PDM - Assignment1

Graph Search

Xiaotong Li 5965373

November 2023

Contents

1 Graph Search

Consider a directed graph $G = (V, E)$, with distances $d(e)$ for each $e \in E$, and consider two nodes $s, t \in V$. For each node v we define the function $P_s(v)$, which gives the length of the shortest path from s to v. Similar we define the function $P_t(v)$, which gives the length of the shortest path from v to t.

1.1 Question1.1

Show that for every edge $e = (u, v)$, the length of the shortest path from node s to node t that uses the edge e is $P_s(u) + d(e) + P_t(v)$.

Answer: Since G is a directed graph, so if you want to get to node t via edge $e = (u, v)$ from node s, you must first get to node u from node s, then get to node t from node v. Since $d(e)$ is fixed, it is obvious that the shortest path from s to t is $P_s(u) + d(e) + P_t(v)$.

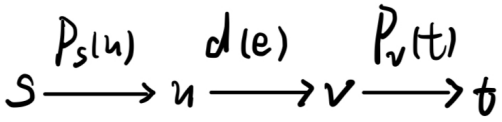

Figure 1: Shortest path from s to t

1.2 Question1.2

Let Q be the shortest path between the nodes s and t. Use the property obtained in Question 1.1, to propose an algorithm that finds the second shortest path from s to t (i.e., considering all paths that are not exactly equal to Q)

Answer: My algorithm is as follows: we choose an edge which is not in the path Q, and then we use the algorithm in Question 1.1 to find the shortest path from s to t that uses the edge we choose. We do this for all edges which are not in the path Q, and then we can get all the shortest paths from s to t that are not exactly equal to Q. Finally, we choose the shortest path from all the paths we get, and this is the second shortest path from s to t. The pseudo-code is as follows:

```
def second_shortest_path(s, t, Q):
    shortest_path = []
    for edge in E:
        if edge not in Q:
            shortest_path.append(P_s(edge.start) + d(edge) + P_t(edge.end))
    return min(shortest_path)

```

Example code of the algorithm in Question 1.2

2 Map to Graph

In most cases a graph to operate on is not provided directly. It needs to be extracted from an environment or map first. Use the given map, shown below, to find the **shortest-path roadmap** that can be used to find the shortest path from the indicated start to the goal location. Obstacles are shown in blue. Also connect the start and goal to the roadmap. Solve this problem graphically. Use the given obstacle sizes and distances to determine the costs of seven edges of your choice.

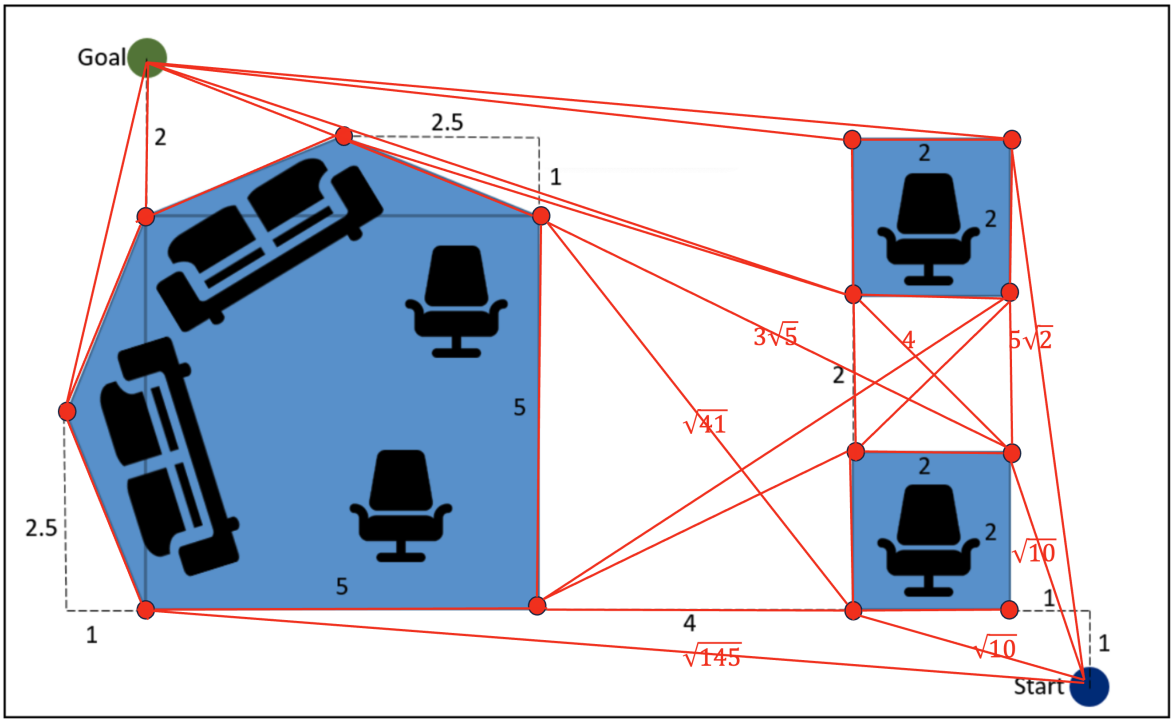

Figure 2: Graph of roadmap

3 Dijkstra and A*

Given an environment of operating drones a graph was extracted, similar to the previous exercise. The directed graph avoids collisions with all drones. The cost to traverse an edge is proportional to the distance of nodes and is indicated at each respective edge.

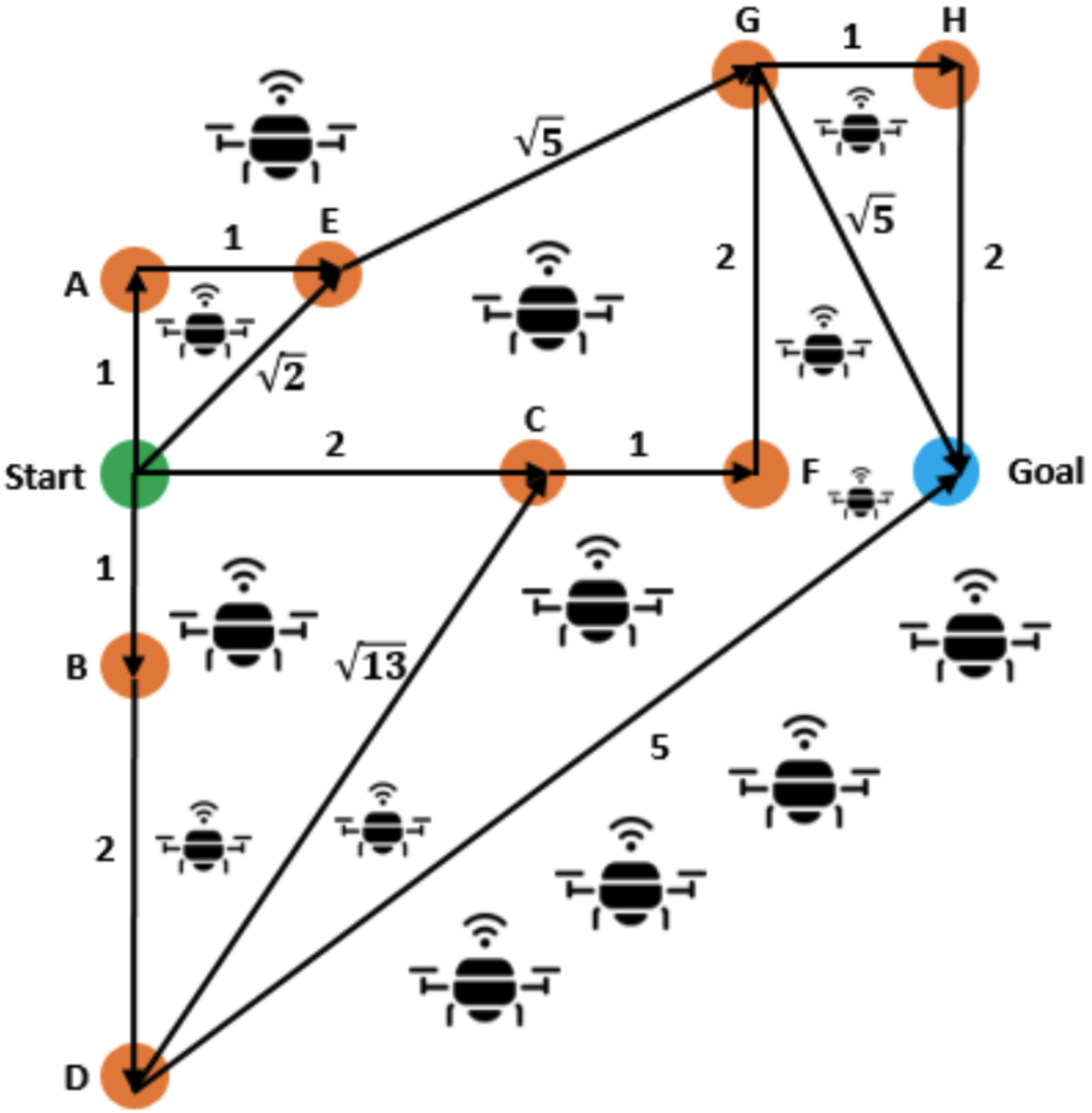

Figure 3: Graph of environment of operating drones

3.1 Question3.1

Find a path form the "start"-node to the "goal"-node using the Dijkstra algorithm. Explain each step.

Expanded	Q	Explaination
Start	A(Start,1), E(Start,$\sqrt{2}$), C(Start,2), B(Start,1)	Explore ABCE from Start
Start→A	E(Start,$\sqrt{2}$), C(Start,2), B(Start,1)	Add A(shortest) to the queue, no new node can be explored from A
Start→A→B	E(Start,$\sqrt{2}$), C(Start,2), D(B,3)	Add B(shortest) to the queue and explore D from B
Start→A→B →E	C(Start,2), D(B,3), G(E,$\sqrt{2}+\sqrt{5}$)	Add E(shortest) to the queue and explore G from E
Start→A→B →E→C	F(C,3), D(B,3), G(E,$\sqrt{2}+\sqrt{5}$)	Add C(shortest) to the queue and explore F from C
Start→A→B →E→C→F	D(B,3), G(E,$\sqrt{2}+\sqrt{5}$)	Add F(shortest) to the queue and no exploration from F
Start→A→B →E→C→F→D	G(E,$\sqrt{2}+\sqrt{5}$), Goal(D,8)	Add D(shortest) to the queue and explore Goal from D
Start→A→B →E→C→F→D →G	Goal(G,$\sqrt{2}+2\sqrt{5}$), H(G,$1+\sqrt{2}+\sqrt{5}$),	Add G(shortest) to the queue and explore Goal and H from G
Start→A→B →E→C→F→D →G→H	Goal(G,$\sqrt{2}+2\sqrt{5}$)	Add H(shortest) to the queue and no explore from H

Table 1: The step of Dijkstra method

So the shortest path is Start→E→G→Goal, and the cost is $\sqrt{2}+2\sqrt{5}$.

3.2 Question3.2

Find a path form the "start"-node to the "goal"-node using the A* algorithm. Define a heuristic H(x) and explain each step.

In this question, I choose the Euclidean distance between the node and the goal as the heuristic function, the value of the function is shown in the following table:

Node	Start	A	B	C	D	E	F	G	H
H(x)	4	$\sqrt{17}$	$\sqrt{17}$	2	5	$\sqrt{10}$	1	$\sqrt{5}$	2

Table 2: The value of heuristic function

Now we can use the heuristic function to calculate the total cost, which is defined as $f(x) = g(x) + h(x)$, where $g(x)$ is the cost between two nodes, and $h(x)$ is the heuristic function. The step of A* algorithm is shown in the following table:

Expanded	Q	Explaination
Start	A(Start,$1+\sqrt{17}$), E(Start,$\sqrt{2}+\sqrt{10}$), C(Start,2+2), B(Start,$1+\sqrt{17}$)	Explore ABCE from Start
Start→C	A(Start,$1+\sqrt{17}$), E(Start,$\sqrt{2}+\sqrt{10}$), F(C,3+1), B(Start,$1+\sqrt{17}$)	Add C(shortest) to the queue and explore F from C
Start→C→F	A(Start,$1+\sqrt{17}$), E(Start,$\sqrt{2}+\sqrt{10}$), G(F, $5+\sqrt{5}$), B(Start,$1+\sqrt{17}$)	Add F(shortest) to the queue and explore G from F
Start→C→F →E	A(Start,$1+\sqrt{17}$), G(E, $\sqrt{2}+2\sqrt{5}$), B(Start,$1+\sqrt{17}$)	Add E(shortest) to the queue and explore G from E
Start→C→F →E→A	G(E, $\sqrt{2}+2\sqrt{5}$), B(Start,$1+\sqrt{17}$)	Add A(shortest) to the queue and no exploration from A
Start→C→F →E→A→B	G(E, $\sqrt{2}+2\sqrt{5}$), D(B,3+5)	Add B(shortest) to the queue and explore D from B
Start→C→F →E→A→B→G	D(B,3+5), Goal(G,$\sqrt{2}+2\sqrt{5}$), H(G,$1+\sqrt{2}+2\sqrt{5}+2$)	Add G(shortest) to the queue and explore Goal and H from G

Table 3: The step of A* method

So the shortest path is Start→E→G→Goal, and the cost is $\sqrt{2}+2\sqrt{5}$.

3.3 Question3.3

State the difference in determining the results between A* and Dijkstra and explain why.

Answer: The difference between A* and Dijkstra is that A* uses the heuristic function to calculate the total cost, while Dijkstra only cares about the cost between two nodes The heuristic function contains more infomation that can help us to find the shortest path more quickly, so the result of A* is better than Dijkstra.

4 Dijkstra

Here a more complex directed graph, where the cost, indicated at each edge, is given. Find a path form the "start"-node to the "goal"-node using the Dijkstra algorithm. Show all steps.

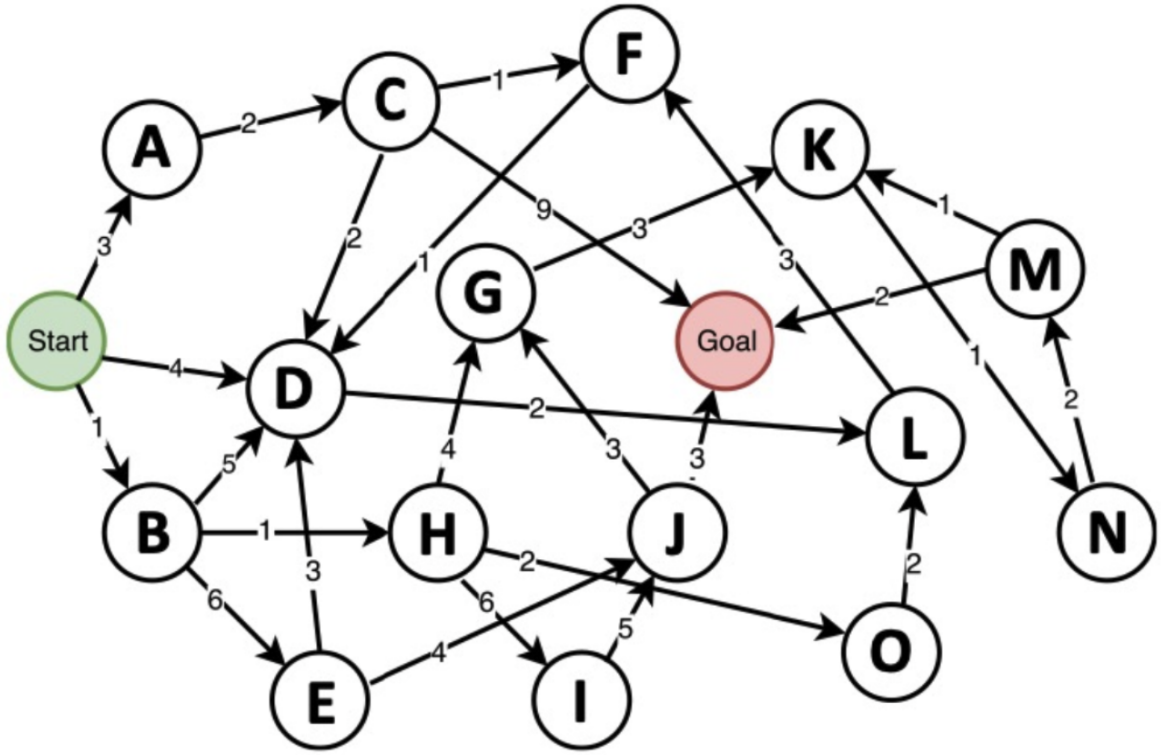

Figure 4: More complex graph

4.1 Question4.1

Find a path form the "start"-node to the "goal"-node using the Dijkstra algorithm. Explain each step.

Expanded	Queue	Explanation
Start	A(Start,3), B(Start,1), D(Start,4)	Explore A and B from Start
Start→B	A(Start,3), D(Start,4), E(B,7), H(B,2)	Add B(shortest) to the queue and explore DEH from B
Start→B→H	A(Start,3), D(Start,4), E(B,7), G(H,6), I(H,8), O(H,4)	Add H(shortest) to the queue and explore GIO from H
Start→B→H →A	D(Start,4), E(B,7), G(H,6), I(H,8), O(H,4),C(A,5)	Add A(shortest) to the queue and explore C from A
Start→B→H →A→O	D(Start,4), E(B,7), G(H,6), I(H,8), L(O,6),C(A,5)	Add O(shortest) to the queue and explore L from O
Start→B→H →A→O→D	E(B,7), G(H,6), I(H,8), L(O,6), C(A,5),	Add D(shortest) to the queue and no exploration from D
Start→B→H →A→O→D →C	E(B,7), G(H,6), I(H,8), L(O,6), Goal(C,14), F(C,6)	Add C(shortest) to the queue and explore F and Goal from C
Start→B→H →A→O→D →C→E	G(H,6), I(H,8), L(O,6), Goal(C,14), F(C,6), J(E,11)	Add E(shortest) to the queue and explore J and Goal from E
Start→B→H →A→O→D →C→E→G	I(H,8), L(O,6), Goal(C,14), F(C,6), J(E,11), K(G,9)	Add G(shortest) to the queue and explore K and Goal from G
Start→B→H →A→O→D →C→E→G →F	I(H,8), L(O,6), Goal(C,14), J(E,11), K(G,9)	Add F(shortest) to the queue and no exploration from F
Start→B→H →A→O→D →C→E→G →F→L	I(H,8), Goal(C,14), J(E,11), K(G,9)	Add L(shortest) to the queue and no exploration from L
Start→B→H →A→O→D →C→E→G →F→L→I	Goal(C,14), J(E,11), K(G,9)	Add I(shortest) to the queue and no exploration from I
Start→B→H →A→O→D →C→E→G →F→L→I →K	Goal(C,14), J(E,11), N(K,10)	Add K(shortest) to the queue and explore N from K
Start→B→H →A→O→D →C→E→G →F→L→I →K→N	Goal(C,14), J(E,11), M(N,12)	Add N(shortest) to the queue and explore M from N
Start→B→H →A→O→D →C→E→G →F→L→I →K→N→J	Goal(C,14), Goal(J,14), M(N,12)	Add J(shortest) to the queue and explore Goal from J
Start→B→H →A→O→D →C→E→G →F→L→I →K→N→J→M	Goal(C,14), Goal(J,14), Goal(M,14)	Add M(shortest) to the queue and explore Goal from M

Table 4: The step of Dijkstra method

So we can get three shortest paths:

- Start→B→E→J→Goal,
- Start→B→H→G→K→N→M→Goal,
- Start→A→C→Goal.

The cost of the three paths are 14, 14, 14 respectively.

4.2 Question4.2

```
def get_node_with_lowest_f_score(open_set, g, stop_node):
    lowest_f_score = float('inf')
    lowest_node = None
    for node in open_set:
        #TODO: Calculate the f_score of node
        if g[node] + heuristic(node, stop_node) < lowest_f_score:
            lowest_f_score = g[node] + heuristic(node, stop_node)
            lowest_node = node
    return lowest_node
```

[6]

Testing your implementation

Now you should get the optimal path for the graph above. Please add the obtained path in your deliverable along with the cost associated.

```
path = aStarAlgo(start_node, stop_node, heuristic)
if path:
    print('Shortest Path:', path)
else:
    print('No path found.')
```

[7]

```
Shortest Path: ['S', 'A', 'C', 'GO']
```

Figure 5: The result of code implementation