Finding a Shortest Path with An Energy Budget
23S1 CZ3005 Assignment 1

1. Problem description

We are given a problem of finding a shortest traverse path between two locations within a certain energy budget. This scenario often happens in the transportation where a shorter path is not necessarily more energy efficient (e.g., the road may be very bumpy or has heavy traffic). We illustrate our problem with the following example:

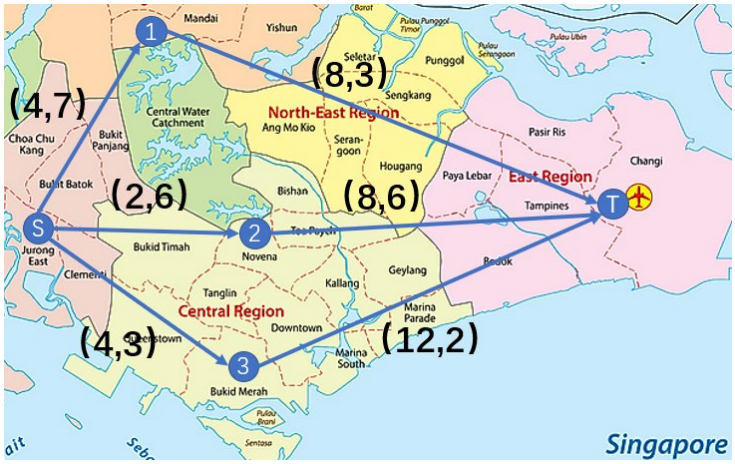

Figure 1: A small example.

Suppose we are given a graph with a predefined source node (S) and a target node (T). Each edge in the graph is associated with a pair of values (X, Y) where X denotes the distance and Y denotes the energy cost. Also see Figure 1 for an illustration. The job is to find the shortest path between S and T such that the accumulated energy cost along the path does not exceed an energy budget. Specifically, we want to find the shortest path from Jurong East to Changi Airport not exceeding an energy budget 11. We can observe that although the path S->2->T has the shortest travel distance 10, its accumulated energy cost 12 exceeds the energy budget 11. Thus, this path is infeasible. In this example, the path S->1->T is the shortest feasible path.

1.1 Problem instance

You will be given a problem instance of New York city (NYC) road network of the United State. This instance is taken and modified from the 9th DIMACS implementation challenge. This instance has 264346 nodes and 730100 edges. You will be given four python dictionary files:

1.1.1 Graph dictionary **G**
The graph is given as an adjacency list where the neighbor list of node 'v' can be accessed with **G**['v'].

1.1.2 Node coordination dictionary **Coord**
The coordination of a node 'v' is a pair (X, Y) which can be accessed with **Coord**['v']

1.1.3 Edge distance dictionary **Dist**
The distance between a pair of node (v, w) can be accessed with **Dist**['v,w'].

1.1.4 Edge cost dictionary **Cost**
The energy cost between a pair of node (v, w) can be accessed with **Cost**['v,w'.]

2. Requirements and guidelines

2.1 Tasks and marking criteria

You will need to solve three tasks that are listed as follow:

Task 1: You will need to solve a relaxed version of the NYC instance where we do not have the energy constraint. You can use any algorithm we discussed in the lectures. Note that this is equivalent to solving the shortest path problem. The solution quality of your algorithm will affect the grade.

(30 marks)

Task 2: You will need to implement an uninformed search algorithm (e.g., the DFS, BFS, UCS) to solve the NYC instance.

(30 marks)

Task 3: You will need to develop an A* search algorithm to solve the NYC instance. The key is to develop a suitable heuristic function for the A* search algorithm in this setting. The solution quality of your algorithm will affect the grade.

(40 marks)

For tasks 2 and 3, the energy budget is set to be 287932. For all the 3 tasks, the starting and ending nodes are set to be '1' and '50', respectively.

(Please note that the NYC instance is provided separately as formatted in section 1.1)

2.2 Output format

The output of your algorithm should be formatted as the following (using Figure 1 as an example):

Shortest path: S->1->T.
Shortest distance: 12.
Total energy cost: 10.

2.3 Programming language

For the sake of running your codes, you are required to use Python or Java. Please name your program file "main.py" or "main.java" and put the file under the main folder named with your team's name. You should also put the instance files into your main folder. You must make sure your codes are runnable. The TAs may run your codes by simply executing the command "python main.py" or "java main". All the algorithms must be implemented by you.

3. Teaming Formation

You are required to form a team of up to 3 people to complete this project. A team with only a single person is also allowed. For team size that is larger than one, you need to clearly state the contribution of each team member.

4. Honor Code

If you use any existing code, libraries, etc. and consult any papers, books, online references, etc. for your project**, you must cite your sources in your report and clearly indicate which parts of the project are your contributions and which parts were implemented by others.** Using others' code/ideas without acknowledgement will lead to failure of your project.

5. Deliverables

You are required to produce:

(1) A final report of no more than 5 pages. Name your report as *<team name>-final-report.pdf*.

(2) Source codes for your project. Your source codes are to be submitted in .zip format. Name the zip file as *<team name>-final-code.zip*.

The following is a suggested structure for the report:

(1) Title, team members.

(2) For each task, explain your approach, output, etc.

(3) Contribution: Please clearly state the contribution of each team member if you have more than one member in your team.

(4) Conclusion: What have you learned?

(5) References: Please cite the sources you consulted in the report.

6. Submission

Please email your two files (*<team name>-final-report.pdf* and *<team name>-final-code.zip*) to the TA for your lab group one month after your lab session in week 5 or week 6. For instance, if your lab session is 10:30am-12:30pm on September 7, please submit your files to your TA by 23:59pm of October 7.

You can find the TA and TA's email in the table below. If you have questions, please ask questions during the lab session or email your TA.

Teaching Assistant Assignment

Lab Groups	TA
TDDB, TDDC, TACD, TCCA	Li Zongrui: ZONGRUI001@e.ntu.edu.sg
TDDA, TEL3, TEL1, TCMA	Wu Haoning: HAONING001@e.ntu.edu.sg
TEL4, TEL6	ZANG SHENG: ZANG0015@e.ntu.edu.sg