Problem Definition

In recent days, transportation applications to move from one place to another have been evolving rapidly. The use of data from many users helps to make transportation better. Especially in reducing transportation time and avoiding heavy traffic on roads.

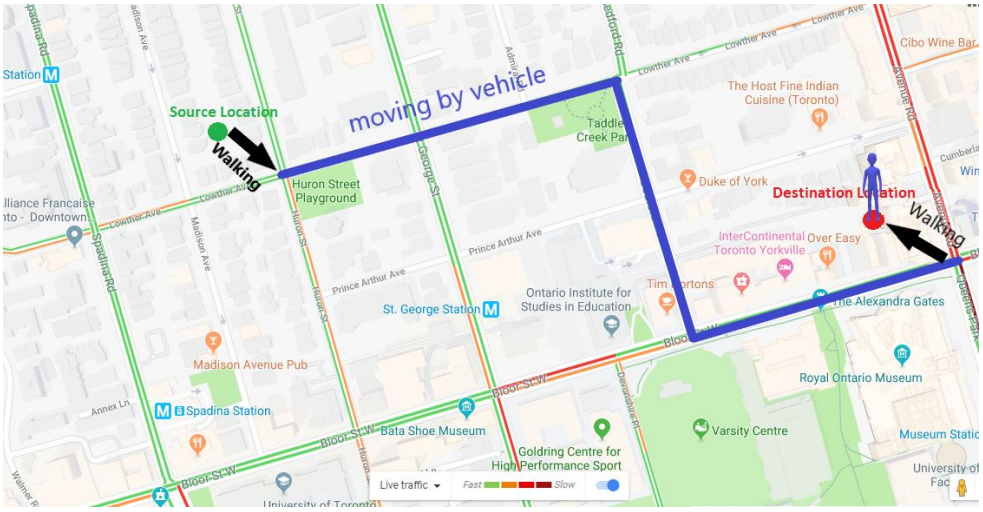
In this project, **you are asked to find the least time** to move from the source location to the destination location. In addition, **find the path** that achieves that time.

The map is represented as a graph. Intersections are represented by nodes and roads are represented by edges. Each road has a speed (in km/h). You may assume that all vehicles moving on the road are moving at that speed.

The moving person can:

1. walk for at most **R** meters from the source location to get to the starting intersection.
2. Then ride a vehicle to move to another intersection close to the destination location.
3. Finally, walk from the intersection to the destination location for at most **R** meters.

The walking speed is always **5 km/h** (from the source location to the nearby intersection and from the intersection near to the destination to the destination location). Example is shown below:

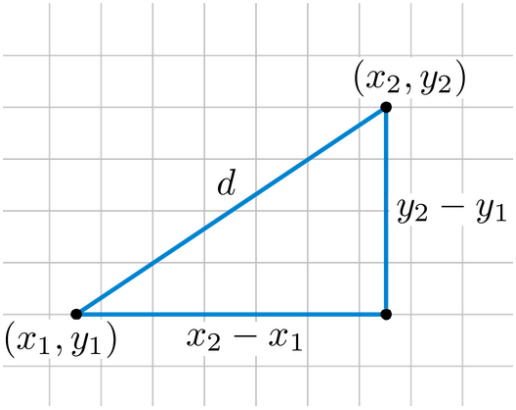


Terminologies

Euclidean Distance

The Euclidean distance or Euclidean metric is the straight-line distance between two points in Euclidean space. In the Euclidean plane, if the first point is p1 = (x1, y1) and the second point p2 = (x2, y2) then the distance between the two points is given by:

$$d(p1, p2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$



Graph construction

To construct the graph, we need to define:

1. Vertices (nodes): which represent the intersections.
2. Connectivity (edges): which represent roads that connect intersections.

The graph is **a weighted graph** (since the time to move through roads differs between roads).

All the roads are **2-way roads**.

Problem Details

- The moving person walks with constant speed (5 km/h) in straight lines (from the source to the starting intersection (node) and from the finishing intersection (node) to the destination).
- The moving person cannot walk more than R meters to move from the source location to the starting intersection. They also cannot walk more than R meters to move from the finishing intersection to the destination location.
- The time to get in a vehicle or get out of a vehicle is negligible (not taken into consideration).
- The moving person will ride only one vehicle. S/he cannot get out of the vehicle unless s/he reaches the final node.
- The vehicle always moves at road speed. It changes speed only if the road is changed.

Project Requirements

Required Implementation

Requirement	Performance
1. Construct the graph for the given map.	**Time:** should be **bounded by O(|E|)** |E|: # of roads
2. Search for possible starting nodes (intersections) (**S**).	
3. Search for the possible finishing nodes (intersections) (**F**).	
4. Find the best path to move from a starting intersection to a finishing intersection (the path of the least time).	**Time:** should be **bounded by O(|S| |E`| log |V`|)** |S|: number of starting nodes |V`|: # of intersection points that are checked until reaching the destinations |E`|: # of roads that are checked until reaching the destinations
5. Construct the best-time path	**Time:** should be **bounded by O(|V`|)** V': # of intersection points that are checked until reaching the destinations

Input

1. **Map file:**

   Which contains the intersections' locations and how the intersections are connected to each other. Each map file is organized as the following (example is shown in the figure below):

   a. The first line contains an integer **N** which represents the **number of intersections**.

   b. Each line of the following N lines contains 3 numbers (separated by single space):

   Intersection_ID X_coordinate Y_coordinate

   c. After that there is an integer **M** which represents the **number of roads**.

   d. Each line of the following M lines contains 4 numbers (separated by single space):

   First_Intersection_ID Second_Intersection_ID Road_Length Road_Speed

   Note: the road length is given in **kilometers** and road speed is given in **kilometer/hour**.

```
6
0 0.25 0.33
1 0.73 0.47
2 1.00 0.54
3 0.34 0.60
4 0.81 0.20
5 1.07 0.28
7
0 1 0.50 20
0 3 0.28 80
3 4 0.62 80
1 4 0.28 40
1 2 0.28 40
4 5 0.27 60
2 5 0.27 60
```

2. **Queries file**:

   Each query file is organized as follows(example is shown in the figure below):

   a. The first line of this file contains an integer **Q** that represents the number of queries.

   b. Each line of the following Q lines contains 5 numbers (separated by single space) represents a single query as the following:

   Source_X Source_Y Destination_X Destination_Y R

Note: the maximum walking distance (**R**) is given in **meters.**

```
3
1.01197 0.480795 0.331957 0.343315 253
0.389941 0.456355 0.911841 0.249623 241
0.251276 0.419062 0.929659 0.204712 346
```

Output

The **output file** should contain the result of the **Q** queries that are in the **query file**. (example is shown in the figure below)

```
2 1 0
4.06 mins
0.92 km
0.14 km
0.78 km

3 4
5.04 mins
0.89 km
0.27 km
0.62 km

0 1 4
4.43 mins
0.99 km
0.21 km
0.78 km

0 ms

2 ms
```

- Each **query output** should contain 5 lines as the following:
  1. **The IDs of the intersections** of the **vehicle path** with the shortest time. The ids should be ordered by the visiting order and **separate by space**.
     EX: 0 3 4 5 2
  2. **The shortest time** to move from the source location to the destination location (**in minutes**).
     EX: 4.63 mins
  3. **The total distance** of the path with the shortest time **(in kilometers).**

EX: 1.72 km

4. **The total walking distance (in kilometers).**
   EX: 0.28 km
5. **The total vehicle distance (in kilometers).**
   EX: 1.44 km

- **ALL values** should be displayed **rounded** to **EXACTLY 2 DIGITS** after the decimal point.
- **Each query** should be **followed** by an **empty line**
- After printing the result of all queries, print the **total execution time without Input/output**
  - Start query time after reading the query until calculating the query output – without Input/output times" (in milliseconds) followed by an empty line.
    EX: 1 ms
- At the end, print the **total execution time** of the **program** "Including Input/output times" (in ms)
  EX: 5 ms

Use text-comparator program like beyond compare to check

How to calculate execution time?

To calculate the execution time of certain piece of code:

1. Define an object from **`Stopwatch`**
2. **`Start()`** the stopwatch before the code
3. **`Stop()`** it after the code
4. Access the **`ElapsedTime`** to get the time of your code

Test Cases

1. Sample Cases

to debug on it (for correctness)

- The number of intersections will not exceed 20 intersections.
- The number of roads will not exceed 50 roads.
- The number of queries will not exceed 10 queries.

2. Medium Cases

for final massive testing (efficiency - beside correctness)

- The number of intersections will not exceed 20000 intersections.
- The number of roads will not exceed 25000 roads.
- The number of queries will not exceed 1000 queries.

3. Large Cases

for final massive testing (efficiency - beside correctness)

- The number of intersections will not exceed 200000 intersections.
- The number of roads will not exceed 250000 roads.
- The number of queries will not exceed 1000 queries.

Refer to the attached document for more details about each case

Deliverables

Implementation (60%)

1. Graph construction.
2. Efficient search for the best path (with the shortest time).
3. Best path building.

Document (40%)

1. Description and code of graph construction.
2. Code of the shortest path algorithm.
3. Detailed analysis of the above codes.

Allowed Codes

- Data structures code, other than the graph (either C# built-in data structures or other open-source data structures). You **MUST** **understand** and **analyze** it!

Important Delivery Notes

1. **Test Cases & Output Files:**
   - You must read the **input file** and write the **output in file** with the **same format** specified in the project document. You will not be allowed to change anything in the file during delivery time.
   - No manual input will be allowed.
2. **Documentation:**
   - You must get the **document printed** (softcopy is not accepted).
   - **Analysis** must be **detailed** (don't write the whole complexity only).
   - In case of using any **built-in/external data structures** or **functions**, you must **analyze them**.

Milestones

	Deliverables	Due to
Milestone1 (final delivery)	1. Construct a weighted graph for the map. 2. The search for the optimal path to move from the source location to the destination location. 3. Building the optimal path (output the nodes in order). 4. The algorithm should run on all cases (sample, medium , and large) 5. Documentation	[LAB EXAMS WEEK]
For Milestone1: ○ **MUST** deliver the required tasks and **ENSURE** it's worked correctly ○ **MUST** deliver in your scheduled time (TO BE ANNOUNCED)		

BONUSES

1. **Changing-speed handling:**
   In the case the speed is given in intervals. You have to consider the speed change in the fastest path calculation.

Refer to the attached document for more details about this bonus

2. **Map Visualization:**
   Visualizing the map data and the shortest path.

3. **Faster implementation**:
   for the shortest path to be less than the given bounded complexity above, (less than **O(S E` log V`)).**