NUS WebMail IVLE LIBRARY MAPS

Search search for... in NUS Websites ▼ GO

CodeCrunch

Home | My Courses | Browse Tutorials | Browse Tasks | Search | My Submissions | Logout | Logged in as: **e0009011**

CS2010 2016/2017 Sem2: PS4 A

Tags & Categories

Tags:

Categories:

Related Tutorials

Task Content

Getting from here to there, v2017 (Subtask A)

Released: Friday, 3rd of March 2017, 12:00 noon
Due: Saturday, 17th of March 2017, 11:59 pm

You are encouraged to work with other students or teaching staffs (inside or outside this module) on solving this problem set. However, you **must** write Java code **by yourself**. In addition, when you write your Java code, you **must** list the names of every collaborator, that is, every other person that you talked to about the problem (even if you only discussed it briefly). This list may include certain posts in CS2010 Facebook group. If you have access to CS2010 files from senior batch (that is, CS2010 problem sets from year 2011-2015), please refrain from looking at their code verbatim or worse... submit your senior's code. Automatic special checks are done especially on the last/hardest subtask to compare older code with this year's version. Any deviation from this policy will be considered as cheating. If the offender is caught beyond reasonable doubt, he/she will be punished severely, including referral to the NUS Board of Discipline.

Story

As you know, pregnant women, parents carrying small babies and handicapped individuals are given priority seats in MRT (the corner seats at each car), buses (usually the front seats), and virtually at every other public places. We feel grateful every time a person who occupied one of those priority seats gave his/her seat for us during those times (although we are also often irritated by young people who 'ignored' the presence of a pregnant woman/parent carrying a small baby/handicapped -- usually either 'sleeping' or 'playing with their smart phone' -- and do not give up their seats[1].

Not just about MRTs and buses, the handicapped especially those who are wheelchair bound also need to take safer and easier paths when getting from one place to another. Climbing a staircase requires a huge effort and might even be impossible for those in a wheelchair unless they have external help, so if there is a lift, an elevator, or a gradually increasing slope somewhere in that building, the wheelchair bound person will prefer to take the *easier path*, even if it means a longer path.

Even though this is the case. There is no reason why the wheelchair bound cannot go out for walks or get from one place to another, especially since computer science can be used to help make it easier for them by computing the easiest path for them to take. Since you have just been taught 'something' in CS2010 that can be used to solve this problem, so please solve this problem.

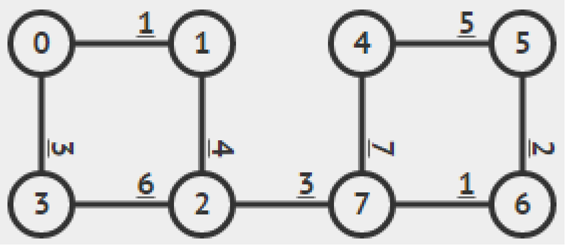
The Actual Problem Description

Given a layout of a building (as a connected undirected weighted graph of course, stored in an Adjacency List data structure), the wheelchair bound person'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, higher weight means harder corridor), the person's source vertex and destination vertex, determine the maximum effort that the wheelchair bound person has to endure in order for him/her to go from the source vertex to the destination vertex (the edge with maximum weight along the person's easiest path).

The wheelchair bound person can take a longer path (detour, etc) as long as his/her maximum effort that has to be endured along that path is minimized.

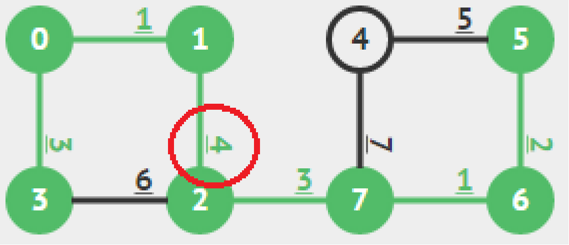
There will be **Q** queries with varying source and destination vertices. For this PS4, we restrict that the source vertices in the query can only range from `[0..min(9,V-1)]` while the destination vertices in the query can range from `[0..`**`V`**`-1]`.

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 the person wants to go from point 3 to point 5, he/she will choose this path: `3 → 0 → 1 → 2 → 7 → 6 → 5`. This is *not* the shortest path (that we will explore later in PS5), but it is the easiest path for him/her as he/she only needs to endure maximum effort rating of 4 when he/she goes through corridor `1-2`. The other corridors along this easiest path have effort ratings ≤ 4.



The path that the wheelchair bound takes

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

The skeleton program GettingFromHereToThere.java (click to view) that can handle all input/output details is already written for you.

Your task is to implement one, two, or more^2 method(s)/function(s):

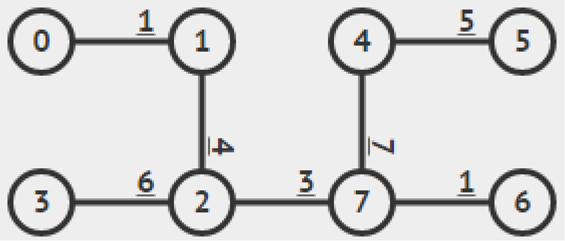
- `void PreProcess()`
  This is an optional method that you may choose to use to speed up your queries.
  You can leave this method blank if you do not need it.
- `int Query(int source, int destination)`
  Query your chosen data structure and return the weight of a corridor (an edge) which has the highest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

Subtask A Constraints (40 points)

Time Limit: 1s.

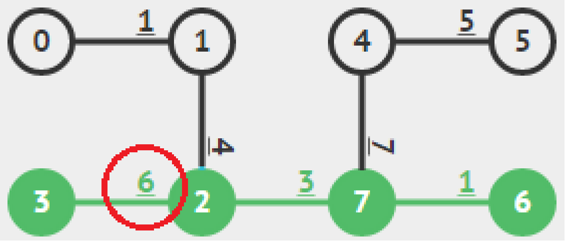
The building is a small weighted tree (2 ≤ **V** ≤ 10, 1 ≤ Q ≤ 5).

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), the wheelchair bound person's easiest path is `3 → 2 → 7 → 6`. The hardest corridor (edge) 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
```

Generating Test Data

The given sample input/output are for illustration purpose and are not enough to verify the correctness of your solution.

You are encouraged to generate and post additional test data in CS2010 Facebook group.

Please use GettingFromHereToThereVerifier.java (click to view) to verify whether your custom-made test data conform with the required specifications.

Problem Author

Dr Steven Halim/Dr Chong Ket Fah
For CS2010.

Footnotes

[1] CS2010 students and teaching staffs!!, please give up your seat in public places to those who need it more!!

[2] If needed, you can write additional helper methods/functions to further simplify your code.

CS2010 2016/2017 Sem2: PS4 B

Tags & Categories

Tags:

Categories:

Related Tutorials

Task Content

Getting from here to there, v2017 (Subtask B)

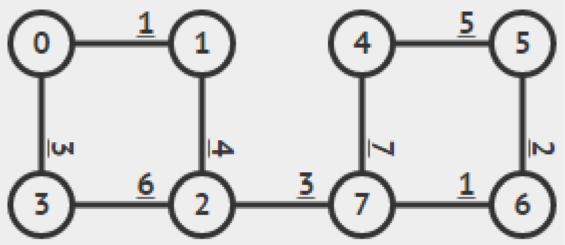
The Actual Problem Description

Please refer to Subtask A for the full problem description.

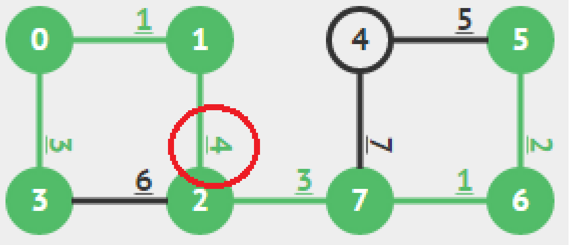
Subtask B Constraints (additional 10 points)

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
```

Problem Author

Dr Steven Halim/Dr Chong Ket Fah
For CS2010.

Terms of Use | Privacy | Non-discrimination

MySoC | Computing Facilities | Search | Campus Map
School of Computing, National University of Singapore

CS2010 2016/2017 Sem2: PS4 C

Tags & Categories

Tags:

Categories:

Related Tutorials

Task Content

Getting from here to there, v2017 (Subtask C)

The Actual Problem Description

Please refer to Subtask A for the full problem description.

Subtask C Constraints (additional 40 points)

Time Limit: 1s.

The building is a large weighted graph (2 ≤ **V** ≤ 2 000, 0 ≤ **E** ≤ 100 000, 1 ≤ **Q** ≤ 5).

Problem Author

Dr Steven Halim/Dr Chong Ket Fah
For CS2010.

CS2010 2016/2017 Sem2: PS4 D

Tags & Categories

Tags:

Categories:

Related Tutorials

Task Content

Getting from here to there, v2017 (Subtask D)

The Actual Problem Description

Please refer to Subtask A for the full problem description.

Subtask D Constraints (additional 10 points)

Time Limit: 1s.

The building is a large weighted graph with lots of queries ($2 \leq \mathbf{V} \leq 2\ 000$, $0 \leq \mathbf{E} \leq 100\ 000$, $1 \leq \mathbf{Q} \leq 100\ 000$).

Hint: Re-read the full problem description that is written in Subtask A again **very carefully**.

Problem Author

Dr Steven Halim/Dr Chong Ket Fah
For CS2010.