CP468 – Artificial Intelligence- Group 5

Path Planning

Names:Junda Hui

ID: 134169380

Names:Zeqi Peng

ID: 150877050

Names:Joshua Jacob

ID: 100563670

Names:Yiyang zhang

ID: 184172400

2018/12/02

1. Design choices

We use python to solve the path planning problems. The robot is encapsulated into “Robot” class that implements pathfinding through A* algorithm. Each time an instance of the class is created. A sub-process is started to execute the A* algorithm. The following is a detailed introduction to our pseudocode, data structure, functions and methods.

1.1 Pseudocode

```
//OPEN-->CLOSE，  Initial point-->Any point-->rendezvous point

closedset := the empty set                  //Have passed point
openset := set containing the initial node //Pointns that the algorithm maybe will pass
 g_score[start] := 0                        //g(n)
 h_score[start] := heuristic_estimate_of_distance(start, goal)     //h(n)
 f_score[start] := h_score[start]

 while openset is not empty     //If openset is not empty
    x := the node in openset having the lowest f_score[] value //x is the smallest point in the openset
    if x = goal                                     //if x is rendezvous point
       return construct_path(came_from,goal)            //
    remove x from openset
    add x to closedset                          //add x to colsedset

 CLSOE SET
    for each y in neighbor_nodes(x)      //neighbor point do not exist other things like people
       if y in closedset
          continue
       tentative_g_score := g_score[x] + dist_between(x,y)

       if y not in openset
          add y to openset
          tentative_is_better := true
       else if tentative_g_score < g_score[y]
          tentative_is_better := true
       else
          tentative_is_better := false
       if tentative_is_better = true
          came_from[y] := x
          g_score[y] := tentative_g_score
          h_score[y] := heuristic_estimate_of_distance(y, goal)   //x-->y-
```

->goal

f_score[y] := g_score[y] + h_score[y]

return failure

1.2 Classes and Data structure

We saved all the robot information in the 'Robot' class. This class contains the instance variables "coordinate_tuple", "init_position", "path", and "rendezvous_point".

"coordinate_tuple" is a tuple that save all coordinates. We use the normal coordinates to save the coordinate information of the room instead of the Cartesian coordinates in the plane. When we read the matrix of stored information from the file, we convert it from the Cartesian coordinates to the normal coordinates.

"init_position" is a list that contains the coordinates of the initial position, like [2, 8].

"rendezvous_point" is a list save the coordinate of the rendezvous position, like [4, 7].

"path" is a list that contains the path from the robot initial point to rendezvous point.

1.3 Functions and Methods

There are three methods "find_path", "set_path" and "print_path" in 'Robot' class.

"find_path" is a method to find the path of the current robot from the initial point to the rendezvous point. In this method, we use the instance variables and A* algorithm to find a path, it returns the path to method "set_path".

"set_path" is a method to assign value to variable "self.path" by calling "find_path".

"print_path" is a method show result by standard format. If the robot has a path, it will output the result to the console. Then display the picture of the path, and finally save the path to the "output.txt" file. If no path exists, it will output a hint that no path exists to the console.

In addition, there are several functional functions in this program.

"read_info" is a function to read the room information from the test file.

"init_all" is a function used to initialize all instances of the Robot class, and create a subprocess for each instance to execute the A* algorithm.

"get_result" is a function used to call the "print_path" method and then output the result.

2. Instructions

2.1 Install

To run this program, the user must install Python 3.6 or later. Once Python 3.6 is installed, numpy, matplotlib.lines, multiprocessing and matplotlib.pyplot need to be installed on the computer. If pip has been installed with Python, simply open command prompt and type 'pip install numpy',' pip install matplotlib' and 'pip install multiprocessing' and numpy, matplotlib, multiprocessing should be installed.

2.2 Execute

The program can be run by running path_planning.py in an IDE like PyCharm. If you want to test with different files, first you need to put the file and "path_planning.py" in the same folder. Then, you should modify the file name in the second line under the "main()" function in the code. You should modify the parameter to the name of the file you want to test. The format of your document should be written in strict accordance with the standard format and examples of standard documents are provided in the following sections. After completing the above steps, you can run the program.

3. Results of our own test files

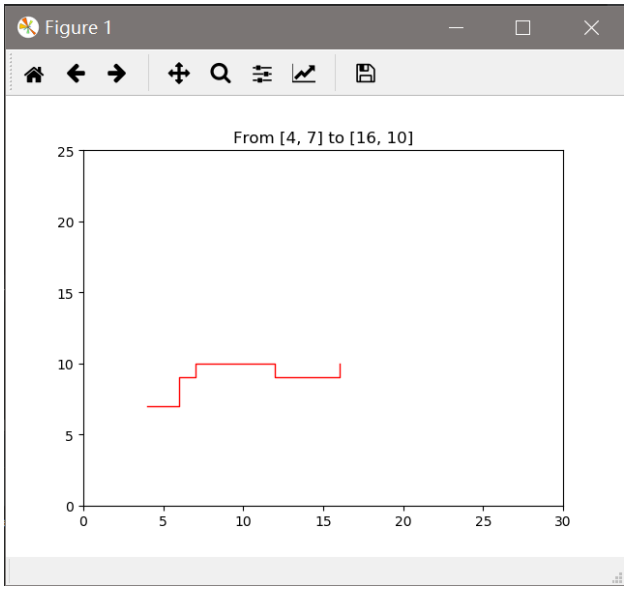
All of test fill use 25 * 30 room dimensions. The test results consist of the plot of the path, the output of the console and the input of the file

3.1 test1.txt

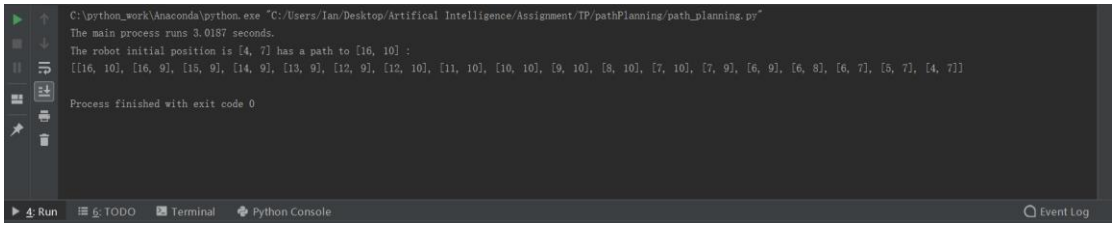
1 robot
1st robot Initial position: point (4,7)
The rendezvous point: point (16,10)

3.1.1 Result

The main process runs 3.0187 seconds.



Plot



Console

```
One robot path from [4, 7] to [16, 10]:
(4, 7) (5, 7) (6, 7) (6, 8) (6, 9) (7, 9) (7, 10) (8, 10) (9, 10) (10, 10) (11, 10)
(12, 10) (12, 9) (13, 9) (14, 9) (15, 9) (16, 9) (16, 10)
```

File

3.1.2 Test file

```
25 30
1
4 7
16 10
10000000011000000011001111001
11000000111100000111100000011
00000000001100000111100111100
10001100010000000001100111100
10011110011001111001000000110
00011110000000110000100000010
11000000111100000110000000001
00001100001100000000100000001
11000000111001111001100000110
10000000011000000010000000000
11000000111000000111000000110
00000000001100000110110011110
10001100010000000000000000111
10011110011001111001000001111
00011110000000110000100000110
11000000111100000111000011000
00001100001100000000100000110
11000000111001111001111000000
00001100001100000000001111000
11000000111001111001010000000
10000000011000000000000001100
11000000111100000111100000011
00000000001100000011011000100
00000000110000000010000000001
11000000111100000111000000110
```

3.2 test2.txt

2 robots

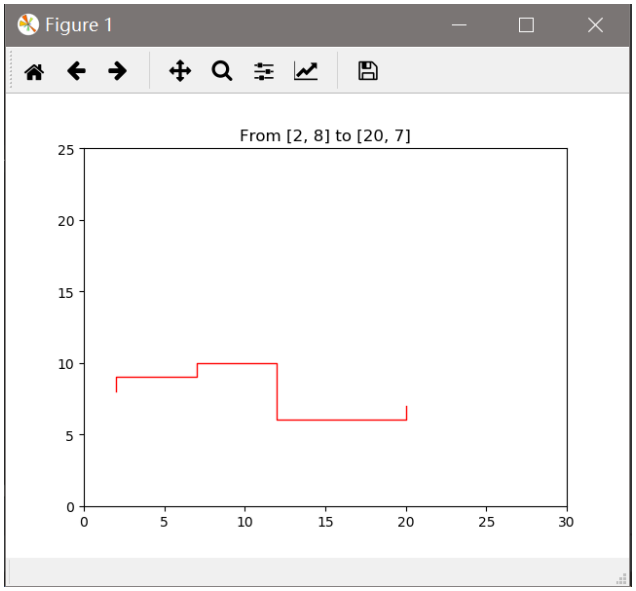
1st robot Initial position: point (2,8)

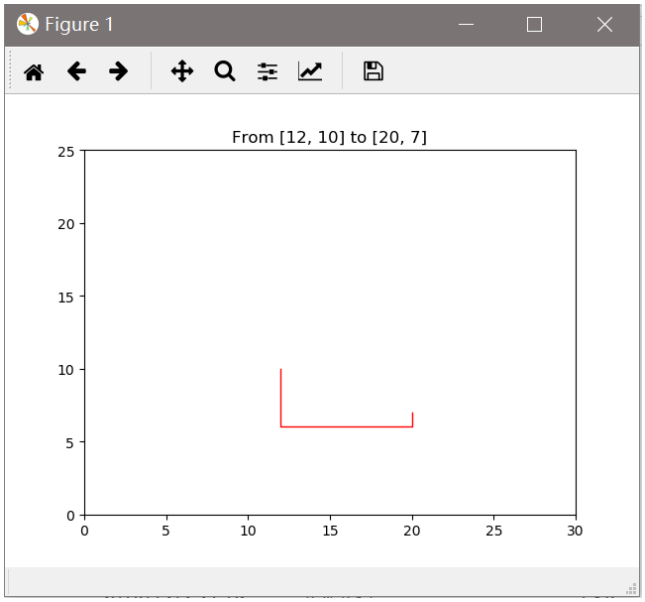
2nd robot Initial position: point (12,10)

The rendezvous point: point (20,7)

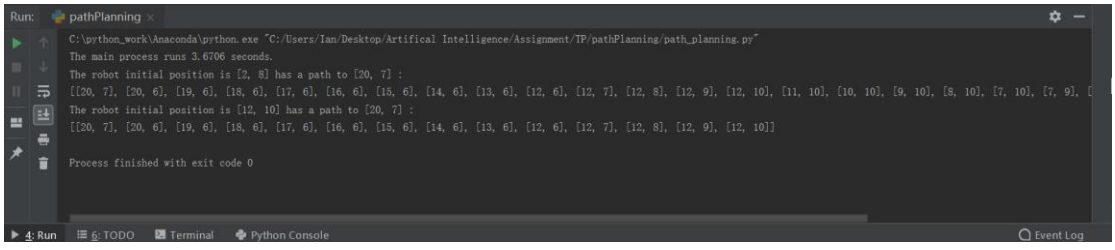
3.2.1 Result

The main process runs 3.6706 seconds.





Plot



Console

```
One robot path from [2, 8] to [20, 7]:
(2, 8) (2, 9) (3, 9) (4, 9) (5, 9) (6, 9) (7, 9) (7, 10) (8, 10) (9, 10) (10, 10) (11, 10) (12, 10) (12, 9) (12, 8) (12, 7) (12, 6) (13, 6) (14, 6) (15, 6) (16, 6) (17, 6) (18, 6) (19, 6) (20, 6)
(20, 7)
One robot path from [12, 10] to [20, 7]:
(12, 10) (12, 9) (12, 8) (12, 7) (12, 6) (13, 6) (14, 6) (15, 6) (16, 6) (17, 6) (18, 6) (19, 6) (20, 6) (20, 7)
```

File

3.2.2 Test file

```
25 30
2
2 8
12 10
20 7
100000000110000000011001111001
110000001111000000111100000011
000000000011000000111100111100
100011000100000000001100111100
100111100110011110010000000110
000111100000001100001000000010
110000001111000000110000000001
000011000011000000001000000001
110000001110011110011000000110
100000000110000000010000000000
110000001110000000111000000110
000000000011000000110110011110
100011000100000000000000000111
100111100110011110010000001111
000111100000001100001000000110
110000001111000000111000011000
000011000011000000001000000110
110000001110011110010111000000
000011000011000000000001111000
110000001110011110010100000000
100000000110000000000000001100
110000001111000000111100000011
000000000011000000110110001000
000000001100000000100000000001
110000001111000000111000000110
```

3.3 test3.txt

3 robots
1st robot Initial position: point (8,21)
2nd robot Initial position: point (12,7)
3rd robot Initial position: point (16,10)
The rendezvous point: point (2,4)

3.3.1 Result

The main process runs 3.4545 seconds.

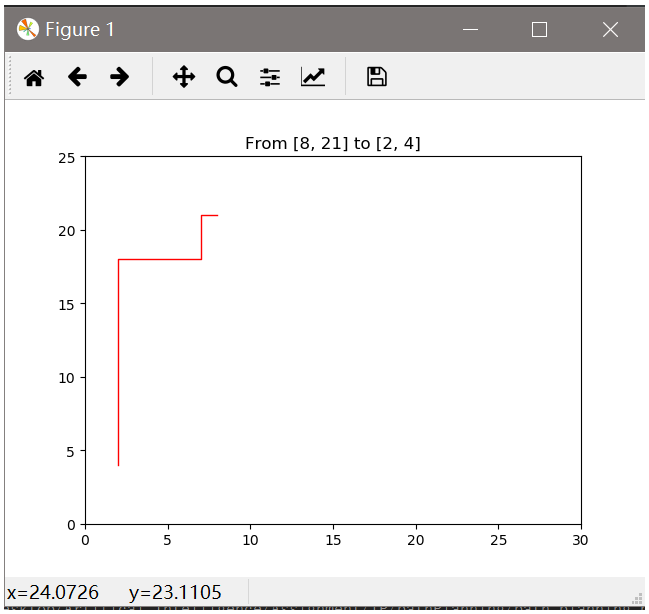


Figure 1
From [8, 21] to [2, 4]
x=24.0726 y=23.1105

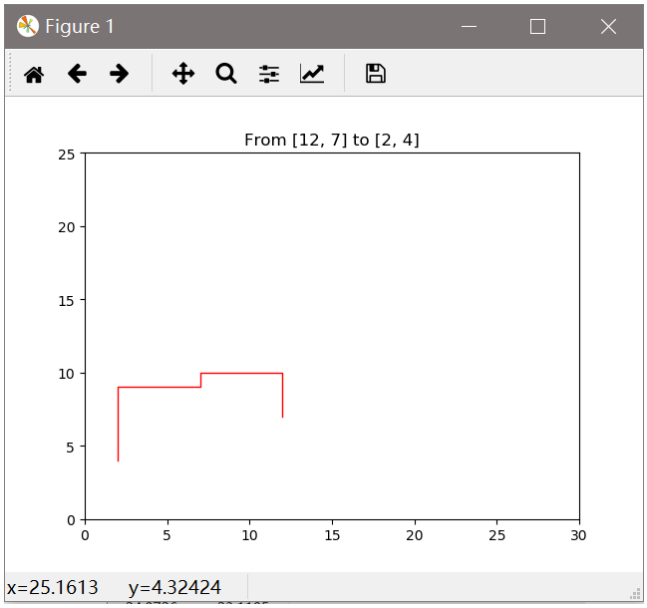


Figure 1
From [12, 7] to [2, 4]
x=25.1613 y=4.32424

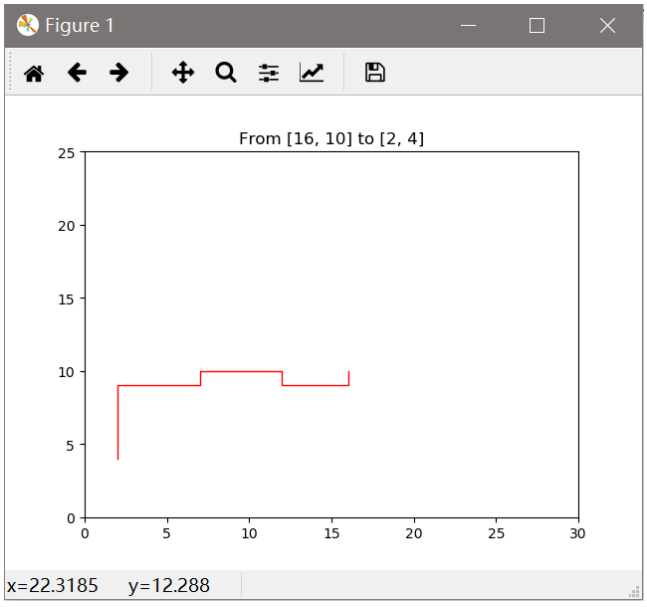
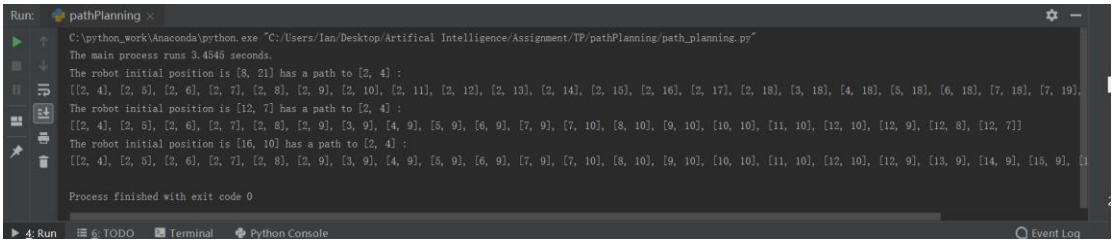


Figure 1
From [16, 10] to [2, 4]
x=22.3185 y=12.288

Plot



Console

```
One robot path from [8, 21] to [2, 4]:
(8, 21) (7, 21) (7, 20) (7, 19) (7, 18) (6, 18) (5, 18) (4, 18) (3, 18) (2, 18) (2, 17)
(2, 16) (2, 15) (2, 14) (2, 13) (2, 12) (2, 11) (2, 10) (2, 9) (2, 8) (2, 7) (2, 6)
(2, 5) (2, 4)
One robot path from [12, 7] to [2, 4]:
(12, 7) (12, 8) (12, 9) (12, 10) (11, 10) (10, 10) (9, 10) (8, 10) (7, 10) (7, 9)
(6, 9) (5, 9) (4, 9) (3, 9) (2, 9) (2, 8) (2, 7) (2, 6) (2, 5) (2, 4)
One robot path from [16, 10] to [2, 4]:
(16, 10) (16, 9) (15, 9) (14, 9) (13, 9) (12, 9) (12, 10) (11, 10) (10, 10) (9, 10)
(8, 10) (7, 10) (7, 9) (6, 9) (5, 9) (4, 9) (3, 9) (2, 9) (2, 8) (2, 7) (2, 6) (2, 5) (2, 4)
```

File

3.3.2 Test file

```
25 30
3
8 21
12 7
16 10
2 4
1000000001100000000011001111001
1100000011110000001111100000011
0000000000110000001111001111100
1000110001000000000011001111100
1001111001100111110010000000110
0001111000000001100001000000010
1100000011110000001100000000001
0000110000110000000010000000001
1100000011100111100110000000110
1000000001100000000010000000000
1100000011100000001110000000110
0000000000110000001101100111110
1000110001000000000000000000111
1001111001100111110010000001111
0001111000000001100001000000110
1100000011110000001110000011000
0000110000110000000010000000110
1100000011100111100111111000000
0000110000110000000000001111000
1100000011100111100101000000000
1000000001100000000000000001100
1100000011110000001111100000011
0000000000110000001101100011000
0000000011000000001000000000001
1100000011110000001110000000110
```

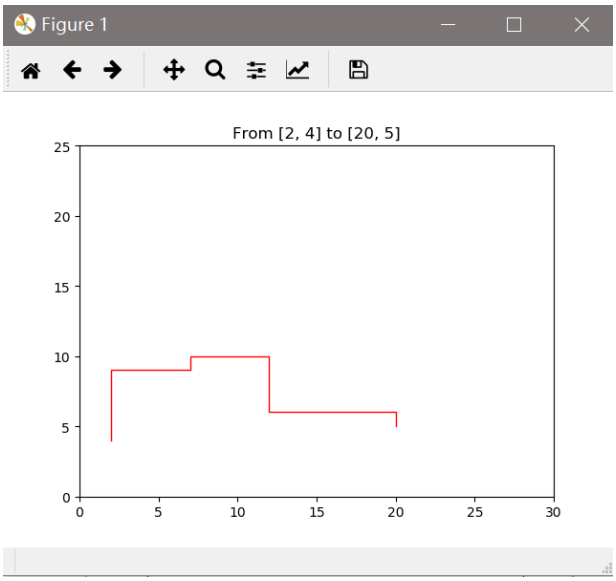
3.4 test4.txt

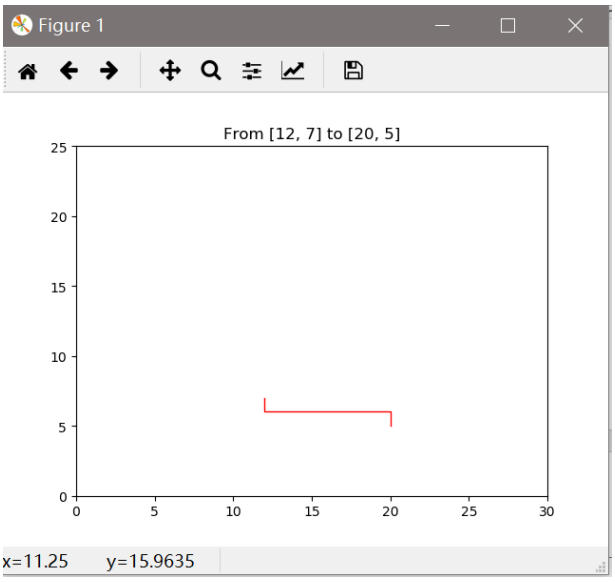
4 robots

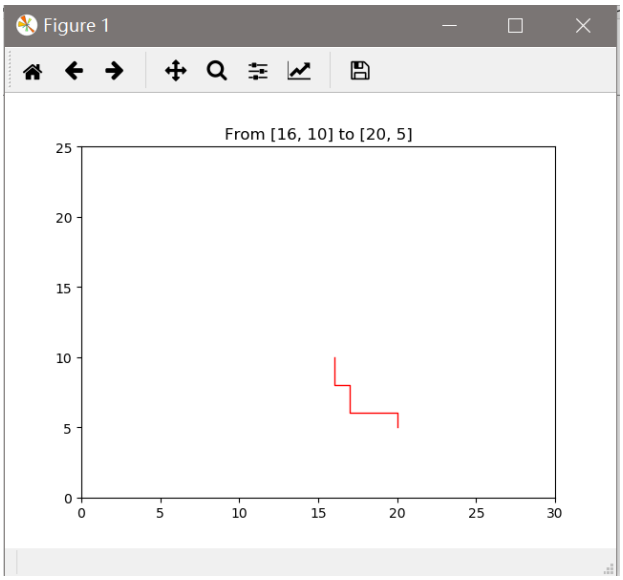
1st robot Initial position: point (2,4)
2nd robot Initial position: point (8,21)
3rd robot Initial position: point (12,7)
4th robot Initial position: point (16,10)
The rendezvous point: point (20,5)

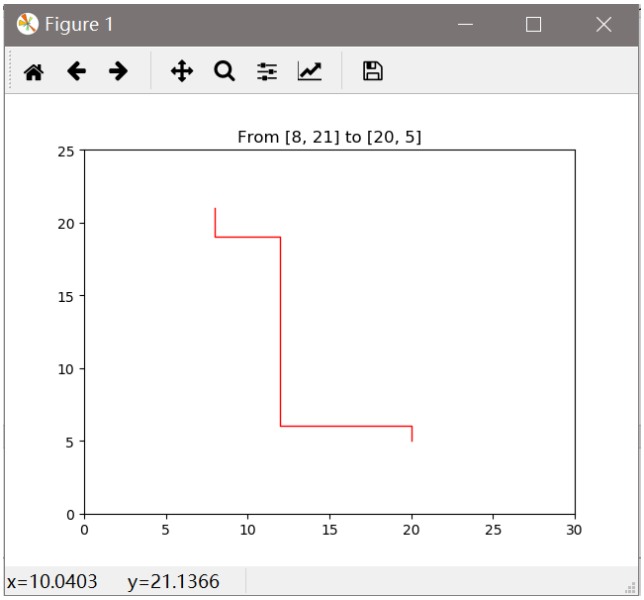
3.4.1 Result

The main process runs 3.3629 seconds.



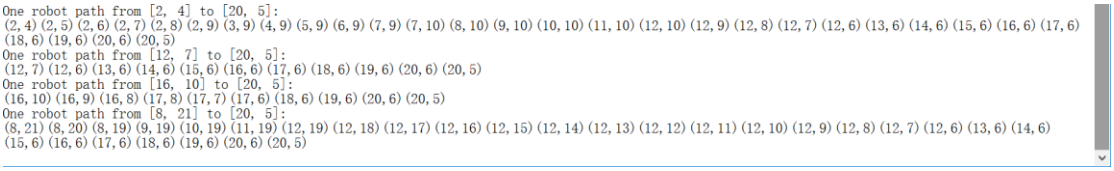






Plot

Console



File

3.4.2 Test file

```
25 30
4
2 4
8 21
12 7
16 10
20 5
10000000011000000001100111100 1
11000000111100000011110000001 1
00000000001100000011110011110 0
10001100010000000000110011110 0
10011110011001111001000000011 0
00011110000000110000100000001 0
11000000111100000011000000001 1
00001100001100000000100000000 1
11000000111001111001100000011 0
10000000011000000001000000000 0
11000000111000000011100000011 0
00000000001100000011011001111 0
10001100010000000000000000111 1
10011110011001111001000000111 1
00011110000000110000100000011 0
11000000111100000011100001100 0
00001100001100000000100000011 0
11000000111001111001111100000 0
00001100001100000000000011110 0
11000000111001111001010000000 0
10000000011000000000000000110 0
11000000111100000011110000001 1
00000000001100000011011000100 0
00000000110000000010000000001 1
11000000111100000011100000011 0
```

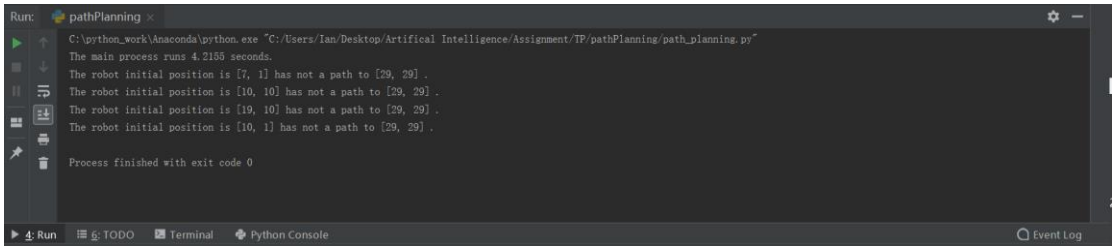
4. Five benchmark problems

4.1 path_planning_1.txt

4.1.1 Result

The main process runs 4.2155 seconds.

None of the four robots can move from the initial point to the rendezvous point.



Console

4.2 path_planning_2.txt

4.2.1 Result

The main process runs 20.5587 seconds.

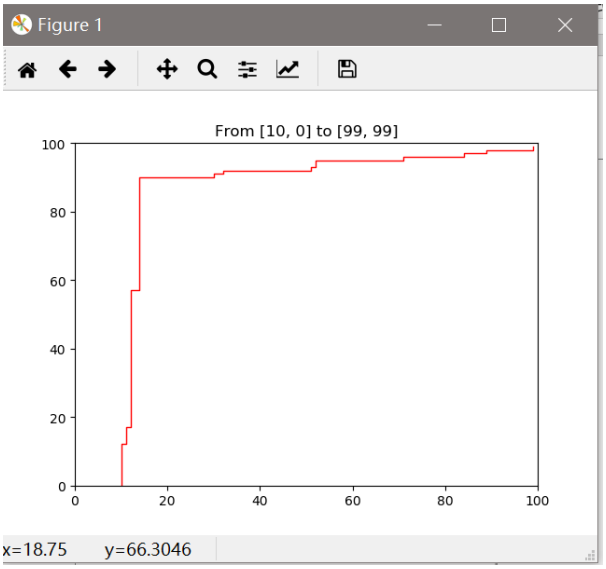


Figure 1
From [10, 0] to [99, 99]
100
80
60
40
20
0
0
20
40
60
80
100
x=18.75 y=66.3046

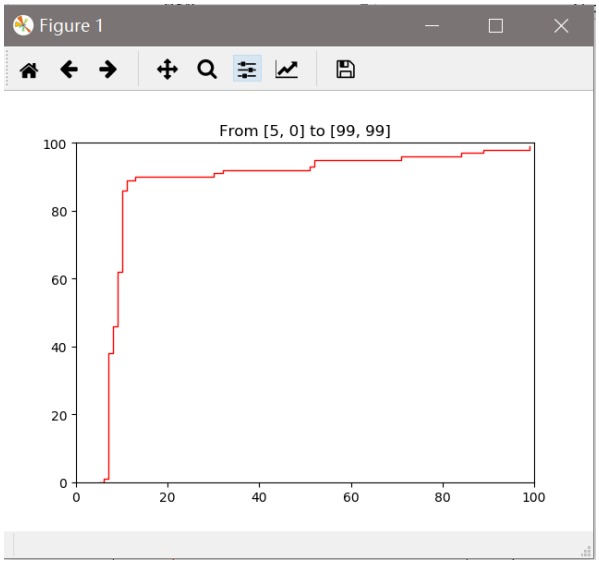


Figure 1
From [5, 0] to [99, 99]
100
80
60
40
20
0
0
20
40
60
80
100

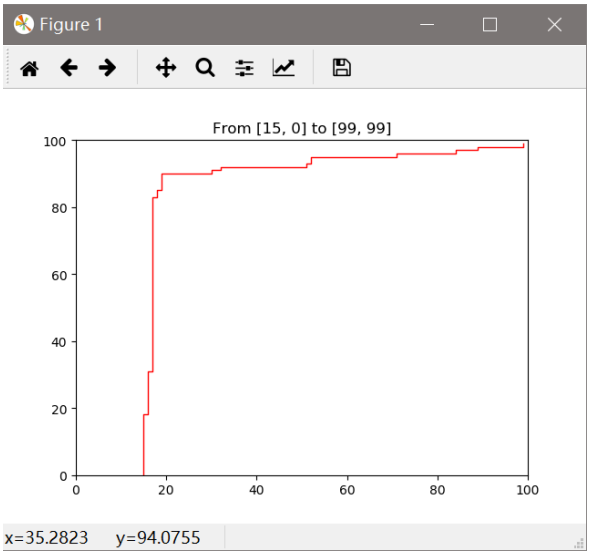
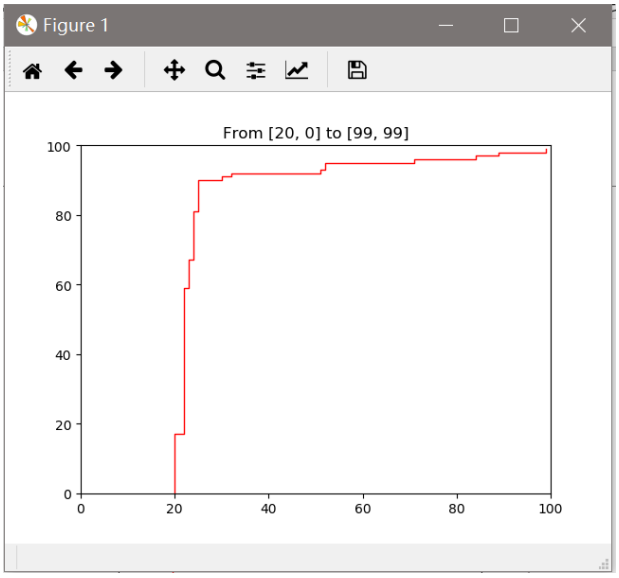
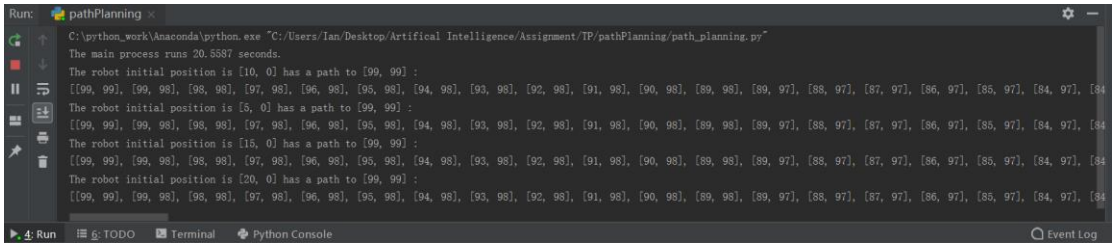


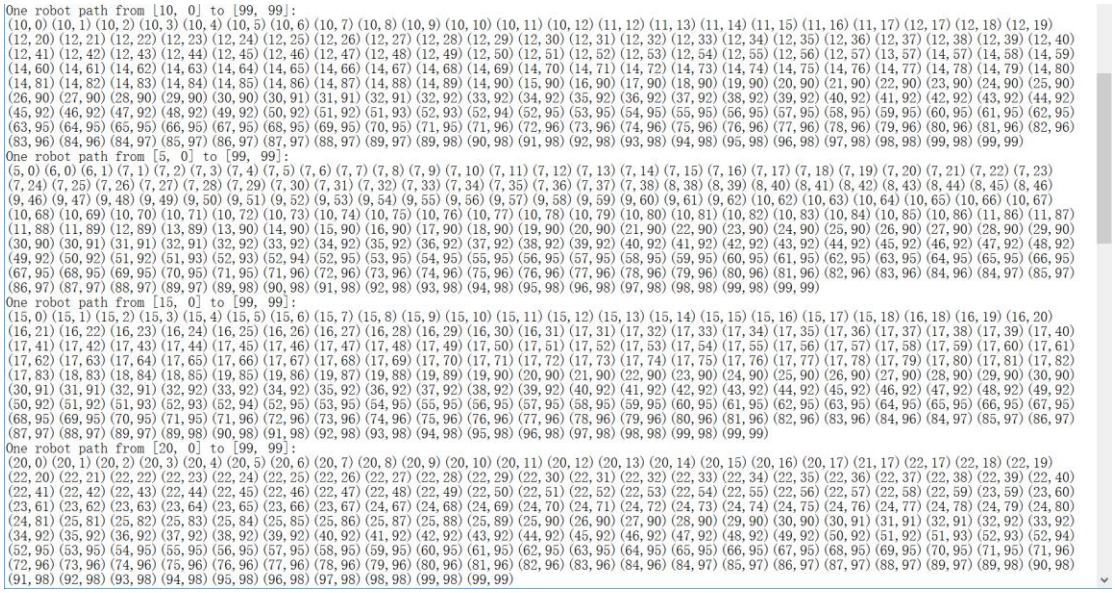
Figure 1
From [15, 0] to [99, 99]
100
80
60
40
20
0
0
20
40
60
80
100
x=35.2823 y=94.0755



Plot



Console



File

4.3 path_planning_3.txt

4.3.1 Result

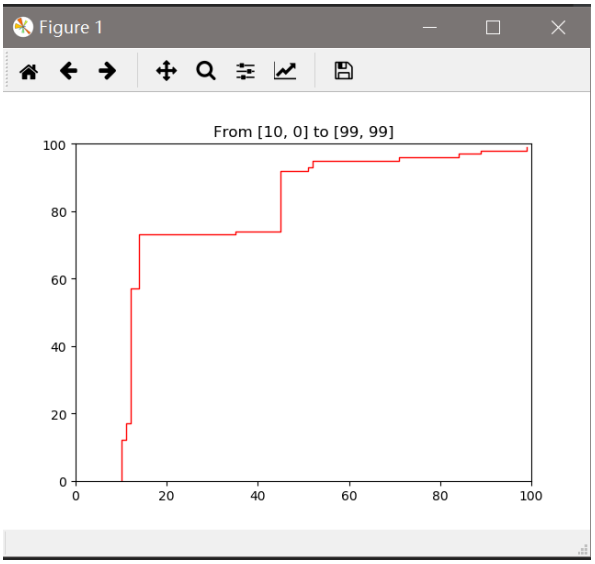
This test file is too large for our program. Our laptops are quad-core processors, so we cannot run four processes at the same time for a long time.

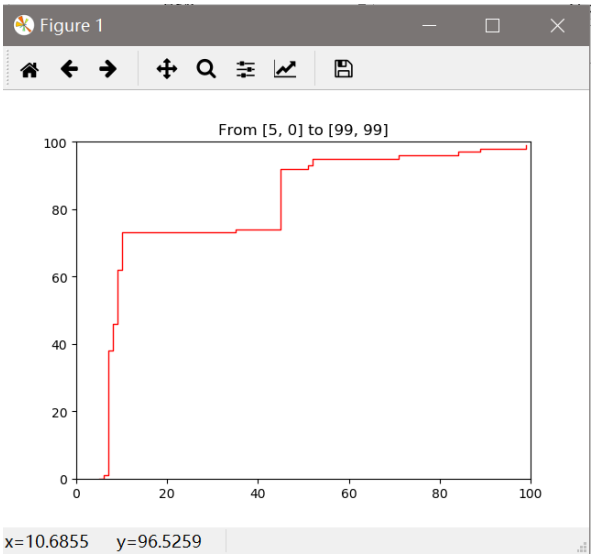
If we only use two processes, it will take too much time

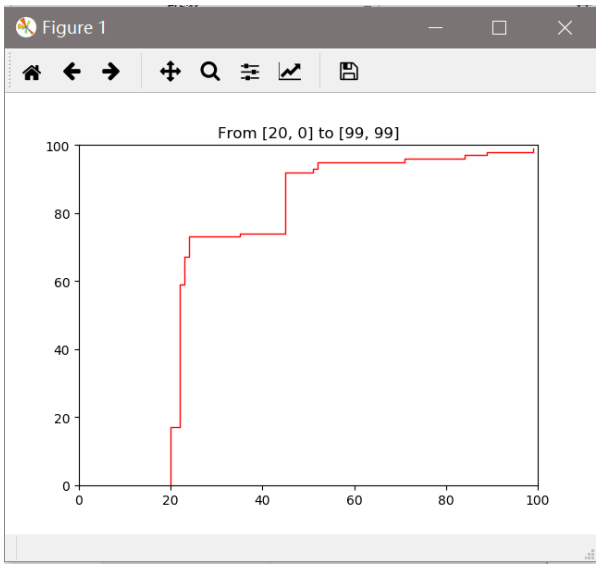
4.4 path_planning_4.txt

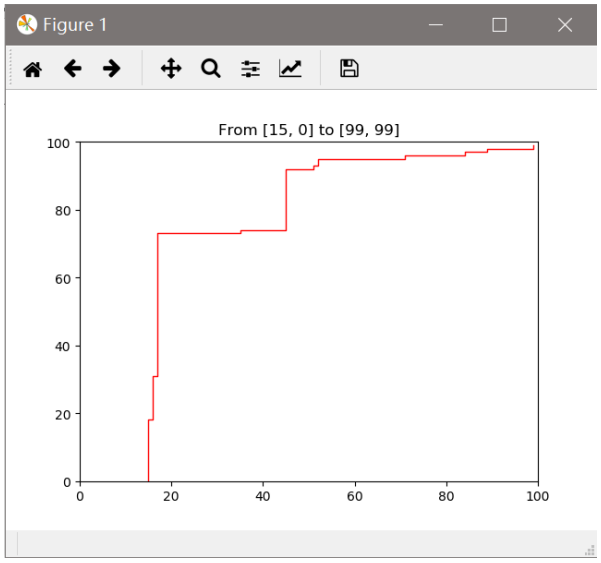
4.4.1 Result

The main process runs 19.3673 seconds.

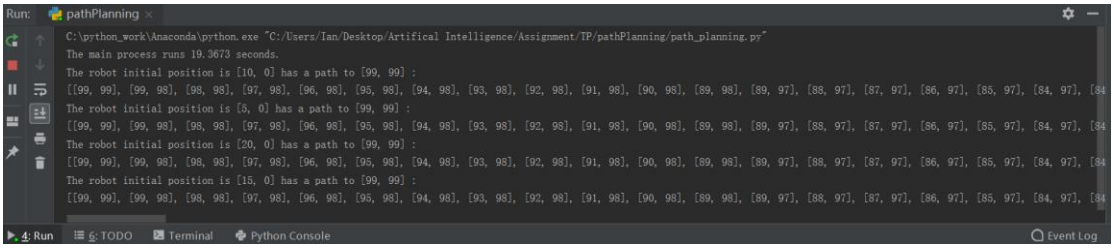








Plot



Console

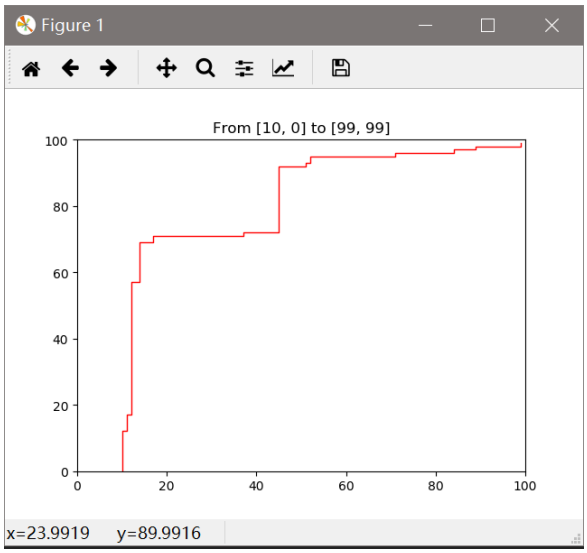
```
One robot path from [10, 0] to [99, 99]:
(10, 0) (10, 1) (10, 2) (10, 3) (10, 4) (10, 5) (10, 6) (10, 7) (10, 8) (10, 9) (10, 10) (10, 11) (10, 12) (11, 12) (11, 13) (11, 14) (11, 15) (11, 16) (11, 17) (12, 17) (12, 18) (12, 19)
(12, 20) (12, 21) (12, 22) (12, 23) (12, 24) (12, 25) (12, 26) (12, 27) (12, 28) (12, 29) (12, 30) (12, 31) (12, 32) (12, 33) (12, 34) (12, 35) (12, 36) (12, 37) (12, 38) (12, 39) (12, 40)
(12, 41) (12, 42) (12, 43) (12, 44) (12, 45) (12, 46) (12, 47) (12, 48) (12, 49) (12, 50) (12, 51) (12, 52) (12, 53) (12, 54) (12, 55) (12, 56) (12, 57) (13, 57) (14, 57) (14, 58) (14, 59)
(14, 60) (14, 61) (14, 62) (14, 63) (14, 64) (14, 65) (14, 66) (14, 67) (14, 68) (14, 69) (14, 70) (14, 71) (14, 72) (14, 73) (15, 73) (16, 73) (17, 73) (18, 73) (19, 73) (20, 73) (21, 73)
(22, 73) (23, 73) (24, 73) (25, 73) (26, 73) (27, 73) (28, 73) (29, 73) (30, 73) (31, 73) (32, 73) (33, 73) (34, 73) (35, 73) (35, 74) (36, 74) (37, 74) (38, 74) (39, 74) (40, 74) (41, 74)
(42, 74) (43, 74) (44, 74) (45, 74) (45, 75) (45, 76) (45, 77) (45, 78) (45, 79) (45, 80) (45, 81) (45, 82) (45, 83) (45, 84) (45, 85) (45, 86) (45, 87) (45, 88) (45, 89) (45, 90) (45, 91)
(45, 92) (46, 92) (47, 92) (48, 92) (49, 92) (50, 92) (51, 92) (51, 93) (52, 93) (52, 94) (52, 95) (53, 95) (54, 95) (55, 95) (56, 95) (57, 95) (58, 95) (59, 95) (60, 95) (61, 95) (62, 95)
(63, 95) (64, 95) (65, 95) (66, 95) (67, 95) (68, 95) (69, 95) (70, 95) (71, 95) (71, 96) (72, 96) (73, 96) (74, 96) (75, 96) (76, 96) (77, 96) (78, 96) (79, 96) (80, 96) (81, 96) (82, 96)
(83, 96) (84, 96) (84, 97) (85, 97) (86, 97) (87, 97) (88, 97) (89, 97) (89, 98) (90, 98) (91, 98) (92, 98) (93, 98) (94, 98) (95, 98) (96, 98) (97, 98) (98, 98) (99, 98) (99, 99)
One robot path from [5, 0] to [99, 99]:
(5, 0) (6, 0) (6, 1) (7, 1) (7, 2) (7, 3) (7, 4) (7, 5) (7, 6) (7, 7) (7, 8) (7, 9) (7, 10) (7, 11) (7, 12) (7, 13) (7, 14) (7, 15) (7, 16) (7, 17) (7, 18) (7, 19) (7, 20) (7, 21) (7, 22) (7, 23)
(7, 24) (7, 25) (7, 26) (7, 27) (7, 28) (7, 29) (7, 30) (7, 31) (7, 32) (7, 33) (7, 34) (7, 35) (7, 36) (7, 37) (7, 38) (8, 38) (8, 39) (8, 40) (8, 41) (8, 42) (8, 43) (8, 44) (8, 45) (8, 46)
(9, 46) (9, 47) (9, 48) (9, 49) (9, 50) (9, 51) (9, 52) (9, 53) (9, 54) (9, 55) (9, 56) (9, 57) (9, 58) (9, 59) (9, 60) (9, 61) (9, 62) (10, 62) (10, 63) (10, 64) (10, 65) (10, 66) (10, 67)
(10, 68) (10, 69) (10, 70) (10, 71) (10, 72) (10, 73) (11, 73) (12, 73) (13, 73) (14, 73) (15, 73) (16, 73) (17, 73) (18, 73) (19, 73) (20, 73) (21, 73) (22, 73) (23, 73) (24, 73) (25, 73)
(26, 73) (27, 73) (28, 73) (29, 73) (30, 73) (31, 73) (32, 73) (33, 73) (34, 73) (35, 73) (35, 74) (36, 74) (37, 74) (38, 74) (39, 74) (40, 74) (41, 74) (42, 74) (43, 74) (44, 74) (45, 74)
(45, 75) (45, 76) (45, 77) (45, 78) (45, 79) (45, 80) (45, 81) (45, 82) (45, 83) (45, 84) (45, 85) (45, 86) (45, 87) (45, 88) (45, 89) (45, 90) (45, 91) (45, 92) (46, 92) (47, 92) (48, 92)
(49, 92) (50, 92) (51, 92) (51, 93) (52, 93) (52, 94) (52, 95) (53, 95) (54, 95) (55, 95) (56, 95) (57, 95) (58, 95) (59, 95) (60, 95) (61, 95) (62, 95) (63, 95) (64, 95) (65, 95) (66, 95)
(67, 95) (68, 95) (69, 95) (70, 95) (71, 95) (71, 96) (72, 96) (73, 96) (74, 96) (75, 96) (76, 96) (77, 96) (78, 96) (79, 96) (80, 96) (81, 96) (82, 96) (83, 96) (84, 96) (84, 97) (85, 97)
(86, 97) (87, 97) (88, 97) (89, 97) (89, 98) (90, 98) (91, 98) (92, 98) (93, 98) (94, 98) (95, 98) (96, 98) (97, 98) (98, 98) (99, 98) (99, 99)
One robot path from [20, 0] to [99, 99]:
(20, 0) (20, 1) (20, 2) (20, 3) (20, 4) (20, 5) (20, 6) (20, 7) (20, 8) (20, 9) (20, 10) (20, 11) (20, 12) (20, 13) (20, 14) (20, 15) (20, 16) (20, 17) (21, 17) (22, 17) (22, 18) (22, 19)
(22, 20) (22, 21) (22, 22) (22, 23) (22, 24) (22, 25) (22, 26) (22, 27) (22, 28) (22, 29) (22, 30) (22, 31) (22, 32) (22, 33) (22, 34) (22, 35) (22, 36) (22, 37) (22, 38) (22, 39) (22, 40)
(22, 41) (22, 42) (22, 43) (22, 44) (22, 45) (22, 46) (22, 47) (22, 48) (22, 49) (22, 50) (22, 51) (22, 52) (22, 53) (22, 54) (22, 55) (22, 56) (22, 57) (22, 58) (22, 59) (23, 59) (23, 60)
(23, 61) (23, 62) (23, 63) (23, 64) (23, 65) (23, 66) (23, 67) (24, 67) (24, 68) (24, 69) (24, 70) (24, 71) (24, 72) (24, 73) (25, 73) (26, 73) (27, 73) (28, 73) (29, 73) (30, 73) (31, 73)
(32, 73) (33, 73) (34, 73) (35, 73) (35, 74) (36, 74) (37, 74) (38, 74) (39, 74) (40, 74) (41, 74) (42, 74) (43, 74) (44, 74) (45, 74) (45, 75) (45, 76) (45, 77) (45, 78) (45, 79) (45, 80)
(45, 81) (45, 82) (45, 83) (45, 84) (45, 85) (45, 86) (45, 87) (45, 88) (45, 89) (45, 90) (45, 91) (45, 92) (46, 92) (47, 92) (48, 92) (49, 92) (50, 92) (51, 92) (51, 93) (52, 93) (52, 94)
(52, 95) (53, 95) (54, 95) (55, 95) (56, 95) (57, 95) (58, 95) (59, 95) (60, 95) (61, 95) (62, 95) (63, 95) (64, 95) (65, 95) (66, 95) (67, 95) (68, 95) (69, 95) (70, 95) (71, 95) (71, 96)
(72, 96) (73, 96) (74, 96) (75, 96) (76, 96) (77, 96) (78, 96) (79, 96) (80, 96) (81, 96) (82, 96) (83, 96) (84, 96) (84, 97) (85, 97) (86, 97) (87, 97) (88, 97) (89, 97) (89, 98) (90, 98)
(91, 98) (92, 98) (93, 98) (94, 98) (95, 98) (96, 98) (97, 98) (98, 98) (99, 98) (99, 99)
One robot path from [15, 0] to [99, 99]:
(15, 0) (15, 1) (15, 2) (15, 3) (15, 4) (15, 5) (15, 6) (15, 7) (15, 8) (15, 9) (15, 10) (15, 11) (15, 12) (15, 13) (15, 14) (15, 15) (15, 16) (15, 17) (15, 18) (16, 18) (16, 19) (16, 20)
(16, 21) (16, 22) (16, 23) (16, 24) (16, 25) (16, 26) (16, 27) (16, 28) (16, 29) (16, 30) (16, 31) (17, 31) (17, 32) (17, 33) (17, 34) (17, 35) (17, 36) (17, 37) (17, 38) (17, 39) (17, 40)
(17, 41) (17, 42) (17, 43) (17, 44) (17, 45) (17, 46) (17, 47) (17, 48) (17, 49) (17, 50) (17, 51) (17, 52) (17, 53) (17, 54) (17, 55) (17, 56) (17, 57) (17, 58) (17, 59) (17, 60) (17, 61)
(17, 62) (17, 63) (17, 64) (17, 65) (17, 66) (17, 67) (17, 68) (17, 69) (17, 70) (17, 71) (17, 72) (17, 73) (18, 73) (19, 73) (20, 73) (21, 73) (22, 73) (23, 73) (24, 73) (25, 73) (26, 73)
(27, 73) (28, 73) (29, 73) (30, 73) (31, 73) (32, 73) (33, 73) (34, 73) (35, 73) (35, 74) (36, 74) (37, 74) (38, 74) (39, 74) (40, 74) (41, 74) (42, 74) (43, 74) (44, 74) (45, 74) (45, 75)
(45, 76) (45, 77) (45, 78) (45, 79) (45, 80) (45, 81) (45, 82) (45, 83) (45, 84) (45, 85) (45, 86) (45, 87) (45, 88) (45, 89) (45, 90) (45, 91) (45, 92) (46, 92) (47, 92) (48, 92) (49, 92)
(50, 92) (51, 92) (51, 93) (52, 93) (52, 94) (52, 95) (53, 95) (54, 95) (55, 95) (56, 95) (57, 95) (58, 95) (59, 95) (60, 95) (61, 95) (62, 95) (63, 95) (64, 95) (65, 95) (66, 95) (67, 95)
(68, 95) (69, 95) (70, 95) (71, 95) (71, 96) (72, 96) (73, 96) (74, 96) (75, 96) (76, 96) (77, 96) (78, 96) (79, 96) (80, 96) (81, 96) (82, 96) (83, 96) (84, 96) (84, 97) (85, 97) (86, 97)
(87, 97) (88, 97) (89, 97) (89, 98) (90, 98) (91, 98) (92, 98) (93, 98) (94, 98) (95, 98) (96, 98) (97, 98) (98, 98) (99, 98) (99, 99)
```


File

4.5 path_planning_5.txt

4.5.1 Result

The main process runs 17.1256 seconds.



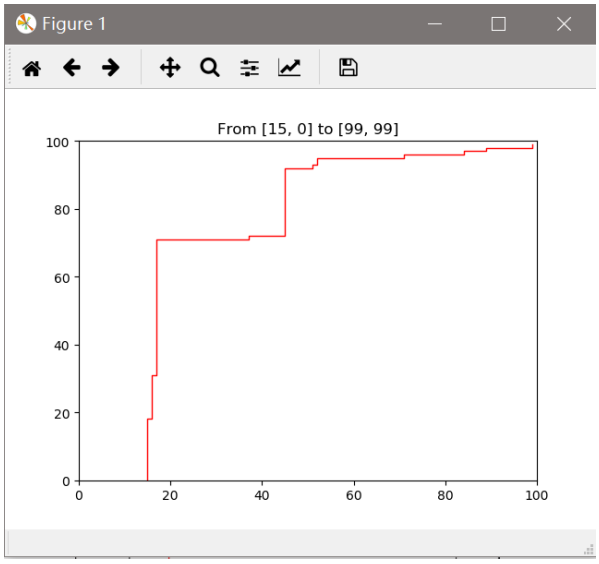


Figure 1
From [15, 0] to [99, 99]
100
80
60
40
20
0
0
20
40
60
80
100

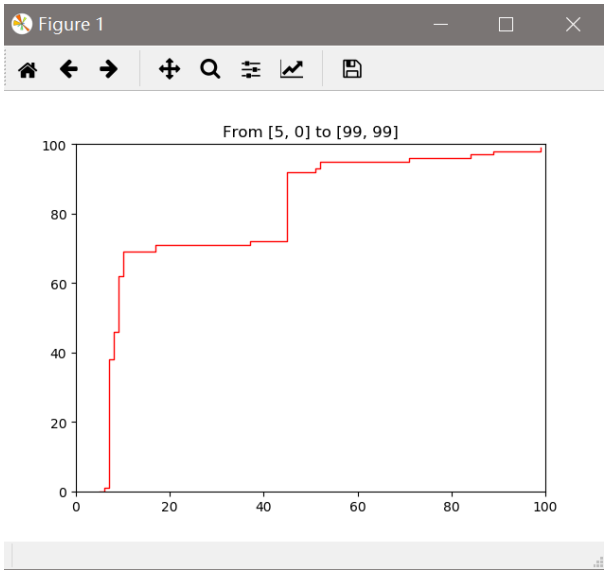


Figure 1
From [5, 0] to [99, 99]
100
80
60
40
20
0
0
20
40
60
80
100

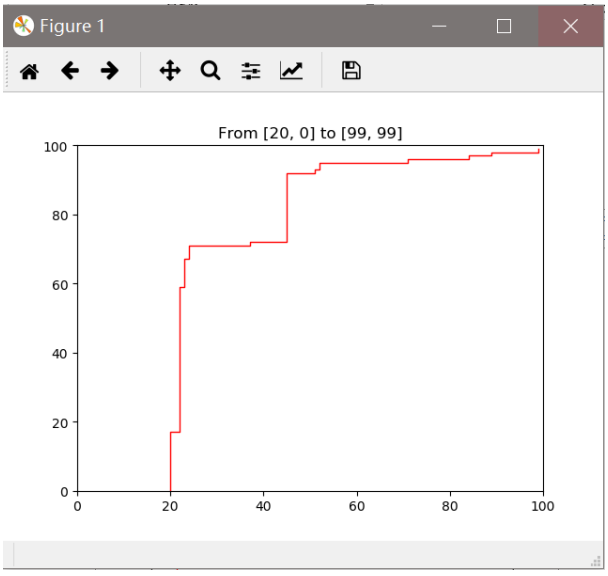
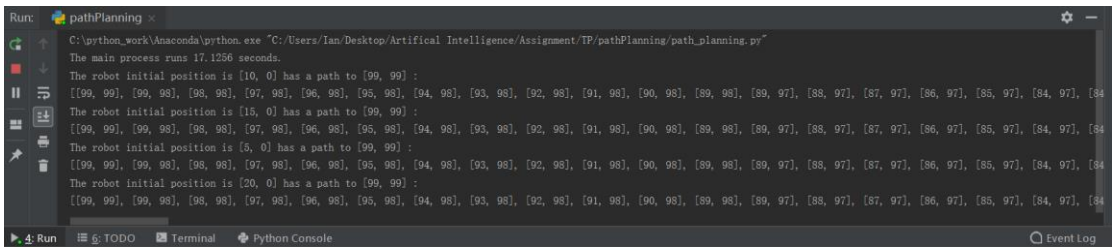


Figure 1
From [20, 0] to [99, 99]
100
80
60
40
20
0
0
20
40
60
80
100

Plot



Console

One robot path from [10, 0] to [99, 99]:
(10, 0) (10, 1) (10, 2) (10, 3) (10, 4) (10, 5) (10, 6) (10, 7) (10, 8) (10, 9) (10, 10) (10, 11) (10, 12) (11, 12) (11, 13) (11, 14) (11, 15) (11, 16) (11, 17) (12, 17) (12, 18) (12, 19)
(12, 20) (12, 21) (12, 22) (12, 23) (12, 24) (12, 25) (12, 26) (12, 27) (12, 28) (12, 29) (12, 30) (12, 31) (12, 32) (12, 33) (12, 34) (12, 35) (12, 36) (12, 37) (12, 38) (12, 39) (12, 40)
(12, 41) (12, 42) (12, 43) (12, 44) (12, 45) (12, 46) (12, 47) (12, 48) (12, 49) (12, 50) (12, 51) (12, 52) (12, 53) (12, 54) (12, 55) (12, 56) (12, 57) (13, 57) (14, 57) (14, 58) (14, 59)
(14, 60) (14, 61) (14, 62) (14, 63) (14, 64) (14, 65) (14, 66) (14, 67) (14, 68) (14, 69) (15, 69) (16, 69) (17, 69) (17, 70) (17, 71) (18, 71) (19, 71) (20, 71) (21, 71) (22, 71) (23, 71)
(24, 71) (25, 71) (26, 71) (27, 71) (28, 71) (29, 71) (30, 71) (31, 71) (32, 71) (33, 71) (34, 71) (35, 71) (36, 71) (37, 71) (37, 72) (38, 72) (39, 72) (40, 72) (41, 72) (42, 72) (43, 72)
(44, 72) (45, 72) (45, 73) (45, 74) (45, 75) (45, 76) (45, 77) (45, 78) (45, 79) (45, 80) (45, 81) (45, 82) (45, 83) (45, 84) (45, 85) (45, 86) (45, 87) (45, 88) (45, 89) (45, 90) (45, 91)
(45, 92) (46, 92) (47, 92) (48, 92) (49, 92) (50, 92) (51, 92) (51, 93) (52, 93) (52, 94) (52, 95) (53, 95) (54, 95) (55, 95) (56, 95) (57, 95) (58, 95) (59, 95) (60, 95) (61, 95) (62, 95)
(63, 95) (64, 95) (65, 95) (66, 95) (67, 95) (68, 95) (69, 95) (70, 95) (71, 95) (71, 96) (72, 96) (73, 96) (74, 96) (75, 96) (76, 96) (77, 96) (78, 96) (79, 96) (80, 96) (81, 96) (82, 96)
(83, 96) (84, 96) (84, 97) (85, 97) (86, 97) (87, 97) (88, 97) (89, 97) (89, 98) (90, 98) (91, 98) (92, 98) (93, 98) (94, 98) (95, 98) (96, 98) (97, 98) (98, 98) (99, 98) (99, 99)
One robot path from [15, 0] to [99, 99]:
(15, 0) (15, 1) (15, 2) (15, 3) (15, 4) (15, 5) (15, 6) (15, 7) (15, 8) (15, 9) (15, 10) (15, 11) (15, 12) (15, 13) (15, 14) (15, 15) (15, 16) (15, 17) (15, 18) (16, 18) (16, 19) (16, 20)
(16, 21) (16, 22) (16, 23) (16, 24) (16, 25) (16, 26) (16, 27) (16, 28) (16, 29) (16, 30) (16, 31) (17, 31) (17, 32) (17, 33) (17, 34) (17, 35) (17, 36) (17, 37) (17, 38) (17, 39) (17, 40)
(17, 41) (17, 42) (17, 43) (17, 44) (17, 45) (17, 46) (17, 47) (17, 48) (17, 49) (17, 50) (17, 51) (17, 52) (17, 53) (17, 54) (17, 55) (17, 56) (17, 57) (17, 58) (17, 59) (17, 60) (17, 61)
(17, 62) (17, 63) (17, 64) (17, 65) (17, 66) (17, 67) (17, 68) (17, 69) (17, 70) (17, 71) (18, 71) (19, 71) (20, 71) (21, 71) (22, 71) (23, 71) (24, 71) (25, 71) (26, 71) (27, 71) (28, 71)
(29, 71) (30, 71) (31, 71) (32, 71) (33, 71) (34, 71) (35, 71) (36, 71) (37, 71) (37, 72) (38, 72) (39, 72) (40, 72) (41, 72) (42, 72) (43, 72) (44, 72) (45, 72) (45, 73) (45, 74) (45, 75)
(45, 76) (45, 77) (45, 78) (45, 79) (45, 80) (45, 81) (45, 82) (45, 83) (45, 84) (45, 85) (45, 86) (45, 87) (45, 88) (45, 89) (45, 90) (45, 91) (45, 92) (46, 92) (47, 92) (48, 92) (49, 92)
(50, 92) (51, 92) (51, 93) (52, 93) (52, 94) (52, 95) (53, 95) (54, 95) (55, 95) (56, 95) (57, 95) (58, 95) (59, 95) (60, 95) (61, 95) (62, 95) (63, 95) (64, 95) (65, 95) (66, 95) (67, 95)
(68, 95) (69, 95) (70, 95) (71, 95) (71, 96) (72, 96) (73, 96) (74, 96) (75, 96) (76, 96) (77, 96) (78, 96) (79, 96) (80, 96) (81, 96) (82, 96) (83, 96) (84, 96) (84, 97) (85, 97) (86, 97)
(87, 97) (88, 97) (89, 97) (89, 98) (90, 98) (91, 98) (92, 98) (93, 98) (94, 98) (95, 98) (96, 98) (97, 98) (98, 98) (99, 98) (99, 99)
One robot path from [5, 0] to [99, 99]:
(5, 0) (6, 0) (6, 1) (7, 1) (7, 2) (7, 3) (7, 4) (7, 5) (7, 6) (7, 7) (7, 8) (7, 9) (7, 10) (7, 11) (7, 12) (7, 13) (7, 14) (7, 15) (7, 16) (7, 17) (7, 18) (7, 19) (7, 20) (7, 21) (7, 22) (7, 23)
(7, 24) (7, 25) (7, 26) (7, 27) (7, 28) (7, 29) (7, 30) (7, 31) (7, 32) (7, 33) (7, 34) (7, 35) (7, 36) (7, 37) (7, 38) (8, 38) (8, 39) (8, 40) (8, 41) (8, 42) (8, 43) (8, 44) (8, 45) (8, 46)
(9, 46) (9, 47) (9, 48) (9, 49) (9, 50) (9, 51) (9, 52) (9, 53) (9, 54) (9, 55) (9, 56) (9, 57) (9, 58) (9, 59) (9, 60) (9, 61) (9, 62) (10, 62) (10, 63) (10, 64) (10, 65) (10, 66) (10, 67)
(10, 68) (10, 69) (11, 69) (12, 69) (13, 69) (14, 69) (15, 69) (16, 69) (17, 69) (17, 70) (17, 71) (18, 71) (19, 71) (20, 71) (21, 71) (22, 71) (23, 71) (24, 71) (25, 71) (26, 71) (27, 71)
(28, 71) (29, 71) (30, 71) (31, 71) (32, 71) (33, 71) (34, 71) (35, 71) (36, 71) (37, 71) (37, 72) (38, 72) (39, 72) (40, 72) (41, 72) (42, 72) (43, 72) (44, 72) (45, 72) (45, 73) (45, 74)
(45, 75) (45, 76) (45, 77) (45, 78) (45, 79) (45, 80) (45, 81) (45, 82) (45, 83) (45, 84) (45, 85) (45, 86) (45, 87) (45, 88) (45, 89) (45, 90) (45, 91) (45, 92) (46, 92) (47, 92) (48, 92)
(49, 92) (50, 92) (51, 92) (51, 93) (52, 93) (52, 94) (52, 95) (53, 95) (54, 95) (55, 95) (55, 95) (56, 95) (57, 95) (58, 95) (59, 95) (60, 95) (61, 95) (62, 95) (63, 95) (64, 95) (65, 95) (66, 95)
(67, 95) (68, 95) (69, 95) (70, 95) (71, 95) (71, 96) (72, 96) (73, 96) (74, 96) (75, 96) (76, 96) (77, 96) (78, 96) (79, 96) (80, 96) (81, 96) (82, 96) (83, 96) (84, 96) (84, 97) (85, 97)
(86, 97) (87, 97) (88, 97) (89, 97) (89, 98) (90, 98) (91, 98) (92, 98) (93, 98) (94, 98) (95, 98) (96, 98) (97, 98) (98, 98) (99, 98) (99, 99)
One robot path from [20, 0] to [99, 99]:
(20, 0) (20, 1) (20, 2) (20, 3) (20, 4) (20, 5) (20, 6) (20, 7) (20, 8) (20, 9) (20, 10) (20, 11) (20, 12) (20, 13) (20, 14) (20, 15) (20, 16) (20, 17) (21, 17) (22, 17) (22, 18) (22, 19)
(22, 20) (22, 21) (22, 22) (22, 23) (22, 24) (22, 25) (22, 26) (22, 27) (22, 28) (22, 29) (22, 30) (22, 31) (22, 32) (22, 33) (22, 34) (22, 35) (22, 36) (22, 37) (22, 38) (22, 39) (22, 40)
(22, 41) (22, 42) (22, 43) (22, 44) (22, 45) (22, 46) (22, 47) (22, 48) (22, 49) (22, 50) (22, 51) (22, 52) (22, 53) (22, 54) (22, 55) (22, 56) (22, 57) (22, 58) (22, 59) (23, 59) (23, 60)
(23, 61) (23, 62) (23, 63) (23, 64) (23, 65) (23, 66) (23, 67) (24, 67) (24, 68) (24, 69) (24, 70) (24, 71) (25, 71) (26, 71) (27, 71) (28, 71) (29, 71) (30, 71) (31, 71) (32, 71) (33, 71)
(34, 71) (35, 71) (36, 71) (37, 71) (37, 72) (38, 72) (39, 72) (40, 72) (41, 72) (42, 72) (43, 72) (44, 72) (45, 72) (45, 73) (45, 74) (45, 75) (45, 76) (45, 77) (45, 78) (45, 79) (45, 80)
(45, 81) (45, 82) (45, 83) (45, 84) (45, 85) (45, 86) (45, 87) (45, 88) (45, 89) (45, 90) (45, 91) (45, 92) (46, 92) (47, 92) (48, 92) (49, 92) (50, 92) (51, 92) (51, 93) (52, 93) (52, 94)
(52, 95) (53, 95) (54, 95) (55, 95) (56, 95) (57, 95) (58, 95) (59, 95) (60, 95) (61, 95) (62, 95) (63, 95) (64, 95) (65, 95) (66, 95) (67, 95) (68, 95) (69, 95) (70, 95) (71, 95) (71, 96)
(72, 96) (73, 96) (74, 96) (75, 96) (76, 96) (77, 96) (78, 96) (79, 96) (80, 96) (81, 96) (82, 96) (83, 96) (84, 96) (84, 97) (85, 97) (86, 97) (87, 97) (88, 97) (89, 97) (89, 98) (90, 98)
(91, 98) (92, 98) (93, 98) (94, 98) (95, 98) (96, 98) (97, 98) (98, 98) (99, 98) (99, 99)

File