

CS4225/CS5425 Big Data Systems for Data Science

Assignment 2: Spark

Outline

- Description of Assignment 2
- Submission requirements

Task Overview

○ Motivation

- OpenStreetMap(OSM) files are big data.
- OSM files processing is useful for road navigation.
- Get familiar with Spark.

Problem Description

- Given an OSM file, you need to extract the nodes (i.e., positions) and edges(roads) information to construct a road graph.
- Write an algorithm that can find the (shortest) path between any two nodes in the road graph.

Problem Description

○ Input

- An OSM file that contains NUS road information, named as 'NUS.osm'
- A file named 'input.txt' that contains multiple node pairs. For each nodes pair, you need to find a path from the source node to the destination node.
- For each nodes pair in input file, there is at least one path between them.

Input file1:

```
1345143432 1854366392
1345143432 1832488252
```

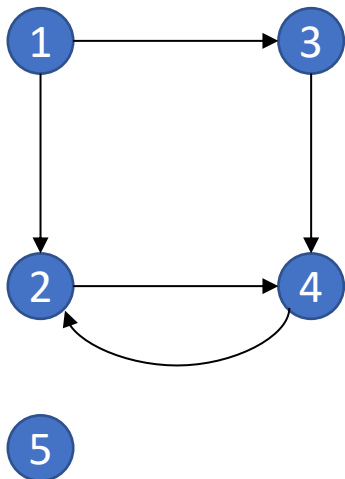
Source node

Destination node

Problem Description

Output 1

- After constructing your road graph, you need to output the graph in adjacent list format.
- In each line, you should output the id for a node, and the id for its **outgoing** neighbors. You should sort the outgoing neighbors in ascending order.
- You should also sort all the lines in ascending order of the id.



If you construct a road graph as the left side, your output of the road graph should be:

```
1 2 3
2 4
3 4
4 2
5
```

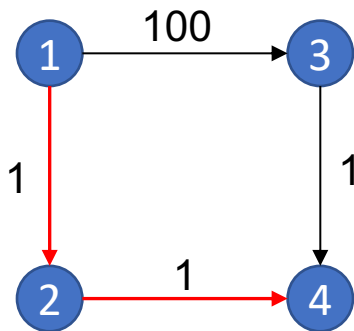
The id for outgoing neighbors are in ascending order

The id for the first node in each line are in ascending order

Problem Description

○ Output 2

- For each node pair in the input file, you need to output a directed path **in one line**.
- The path should contain all the nodes in order included the source and destination nodes. The nodes should be connected by ' -> ' symbol.



If you construct a road graph as the left side, and the input nodes pair is '1 4', your output can be: '1 -> 2 -> 4' or '1 -> 3 -> 4' .

Problem Description

○ Output 2

- For each node pair, you need to output a directed path **in one line**.
- The path should contain all the nodes in order included the source and destination nodes. The nodes should be connected by ' -> ' symbol.

Example output of path for file1:

```
(base) ➔ build.mac git:(master) X cat result.txt
1345143432 -> 5131711927 -> 496801425 -> 1832517056 -> 1832517057 -> 496801423 -> 1832517073 -> 496801421 -> 1832517061 -> 1345143390 -> 1832517063 ->
1832517069 -> 7927504651 -> 7927504650 -> 7927504649 -> 7927504648 -> 7927504647 -> 7927504646 -> 7927169989 -> 496801413 -> 1832513974 -> 1345143448
-> 1832513976 -> 239713813 -> 1832513978 -> 1345143424 -> 1832513980 -> 239713812 -> 1832513985 -> 1345143372 -> 1832513986 -> 7927159376 -> 18325139
89 -> 1345143343 -> 1832513991 -> 239713810 -> 1555047419 -> 1345143437 -> 1601408309 -> 1832513993 -> 1832513995 -> 1601405578 -> 7919717647 -> 79197
17645 -> 1534827400 -> 239712264 -> 7919717642 -> 7919717646 -> 239712263 -> 1534825218 -> 1534825214 -> 4724763212 -> 1556605806 -> 239712262 -> 4724
763211 -> 5243866724 -> 6912796541 -> 1736138338 -> 1345143345 -> 239712261 -> 1534825221 -> 1345143342 -> 4724810956 -> 4724810957 -> 416342635 -> 21
70644532 -> 239712259 -> 610183475 -> 610183473 -> 1345143429 -> 239712258 -> 6114545165 -> 5282980941 -> 687603784 -> 5137605421 -> 239624382 -> 1358
272528 -> 4724827744 -> 1358272552 -> 5137604020 -> 1358272569 -> 2489562875 -> 239712257 -> 1358272605 -> 687603782 -> 1358272624 -> 1358272634 -> 15
53228162 -> 1553269896 -> 1553228152 -> 1553269893 -> 1553228128 -> 1553228134 -> 1553228113 -> 1553228156 -> 1553228163 -> 1553228159 -> 1553228143 ->
7930174104 -> 1553228149 -> 4362557691 -> 1854366366 -> 1358272689 -> 5136423038 -> 6957615527 -> 1854366370 -> 1854366372 -> 1854366374 -> 18543663
76 -> 6957615526 -> 1854366378 -> 1854366385 -> 1854366386 -> 1854366387 -> 1854366388 -> 1854366389 -> 5286734106 -> 1854366390 -> 5286734104 -> 1854
366392
1345143432 -> 5131711927 -> 496801425 -> 1832517056 -> 1832517057 -> 496801423 -> 1832517073 -> 496801421 -> 1832517061 -> 1345143390 -> 1832517063 ->
1832517069 -> 7927504651 -> 7927504650 -> 7927504649 -> 7927504648 -> 7927504647 -> 7927504646 -> 7927169988 -> 7927169987 -> 7927169986 -> 792716998
5 -> 7927159384 -> 7927159383 -> 7927159382 -> 7927159381 -> 7927159380 -> 7927159379 -> 7927159378 -> 7927159377 -> 1748955727 -> 1748955734 -> 17489
55737 -> 1748955732 -> 1748955725 -> 1748955714 -> 1748955712 -> 1748955706 -> 1748955703 -> 977377433 -> 7936899673 -> 977370345 -> 6275692195 -> 513
6287745 -> 1867527121 -> 1345143326 -> 8011567625 -> 7936899663 -> 8011567624 -> 1867527123 -> 1867527125 -> 977383402 -> 1345143441 -> 361030526 -> 1
345143402 -> 976478207 -> 1832488294 -> 496810111 -> 1345143428 -> 496810112 -> 1832488295 -> 687603003 -> 496810113 -> 1832488293 -> 1832488292 -> 18
32488291 -> 1832488290 -> 1832488289 -> 1832488288 -> 1832488287 -> 1832488286 -> 1832488285 -> 7957586760 -> 1832488284 -> 1832488283 -> 3568629948 ->
1838839670 -> 3568629945 -> 1832488282 -> 1832488281 -> 1832488280 -> 1832488279 -> 1832488278 -> 1832488277 -> 496810131 -> 1832488276 -> 183248827
4 -> 496810132 -> 4487888850 -> 3079562407 -> 496810134 -> 1832488261 -> 1832488259 -> 1832488257 -> 7958078027 -> 1832488252
```


Problem Description

○ OSM file format

- For this assignment, you only need to focus on two types of data, i.e., node data and road data.
- A node data represents a location. It will start with '<node' and end with '/>' or '</node>', the id for each node is **unique**. You need to extract the id, latitude and longitude information for each node.

```
<node id='148146073' timestamp='2019-11-13T14:12:46Z' uid='8911947' user='aj_34' visible='true' version='6' changeset='77017228' lat='1.3031694' lon='103.7735031' />
<node id='148146084' timestamp='2007-12-02T14:20:46Z' uid='20032' user='Stefan8' visible='true' version='1' changeset='41870' lat='1.2958415' lon='103.7847878'>
  <tag k='created_by' v='JOSM' />
</node>
<node id='208526076' timestamp='2011-12-16T06:48:12Z' uid='564272' user='huaiwei' visible='true' version='7' changeset='10129053' lat='1.288834' lon='103.8192765' />
<node id='208526086' timestamp='2011-12-16T06:48:12Z' uid='564272' user='huaiwei' visible='true' version='6' changeset='10129053' lat='1.2888672' lon='103.8190241' />
<node id='208526111' timestamp='2011-12-16T06:48:12Z' uid='564272' user='huaiwei' visible='true' version='6' changeset='10129053' lat='1.2899916' lon='103.8162031' />
```

Node id

latitude

longitude

Problem Description

○ OSM file format

- For this assignment, you only need to focus on two types of data, i.e., node data and road data.
- A node data represents a location. It will start with '<node' and end with '>' or '</node>', the id for each node is **unique**. You need to extract the id, latitude and longitude information for each node.

```
<node id='148146084' timestamp='2007-12-02T14:20:46Z' uid='20032' user='Stefan8' visible='true' version='1' changeset='41870' lat='1.2958415' lon='103.7847878'>
  <tag k='created_by' v='JOSM' />
</node>
<node id='208523086' timestamp='2011-12-16T06:48:12Z' uid='564272' user='huaiwei' visible='true' version='6' changeset='10129053' lat='1.2888672' lon='103.8190241' />
<node id='208523097' timestamp='2011-12-16T06:48:12Z' uid='564272' user='huaiwei' visible='true' version='6' changeset='10129053' lat='1.2891463' lon='103.8180233' />
<node id='208523111' timestamp='2011-12-16T06:48:12Z' uid='564272' user='huaiwei' visible='true' version='6' changeset='10129053' lat='1.2899916' lon='103.8162031' />
```

Node id

latitude


longitude

Problem Description

○ OSM file format

- For this assignment, you only need to focus on two types of data, i.e., node data and road data.
- A road data represents a way that connect multiple nodes. It will start with '<way' and end with '</way>', the id for each way is **not unique**.
- For this assignment, we only consider the road data with a highway tag.

```
<way id='223674506' timestamp='2015-07-30T22:07:41Z' uid='1502014' user='saywhaaa' visible='true' version='2' changeset='32995337'>
  <nd ref='1833982457' />
  <nd ref='2325438476' />
  <nd ref='3674112907' />
  <nd ref='1555154266' />
  <tag k='highway' v='footway' />
</way>
```



Road data with
highway tag

Problem Description

○ OSM file format

- For this assignment, you only need to focus on two types of data, i.e., node data and road data.
- A road data represents a way that connect multiple nodes. It will start with '<way' and end with '</way>', the id for each way is **not unique**.
- For this assignment, we only consider the road data with a highway tag.

```
<way id='230638379' timestamp='2021-05-27T05:39:49Z' uid='10774744' user='sriwijoyosastro' visible='true' version='2' changeset='105394485'>  
  <nd ref='2389088994' />  
  <nd ref='2391321865' />  
  <nd ref='8771066619' />  
  <nd ref='2391321869' />  
  <nd ref='1855321078' />  
</way>
```

Road data without highway tag,
you can ignore this road

Problem Description

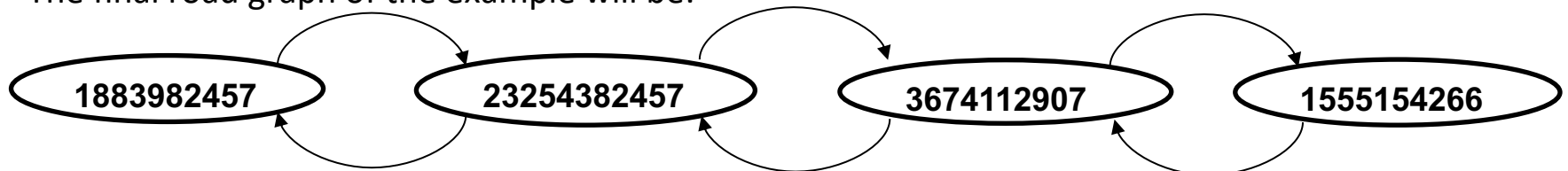
○ OSM file format

- For each road data, you need to extract all the nodes inside it. For each node, it will be reflected as `<nd ref='unique_id' />`.
- Only the adjacent nodes in the road data are directly connected to each other.

```
<way id='223674506' timestamp='2015-07-30T22:07:41Z' uid='1502014' user='saywhaaa' visible='true' version='2' changeset='32995337'>  
  <nd ref='1833982457' />  
  <nd ref='2325438476' />  
  <nd ref='3674112907' />  
  <nd ref='1555154266' />  
  <tag k='highway' v='Footway' />  
</way>
```

A node with id '1833982457'

The final road graph of the example will be:



Problem Description

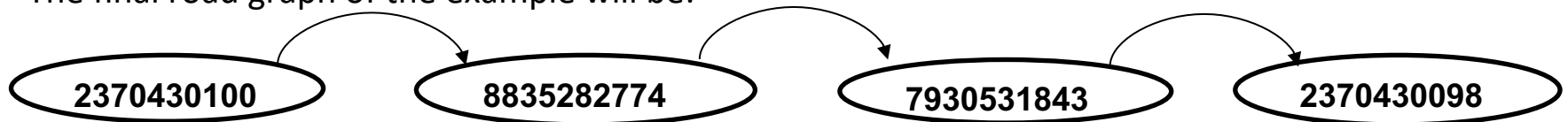
○ OSM file format

- A road is one-way if it has a oneway tag, i.e., the road data contain a line: `<tag k='oneway' v='yes' />`.
- The node in the road data is directly connected to node in the next line in one direction.

```
<way id='228362409' timestamp='2021-06-15T09:52:39Z' uid='10503923' user='ch_ram' visible='true' version='6' changeset='106390255'>
  <nd ref='2370430100' />
  <nd ref='8835282774' />
  <nd ref='7930531843' />
  <nd ref='2370430098' />
  <tag k='access' v='private' />
  <tag k='highway' v='service' />
  <tag k='lanes' v='1' />
  <tag k='oneway' v='yes' />
  <tag k='service' v='driveway' />
  <tag k='turn:lanes' v='through' />
</way>
```

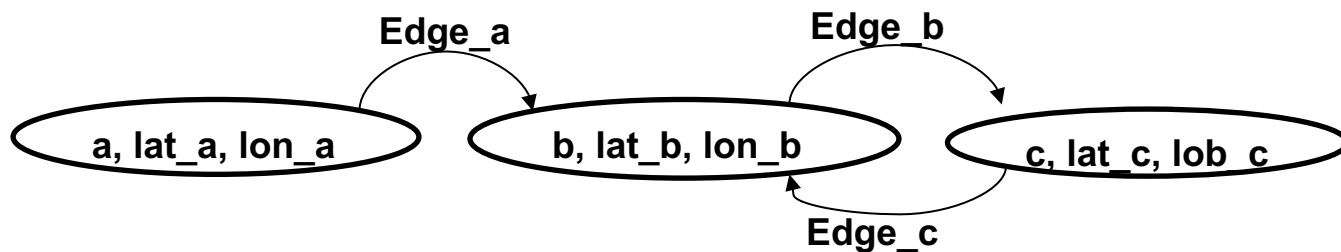
Oneway tag for the road

The final road graph of the example will be:



Problem Description

- After constructing your road graph, you need to use the latitude and longitude information to calculate the length of every edge. You can use our supported **distance** function.



Length of edge_a = distance(lat_a, lat_b, lon_a, lon_b)

Length of edge_b = Length of edge_c = distance(lat_b, lat_c, lon_b, lon_c)

Problem Description

- You need to use spark to design an algorithm that can find a path between any two nodes in the road graph.
- Your algorithm should be able to find the result **within 20 minutes** using the soc cluster. We will **stop your program** if your codes run more than 20 minutes.
- You should flush your result to the disk once you find the path for a test case, so that your buffered output won't lost if we stop your program.

Problem Description

- **Hint:** You can use some basic graph traversal algorithms such as DFS or BFS[1]. You can also use some algorithms that can find the single-source-shortest-path(SSSP), such as Dijkstra[2].
- You can use spark-xml (version 2.12) and graphframe (version 2.12) package to help you design the algorithm. You can not use any other libraries except those two packages and the java, hadoop and spark native libraries.

[1] https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/lecture-videos/MIT6_006F11_lec13.pdf

[2] https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-006-introduction-to-algorithms-fall-2011/lecture-videos/MIT6_006F11_lec16.pdf

Problem Description

- We support a tool name 'osmutil' to help you test your code. You need to choose the correct tool based on your current operating system. For example, if you are running your code on MacOS, you need to choose 'osmutil_mac'.
- You can use './osmutil help' to see the usage of the tool.

```
(base) → build.mac git:(master) X ./osmutils_mac help
Usage:
<...>: mandatory argument [...] : optional argument

Convert a .osm file to a adjmap file:
./osmutils_mac convert-map <map-file> [adjmap-file]
* map-file: must be an .osm map
* adjmap-file: defaults to 'result.adjmap.txt'

Validate an adjmap file:
./osmutils_mac validate-map <map-file> <adjmap-file>
* map-file: must be an .osm map
* adjmap-file: the adjmap to validate

Run dijkstra on a map given input file:
./osmutils_mac dijkstra <map-file> <input-file> [result-file] [result-gpx-dir]
* map-file: must be an .osm map
* input-file: each line is a pair of map node IDs
* result-file: defaults to 'result.txt'
* result-gpx-dir: defaults to 'result-gpx'

Validate paths:
./osmutils_mac validate <map-file> <result-file> [result-gpx-dir]
* map-file: must be an .osm map
* result-file: the results to validate
* result-gpx-dir: defaults to empty (not generating)
                  will generate gpx files if argument is provided

Print this help:
./osmutils_mac help
```

Problem Description

- `./osmutil convert-map <map-file> <adjmap-file>` will output the map file in adjacent format. Your code should output the same map file if you want to pass the test case.

```
(base) → build.mac git:(master) ✗ ./osmutils_mac convert-map ../data/NUS.osm
Map has: 9817 vertices, 18486 edges.
(base) → build.mac git:(master) ✗ head -n 10 ./result.adjmap.txt
148118794 1362160517
148118888 6039364480
148118909 7930212924
148118925 4723145375
148118965 566531260
148118993 1358272907 1362055632
148119004 4723141571
148119032
148144578 2389088990
148144751 4723147230
```

Problem Description

- You can use `./osmutil validate-map <map-file> <adjmap-file>` to check whether your output of the map is correct.
- If your result is correct, the tool will print 'valid'.
- If your result is wrong, the tool will print 'invalid' along with your problems.

```
(base) → build.mac git:(master) X ./osmutils_mac validate-map ../data/NUS.osm ../result.adjmap.txt  
Map has: 9817 vertices, 18486 edges.
```

Valid.

Your output is correct.

```
(base) → build.mac git:(master) X ./osmutils_mac validate-map ../data/NUS.osm ../result.adjmap.txt  
Map has: 9817 vertices, 18486 edges.
```

Missing vertices: 148146073

Invalid. Found 1 errors.

Your output misses one vertex.

Problem Description

- `./osmutil dijkstra <map-file> <input-file>` will find a pseudo-path for each nodes pair in the input file. Your code should give a path same to or shorter than the pseudo-path.

```
(base) → build.mac git:(master) X ./osmutils_mac dijkstra ../data/NUS.osm ../data/input-NUS.txt result.txt
Map has: 9817 vertices, 18486 edges.
Path 1345143432 -> 1854366392 found
Path 1345143432 -> 1832488252 found
(base) → build.mac git:(master) X cat result.txt
1345143432 -> 5131711927 -> 496801425 -> 1832517056 -> 1832517057 -> 496801423 -> 1832517073 -> 496801421 -> 1832517061 -> 1345143390 -> 1832517063 ->
1832517069 -> 7927504651 -> 7927504650 -> 7927504649 -> 7927504648 -> 7927504647 -> 7927504646 -> 7927169989 -> 496801413 -> 1832513974 -> 1345143448
-> 1832513976 -> 239713813 -> 1832513978 -> 1345143424 -> 1832513980 -> 239713812 -> 1832513985 -> 1345143372 -> 1832513986 -> 7927159376 -> 18325139
89 -> 1345143343 -> 1832513991 -> 239713810 -> 1555047419 -> 1345143437 -> 1601408309 -> 1832513993 -> 1832513995 -> 1601405578 -> 7919717647 -> 79197
17645 -> 1534827400 -> 239712264 -> 7919717642 -> 7919717646 -> 239712263 -> 1534825218 -> 1534825214 -> 4724763212 -> 1556605806 -> 239712262 -> 4724
763211 -> 5243866724 -> 6912796541 -> 1736138338 -> 1345143345 -> 239712261 -> 1534825221 -> 1345143342 -> 4724810956 -> 4724810957 -> 416342635 -> 21
70644532 -> 239712259 -> 610183475 -> 610183473 -> 1345143429 -> 239712258 -> 6114545165 -> 5282980941 -> 687603784 -> 5137605421 -> 239624382 -> 1358
272528 -> 4724827744 -> 1358272552 -> 5137604020 -> 1358272569 -> 2489562875 -> 239712257 -> 1358272605 -> 687603782 -> 1358272624 -> 1358272634 -> 15
53228162 -> 1553269896 -> 1553228152 -> 1553269893 -> 1553228128 -> 1553228134 -> 1553228113 -> 1553228156 -> 1553228163 -> 1553228159 -> 1553228143 ->
> 7930174104 -> 1553228149 -> 4362557691 -> 1854366366 -> 1358272689 -> 5136423038 -> 6957615527 -> 1854366370 -> 1854366372 -> 1854366374 -> 18543663
76 -> 6957615526 -> 1854366378 -> 1854366385 -> 1854366386 -> 1854366387 -> 1854366388 -> 1854366389 -> 5286734106 -> 1854366390 -> 5286734104 -> 1854
366392
1345143432 -> 5131711927 -> 496801425 -> 1832517056 -> 1832517057 -> 496801423 -> 1832517073 -> 496801421 -> 1832517061 -> 1345143390 -> 1832517063 ->
1832517069 -> 7927504651 -> 7927504650 -> 7927504649 -> 7927504648 -> 7927504647 -> 7927504646 -> 7927169988 -> 7927169987 -> 7927169986 -> 792716998
5 -> 7927159384 -> 7927159383 -> 7927159382 -> 7927159381 -> 7927159380 -> 7927159379 -> 7927159378 -> 7927159377 -> 1748955727 -> 1748955734 -> 17489
55737 -> 1748955732 -> 1748955725 -> 1748955714 -> 1748955712 -> 1748955706 -> 1748955703 -> 977377433 -> 7936899673 -> 977370345 -> 6275692195 -> 513
6287745 -> 1867527121 -> 1345143326 -> 8011567625 -> 7936899663 -> 8011567624 -> 1867527123 -> 1867527125 -> 977383402 -> 1345143441 -> 361030526 -> 1
345143402 -> 976478207 -> 1832488294 -> 496810111 -> 1345143428 -> 496810112 -> 1832488295 -> 687603003 -> 496810113 -> 1832488293 -> 1832488292 -> 18
32488291 -> 1832488290 -> 1832488289 -> 1832488288 -> 1832488287 -> 1832488286 -> 1832488285 -> 7957586760 -> 1832488284 -> 1832488283 -> 3568629948 ->
> 1838839670 -> 3568629945 -> 1832488282 -> 1832488281 -> 1832488280 -> 1832488279 -> 1832488278 -> 1832488277 -> 496810131 -> 1832488276 -> 183248827
4 -> 496810132 -> 4487888850 -> 3079562407 -> 496810134 -> 1832488261 -> 1832488259 -> 1832488257 -> 7958078027 -> 1832488252
```

Problem Description

- You can use `./osmutil validate <map-file> <result-file>` to check whether your output of path is valid, i.e., there does exist such path.
- If your result is valid, the tool will print the line that begins with 'valid' and followed by the length of your result path.

```
(base) → build.mac git:(master) ✗ ./osmutils_mac validate ../data/NUS.osm ../result.txt
Map has: 9817 vertices, 18486 edges.
valid: 1803.296789
valid: 1488.056660
```



The length for the first path

Problem Description

- You can use `./osmutil validate <map-file> <result-file>` to check whether your output of path is valid, i.e., there does exist such path.
- If your result is invalid, the tool will print the line that begins with 'invalid'.

```
(base) → build.mac git:(master) x ./osmutils_mac validate ../data/NUS.osm result.txt  
Map has: 9817 vertices, 18486 edges.  
invalid.  
valid: 1482.056660
```

The output of first path is invalid.

Submission Requirements

- Deadline: **Apr. 3rd**
- Submit the file **FindPath.java** into the folder AssignmentSubmission/Assignment2 in Luminus.
 - When submitting, make sure you only modify and submit the file FindPath.java.
- Policy on late submission:
 - Code submitted after the deadline but **no more than 48 hours** after the deadline will still be graded, with a **penalty of 20%**, i.e., multiple 80% of your original mark.
 - Code submitted more than 48 hours after the deadline will not be accepted and will get **0 mark**.

Marking

- All the codes will be **automatically compiled and marked** by similar scripts as 'compile_run' using ten test cases
 - The test cases will contain the five public cases we give to you and five private test cases that are **simpler** than the public test cases.
- So, ensure your codes can be compiled by the script in your package.
- **All submitted codes will be auto-checked for plagiarism.** Do NOT copy others' codes or share your codes with others.
- The environment we use for marking will be the same as the one you use for testing your code, i.e., a cluster with ten servers.

Marking

- Full mark: 15%.
- If you output the correct map file, you will get **5 points**.
- We will use **ten** test cases to test your path finding algorithm, each test case will worth 1 point.
- For each test case, you will get 0 point if your path is invalid. Else, your final mark M will be graded based on the length of your result path l_1 and the length of the pseudo-path l_2 , i.e.,
$$M = \min(0.8 + \frac{l_2}{l_1} * 0.2, 1)$$

Notice

- We have zero-tolerance on plagiarism.
- Do not “Copy and paste” from others.
- Do not share your code with others.

Feedbacks are Welcome

- Email: Mr. Yuhang Chen, e0546081@u.nus.edu
Mr. Sixu Hu, e0409758@u.nus.edu
- Or, post your questions in the LumiNUS forum (preferred).
- You may find something useful from FAQ of last semester, which can be found in the attached student guide. Note: we have change the format of our assignment this year (thus this FAQ is just for your reference).