Artificial Intelligence LAB-5

A* Algorithm and Best First Search

Date:8-2-22

A* Algorithm

Solution:

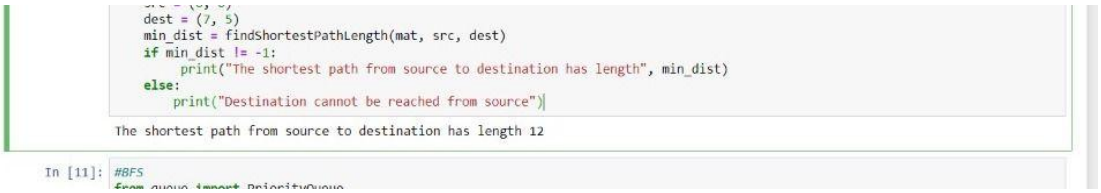
-Source Code:

```
import sys
def isSafe(mat, visited, x, y):
 return 0 <= x < len(mat) and 0 <= y < len(mat[0]) and \
 not (mat[x][y] == 0 or visited[x][y])
def findShortestPath(mat, visited, i, j, dest, min_dist=sys.maxsize, dist=0):

 if (i, j) == dest:
 return min(dist, min_dist)



 visited[i][j] = 1
if isSafe(mat, visited, i + 1, j):
 min_dist = findShortestPath(mat, visited, i + 1, j, dest, min_dist, dist + 1)

 if isSafe(mat, visited, i, j + 1):
 min_dist = findShortestPath(mat, visited, i, j + 1, dest, min_dist, dist + 1)



 if isSafe(mat, visited, i - 1, j):
 min_dist = findShortestPath(mat, visited, i - 1, j, dest, min_dist, dist + 1)

 if isSafe(mat, visited, i, j - 1):
 min_dist = findShortestPath(mat, visited, i, j - 1, dest, min_dist, dist + 1)

 visited[i][j] = 0

 return min_dist
```

```
def findShortestPathLength(mat, src, dest):

i, j = src

x, y = dest

if not mat or len(mat) == 0 or mat[i][j] == 0 or mat[x][y] == 0:
return -1

(M, N) = (len(mat), len(mat[0]))

visited = [[False for _ in range(N)] for _ in range(M)]
min_dist = findShortestPath(mat, visited, i, j, dest)
if min_dist != sys.maxsize:
return min_dist
else:
return -1
if __name__ == '__main__':
mat = [
[1, 1, 1, 1, 1, 0, 0, 1, 1, 1],
[0, 1, 1, 1, 1, 1, 0, 1, 0, 1],
[0, 0, 1, 0, 1, 1, 1, 0, 0, 1],
[1, 0, 1, 1, 1, 0, 1, 1, 0, 1],
[0, 0, 0, 1, 0, 0, 0, 1, 0, 1],
[1, 0, 1, 1, 1, 0, 0, 1, 1, 0],
[0, 0, 0, 0, 1, 0, 0, 1, 0, 1],
[0, 1, 1, 1, 1, 1, 1, 1, 0, 0],
[1, 1, 1, 1, 1, 0, 0, 1, 1, 1],
[0, 0, 1, 0, 0, 1, 1, 0, 0, 1]
]

src = (0, 0)
dest = (7, 5)
min_dist = findShortestPathLength(mat, src, dest)
if min_dist != -1:
```

print("The shortest path from source to destination has length", min_dist)
else:
print("Destination cannot be reached from source")

Output



Best First Search

Solution:

-Source Code:

```
from queue import PriorityQueue
v = 5
graph = [[] for i in range(v)]
def best_first_search(source, target, n):
 visited = [0] * n
 visited[0] = True
 pq = PriorityQueue()
 pq.put((0, source))
 while pq.empty() == False:
 u = pq.get()[1]
 print(u, end=" ")
 if u == target:
 break
 for v, c in graph[u]:
 if visited[v] == False:
 visited[v] = True
 pq.put((c, v))
 print()
def addedge(x, y, cost):
 graph[x].append((y, cost))
 graph[y].append((x, cost))
addedge(0, 1, 5)
addedge(0, 2, 1)
addedge(2, 3, 2)
addedge(1, 4, 1)
addedge(3, 4, 2)
source = 0
target = 4
best_first_search(source, target, v)
```

Output

```
addedge(3, 4, 2)
source = 0
target = 4
best_first_search(source, target, v)
```

```
0 2 3 4
```