Semester 2 2014
COMP3702/7702 ARTIFICIAL INTELLIGENCE
ASSIGNMENT 1

Note:

- You can do this assignment individually or in a group of 2-3 students.
- For those who choose to work in a group:
  - All students in the group must be enrolled in the same course code, i.e., all COMP3702 students or all COMP7702 students.
  - Send the group name and members (name and student ID) to comp3702-staff@itee.uq.edu.au with subject **group for Assignment1** before **11.59pm on Monday, August, 18th**. If we do not receive your e-mail by then, you will need to work on the assignment individually.
- Please submit your source code by e-mail to comp3702-staff@itee.uq.edu.au with subject **Assignment1** before **11.59pm on Sunday, September, 7th**.
  - IMPORTANT: Please submit only the source code (excluding object files or binaries).
  - If your code is more than one file, please put them in one folder named assg1_studentID(s) and send the folder as a zipped file.
  - If your code consists of only one file, please name it assg1_studentID(s).[extension].
- Please choose a time slot to demo your program. Stay tuned for the exact time and venue.
- For the demo, you can use your own laptop or your "favourite" desktop in the lab, but you need to use the program that you submitted on September 7th.
- Please submit a hardcopy format of your report by **midday on Thursday, September, 11th** via the assignment chute at Hawken. You need to use the assignment cover sheet (https://student.eait.uq.edu.au/coversheets/?pid=67354).

Congratulations!!! You have been hired by UQ Inc. to work on their most recent product: RiverCleaner. RiverCleaner is a robotic solution to clean rivers of floating trash. It consists of multiple Autonomous Surface Vehicles (ASVs) carrying steel-mesh booms (e.g. Fig. 1). Your task is to develop the motion planning software for the ASVs, so that RiverCleaner's movements always satisfy the requirements set by UQ Inc. Environmental Engineering Unit (UQ-EEU).

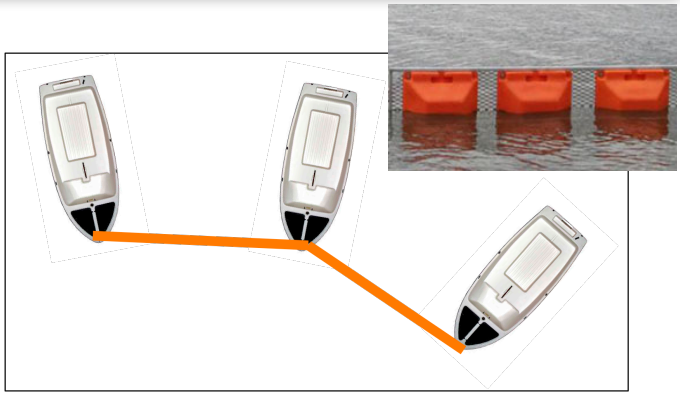

Fig. 1. A sketch of RiverCleaner. Autonomous Surface Vehicles (ASVs) carry steel-mesh booms (the orange lines, picture at upper right).

Before discussing the requirements of UQ-EEU, let's first simplify the problem as follows:

1. Each ASV is modeled as a point, each boom as a straight line segment, and the environment as a normalized 2D plane ([0,1]X[0,1]).
2. You only need be concerned about the positions of the ASVs (i.e., velocity, acceleration, momentum, etc. can be ignored).
3. The environment is fully observable, deterministic, static, and continuous.
4. All obstacles in the environment have rectangular form, and are axis-aligned.
5. The cost of moving from one configuration to another is the sum of the straight-line distance traveled by each ASV.

The requirements for the RiverCleaner as set by UQ-EEU are:

1. The length of each boom is fixed at 0.05 units in length.
2. The system must form a connected chain at all times. A connected chain means each ASV can be connected to at most two booms and each end of each boom is tied to an ASV.
3. The polygon formed by connecting the two ends of the connected chain must, at all times, be convex and have an area of at least πr_{min}^2, where $r_{min} = 0.007(n\text{-}1)$ and n is the number of ASVs. This geometry is needed to ensure the trash that has been contained does not spill out again.
4. The booms must never intersect with each other.
5. Booms and ASVs must never intersect with obstacles.
6. Booms & ASVs cannot move outside the [0,1]X[0,1] workspace.
7. The planned path for RiverCleaner must be given as a sequence of positions (primitive steps) such that on each step, each individual ASV moves by a distance of at most 0.001 units.
8. Each requirement (1-6) must hold for each primitive step. Since the distances are very small (at most 0.001 unit length for each ASV), it is sufficient to test the requirements only at the end of each primitive step.

Your task is to develop a program that finds a collision free path for the ASVs to move from a given initial configuration to a given goal configuration, while satisfying the above requirements. In relatively open environments, the path must be found in under 30sec (average of 5 runs) on a PC with Intel Core i7-2600 3.4GHz and 16GB RAM (e.g., any PC in 78-116). In cluttered environments, the path must be found in under 2 minutes (average of 5 runs) on the same PC.

The following steps denote one possible approach to developing a solution:

1. Define the state space, action space, world dynamics, and utility function of the problem.
   Hint: This is your chance to make the problem easier to solve.
2. Define a strategy to generate a state graph representation of the problem definition in (1) and implement this strategy. If you use a Probabilistic Roadmap (PRM) approach, be careful with your choice of sampling strategy (esp. if you want to get > 5).

3. Implement a Uniform Cost search on the state graph to find a collision-free path that satisfies the requirements set by UQ-EEU.

We have provided functions to perform some of the the necessary primitive computations, e.g. tests for collisions, convexity, and configuration area, in Java. If you do not use Java, you can copy those algorithms and rewrite the functionality in the language of your choice. In your implementation, you may use libraries for data structures (e.g. priority queues, stacks, hashmaps) – no other software libraries are allowed.

Submission information

You need to submit your solution in the form of source code and a report. The report must be at most 3 A4 pages, and should contain:

- A description of the proposed method.
- An explanation why you think the proposed method is a good solution.
- An explanation of the limitations of the proposed method.

The software should take a text file as input, and output a solution path to another text file. The format of the input and output files are as follows.

Input format:

a. The file consists of k+4 lines, where k is the number of obstacles. The first line is n, the number of ASVs. The next two lines are the initial and goal configurations. The next line is k, the number of obstacles. Each line in the last k lines represents the vertices of each rectangular obstacle.

b. The initial & goal configurations are written as x-y positions of each ASV. Boom-1 is attached to ASV-1 and ASV-2, ..., boom-(n-1) is attached to ASV-(n-1) and ASV-n.

c. Each rectangular obstacle is written as a quadruple of the x-y coordinates of its vertices in counter-clockwise order, starting with the lower-left vertex.

d. Example of an input file:

```
2
0.3 0.5 0.36 0.5
0.6 0.5 0.66 0.5
2
0 0.2 0.2 0.2 0.2 0.4 0 0.4
0.5 0.6 0.7 0.6 0.7 0.9 05 0.9
```

The input file means:

- The system consists of 2 ASVs (and 1 boom).
- The initial positions for ASV-1 and ASV-2 are (0.3, 0.5) and (0.35, 0.5).
- The goal positions for ASV-1 and ASV-2 are (0.6, 0.5) and (0.65, 0.5).
- There are two rectangular obstacles. The vertices of the first obstacle are (0, 0.2), (0.2, 0.2), (0.2, 0.4), (0, 0.4). The vertices of the second obstacle are (0.5, 0.6), (0.7, 0.6), (0.7, 0.9), (0.5, 0.9).

Output format:

The file consists of m+2 lines. The first line is the number of primitive steps, m, and the total length of the path, separated by a space. The next

line is the initial configuration, and then each line in the next *m* lines after that contains the position of each ASV at the end of every primitive step. For example:

300 0.6
0.3 0.5 0.35 0.5
0.301 0.5 0.351 0.5
:
0.4 0.5 0.45 0.5
0.401 0.5 0.451 0.5
:
0.5 0.5 0.55 0.5
0.501 0.5 0.551 0.5
:
0.6 0.5 0.65 0.5

The above output means ASV-1 moves from (0.3, 0.5) to (0.301, 0.5) to … to (0.6, 0.5), ASV-2 moves from (0.35, 0.5) to (0.351, 0.5) to … to (0.65, 0.5).

We have provided a Java program to visualize the input and output files.

Grading

Solution & program: 15%.
Report: 5%.

Marking for solution & software will be based on 4 test cases that we will give during demo time. The grading is as follows:

COMP3702:

- 4: Solve all 4 test cases for > 1 and <= 3 ASVs in open environments.
- 5: Solve all 4 test cases for > 3 and <= 10 ASVs in open environments.
- 6: Solve all 4 test cases for > 3 and <= 10 ASVs in cluttered environments.
- 7: Solve all 4 test cases for > 10 ASVs in cluttered environments.

COMP7702:

- 4: Solve all 4 test cases for > 1 and <= 3 ASVs in cluttered environments.
- 5: Solve all 4 test cases for > 3 and <= 10 ASVs in cluttered environments.
- 6: Solve all 4 test cases for > 10 and <= 15 ASVs in cluttered environments.
- 7: Solve all 4 test cases for > 15 ASVs in cluttered environments.

Marking scheme for the report:

COMP3702/7702:

- 4: Clear definition of the problem you are trying to solve (i.e., state space, action space, world dynamics, and utility function of the problem).
- 5: Requirements for getting a 4 + clear explaination of the concepts behind your proposed solution.
- 6: Requirements for getting a 5 + clear explaination of the limitations of your solution.
- 7: Requirements for getting a 6 + convincing arguments that the proposed solution is the most suitable, compared to other existing solutions.