Edward Nataniel C. Apostol
2010-27627

Solution by Searching

The goal of this machine exercise is to compare the performance of greedy and a-star searching algorithms. In this machine exercise, we will solve mazes by searching for the path with the minimal cost.

I. Input Format

```
5 5
1 1 1 1 1
0 1 1 1 1
1 1 0 0 1
1 1 0 0 0
1 1 1 1 0
Source: 2 2
Destination: 4 4
Up: 4
Down 4
Left: 5
Right: 5
Diagonal: 6
```

Above is a sample input. The first line of the input file is the dimensions of the maze. The first integer is the number of rows (m) and the second integer is the number of columns (n). The next m lines contains the topology of the map. All cells marked with '0' are passable while cells marked with '1' are the boundaries of the maze.

The input file also contains the source and destination of the map of the path that we would like to find. Note that the numbering of the rows and columns start with 0, so the first row is referred as row 0.

The costs of moving to a particular direction are also specified. In the example above, the cost of moving diagonally (cost of 6) is more expensive than moving downward (cost of 4).

II. Building the adjacency matrix

The program will convert the map into an adjacency dictionary. The key for the dictionary is the coordinate. Its value is a list of coordinates that are adjacent to the key.

```
{ 1 0 : [],
  2 2 : [2 3, 3 2, 3 3],
  2 3 : [2 2, 3 2, 3 3, 3 4],
  3 2 : [2 2, 2 3, 3 3],
  3 3 : [2 2, 2 3, 3 2, 3 4, 4 4],
  3 4 : [2 4, 3 3, 4 4],
  4 4 : [3 3, 3 4]}
```

III. Computation of the heuristic values

For this machine exercise, euclidean distance from each cell to the destination cell will be used as the heuristic values. A dictionary was also created to store heuristic values.

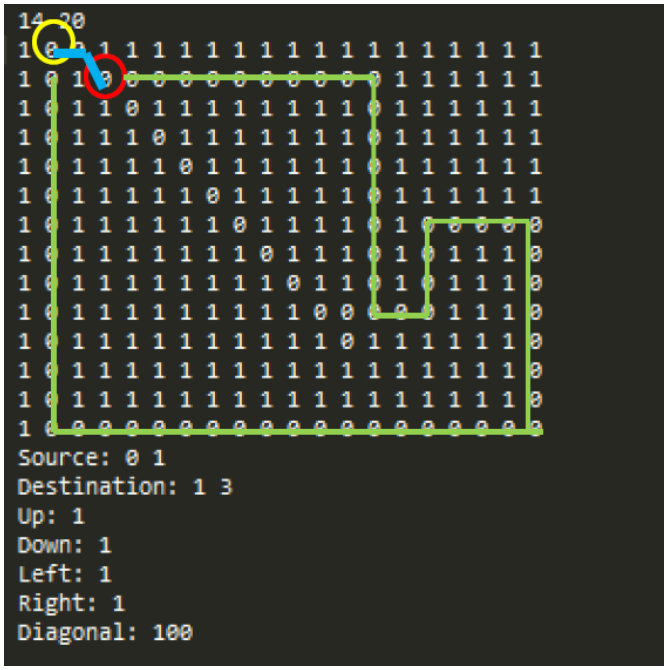
```
{ 1 0 : 5
2 2 : 2.828
2 3 : 2.236
3 2 : 2.236
3 3 : 1.414
3 4 : 1
4 4 : 0 }
```

IV. Searching algorithm

a. A class named “Node” will be declared. Each node has a coordinate, path_cost, f(n) value, and parent node. The f(n) value is the evaluation function. It is equal to h(n) in greedy best-first search where h(n) is the heuristic function. It is equal to path cost + h(n) in A* search.
b. Two lists are initialized to be empty. The fringe (open list / nodes that are not yet expanded) and the expanded (closed list).
c. The source node is first inserted into the open list. Then, the f(n) value of the nodes in the open list will be compared. The node with the lowest f(n) value will be removed from the open list and transferred to the closed list.
d. All unexpanded nodes adjacent to the node with the lowest f(n) value will be inserted into the open list. For greedy best-first search, f(n) value is equal to the heuristic value. In A* search, f(n) value, is equal to heuristic value + new path cost (where new path cost = old path cost + step cost)
e. The step cost will be computed. The path cost per node in the open list will be updated.
f. The program will compare again the f(n) values of the nodes in the open list. The node with the minimum f(n) value will be removed from the open list and will be transferred to the closed list. All unexpanded nodes adjacent to the transferred node will be added to the open list. Their path cost and f(n) values will be updated.
g. The algorithm will terminate if the destination coordinate is selected as the minimum on the open list. If the open list becomes empty, yet the destination coordinate is not yet found, the algorithm terminates and will output “no path found”.
h. The program will “backtrack” from destination to source to determine the path. Since the node’s parent is stored in every node, the source can easily be reached using a loop.

V. Result

There are 10 different test cases used. For all of test cases, the paths found by using a-star search have always either lower or the same cost as greedy best-first search. Below is the solution for one of the test cases.



In this example, the source is encircled yellow while the destination is encircled red.

The greedy search outputs the blue line path with a cost of 101.0. Though it only took 2 steps to get from the destination from the source, it's total cost is high since a diagonal movement has a high cost.

The A* search outputs the green line path. The path is longer but the costs is much smaller than the one found by greedy search.

Below is the summary of the costs of the path found for all the 10 test cases.

test case number	**greedy**	**a-star**	**difference**
1	101	65	+36
2	40	24	+16
3	385	254	+131
4	400	20	+380
5	3.8284	3.8284	0
6	12.3137	12.3137	0
7	12.8995	12.8995	0
8	25.5999	23.9999	+1.6
9	9	6	+3
10	41	23	+18