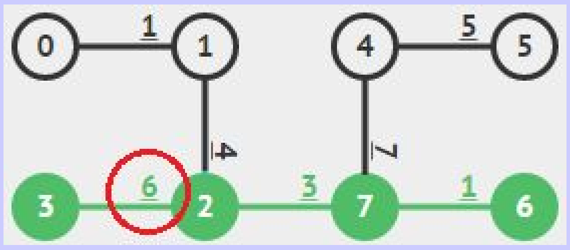
Subtask A Constraints

Time Limit: 1s.
The building is a small weighted tree (2 ≤ **V** ≤ 10, 1 ≤ Q ≤ 5).
See Figure below for an example (source vertex: 3, destination vertex: 6).



Another Sample Building (Tree), first test case in Sample Input; For your convenience, view this connected undirected weighted graph in VisuAlgo

In this building (weighted tree), Grace's easiest path is 3 → 2 → 7 → 6. The hardest corridor (edge) for Grace is 3-2 with weight 6. Therefore, the answer is 6.



The output for first test case, first query in Sample Input

Sample Input

```
1

8
1   1 1
2   0 1   2 4
3   1 4   3 6   7 3
1   2 6
2   5 5   7 7
1   4 5
1   7 1
3   2 3   4 7   6 1
3
3 6
3 5
0 3
```

Sample Output

```
6
7
6
```

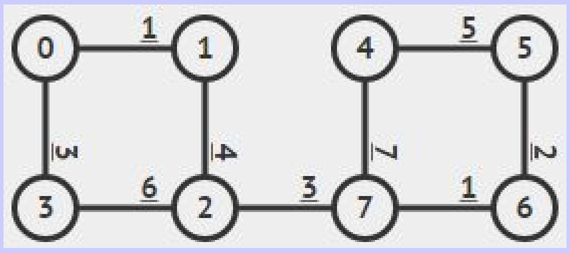
Out For a Walk, v2015 (Subtask B)

The Actual Problem Description

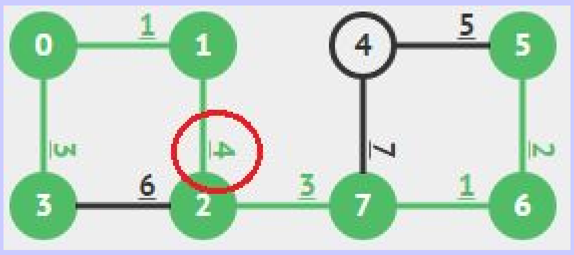
Please refer to Subtask A for the full problem description.

Subtask B Constraints

Time Limit: 1s.
The building is a medium weighted graph (2 ≤ V ≤ 400, 0 ≤ E ≤ 100 000, 1 ≤ Q ≤ 5).



A Sample Building, first test case in Sample Input; For your convenience, view this connected undirected weighted graph in VisuAlgo



The output for first test case, first and second queries in Sample Input

Sample Input

```
1

8
2   1 1   3 3
2   0 1   2 4
3   1 4   3 6   7 3
2   0 3   2 6
2   5 5   7 7
2   4 5   6 2
2   5 2   7 1
3   2 3   4 7   6 1
3
```

```
3 6
3 5
0 3
```

Sample Output

```
4
4
3
```