DAT565 Assignment 8 – Group 79

Yuchuan Dong - (5 hrs)
Ziyuan Wang - (5 hrs)
Meixi Lin - (5 hrs)

October 22, 2023

Problem 1

The branching factor d of a directed graph is the maximum number of children (outer degree) of a node in the graph. Suppose that the shortest path between the initial state and the goal is of length r.

(a) What is the maximum number of breadth first search (BFS) iterations required to reach the solution in terms of d and r?

Maximum number of breadth first search iterations required is

$$\sum_{i=0}^{r} d^i = \frac{1-d^{r+1}}{1-d}$$

i represents the level of the current node that is being visited.

The number of iterations depends on how many times a new node is visited. In terms of d and r, the first one is the maximum number of children of a node, the other one is the shortest path between the initial and goal states. Therefore, the total amount of nodes visited can be calculated summing the nodes visited, taking into consideration the maximum number of children of a node in each level. As the math formula below.

$$1 + d + d^2 + d^3 + ... + d^r$$

Finally, we can get the formula shown firstly.

(b) Suppose the storing each node requires one unit of memory and the search algorithm stores each entire path as a list of nodes. Hence, storing a path with k nodes requires k units of memory. What is the maximum amount of memory required for BFS in terms of d and r?:

Firstly, we get the formula shown in (a), then in order to calculate the maximum amount of memory required for BFS, we should use $(i + 1)$ as an increase.Therefore, the maximum amount of memory needed for this, is calculated as:

$$\sum_{i=0}^{r} (i+1) * d^i = \frac{1-(r+2)d^{r+1}+(r+1)d^{r+2}}{(1-d)^2}$$

Problem 2

Take the following graph where 0 and 2 are respectively the initial and the goal states. The nodes are to be labelled by 1, 3, and 4. Suppose that we use the depth first search (DFS) method and, in the case of a tie, we choose the smaller label.

(a) Find all the labellings of these three nodes where DFS will never reach the goal.

Two cases, which will never reach the goal, shown in figure 1 and figure 2.

Using DFS to choose a node depends on the node with the smallest label, if we label the top node with a smaller label than the goal node, then the algorithm is always going to choose the top one when positioned at node 4 in figure 1, entering a loop and never reaching the goal node.

So we can label the top node with 1, since 1 is smaller than 2, the algorithm is always going to choose go to the top node, and the other two nodes only have one possible direction they could move to, the number of the label does not affect because the algorithm can only move into one direction.

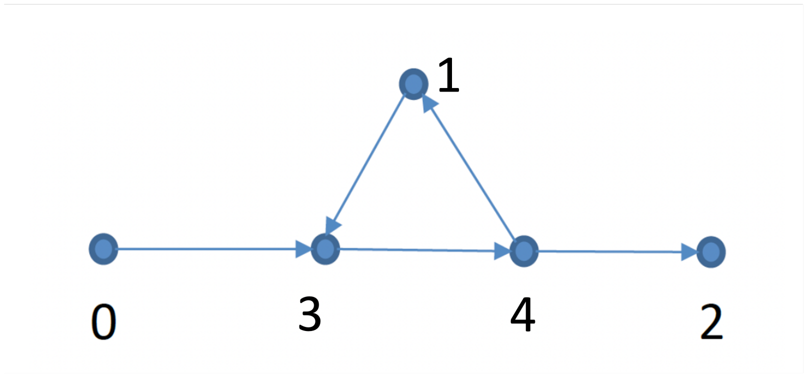

Figure 1: Case 1

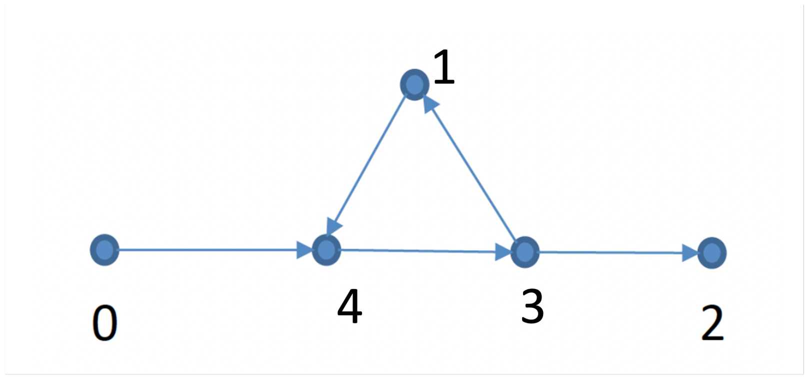

Figure 2: Case 2

(b) Discuss how DFS should be modified to avoid this situation.
Add a variable in the node's data structure, to indicate whether this node have been explored. IF the next smallest node has been explored, we choose the second smallest node.

Problem 3

a

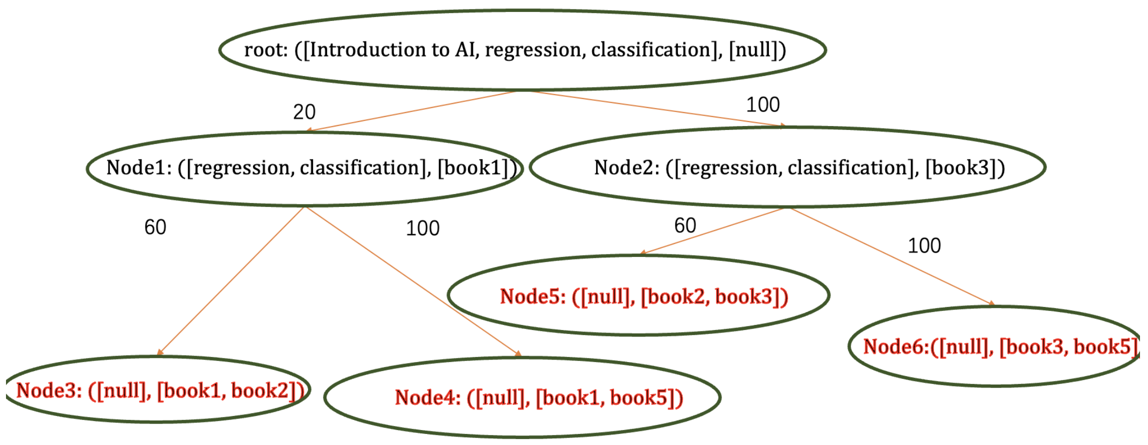
root: ([Introduction to AI, regression, classification], [null])
20
100
Node1: ([regression, classification], [book1])
Node2: ([regression, classification], [book3])
60
100
60
100
Node5: ([null], [book2, book3])
Node6:([null], [book3, book5]
Node3: ([null], [book1, book2])
Node4: ([null], [book1, book5])

Figure 3: Search Space

Figure 3 is the tree for the given topics when searching space. We can conclude from the graph that the goal node is $Node_3$. Frontiers are $Node_2$ and $Node_4$.

b

We formulate the heuristic function

$$h(n) = min\ page(i) \quad i \in (S_{Book} \backslash S_n)$$

if topics in $node_n$ is not empty and S_{Book} is the set of all the books and S_n is the set of books in $node_n$. This formula motivates that in order to get the goal node with the lowest cost, we tend to choose the book with lowest pages to join our set in the following node. And $h(p) \leq cost(p')$ holds for every node which is not goal node. So this heuristic function is admissible.

Problem 4

We use the Manhattan distance as the heuristic, assume g(n) as the cost of the path from the start node to the current node, h(n) as estimate of the cost from the current node to the goal.

a

Iteration1:
Open List: (4,3) g(n)=0, h(n)=4
Closed List: Empty
select the (4,3) for the expansion.
Iteration2:
Open List: (5,3) g(n)=1, h(n)=5;(3,3) g(n)=1, h(n)=∞;(4,4) g(n)=1, h(n)=3; (4,2) g(n)=1, h(n)=5
Closed List: (4,3)
Select the (4,4) for the expansion.
Iteration3:
Open List: (5,4) g(n)=2, h(n)=4;(3,4) g(n)=2, h(n)=2; (5,5) g(n)=2, h(n)=∞; (4,3) g(n)=2, h(n)=4
Closed List: (4,3), (4,4)
Select the (3,4) for the expansion.
Iteration4:
Open List: (4,4) g(n)=3, h(n)=3;(2,4) g(n)=3, h(n)=∞; (3,5) g(n)=3, h(n)=∞; (3,3) g(n)=3, h(n)=∞
Closed List: (4,3), (4,4), (3,4)
Select the (4,4) for the expansion, but (4,4) has existed in the closed set. So we back to the path to the (4,4), update the closed set: (4,3). we choose (5,3) for the expansion
Iteration5:
Open List: (6,3) g(n)=2, h(n)=6;(4,3) g(n)=2, h(n)=4; (5,4) g(n)=2, h(n)=4; (5,2) g(n)=2, h(n)=6
Closed List: (4,3),(5,3)
Select the (4,3) for the expansion.but (4,3) has existed in the closed set. So we back to the (4,3), update the closed set: (4,3). we choose (4,2) for the expansion

b

Figure 4 to figure 6 shows A*, BFS, Best-First-Search algorithm's result. From the result, we found that three different algorithm both find the shortest path, and three algorithm cost 0.5ms, 0.7ms, 0.2ms respectively.

Without doubt, BFS is the most inefficient algorithm due to exploring all the nodes before reaching the goal. Comparing A* algorithm with Best-First-Search algorithm, the latter is more efficient, because it ignored the cost of the path from the start node and only heuristic function plays role in the path search.

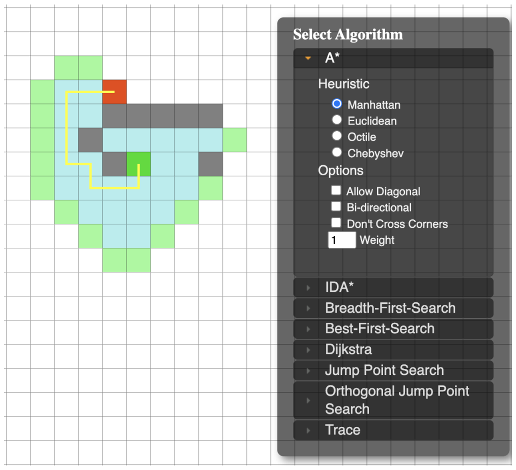

Figure 4: A* algorithm

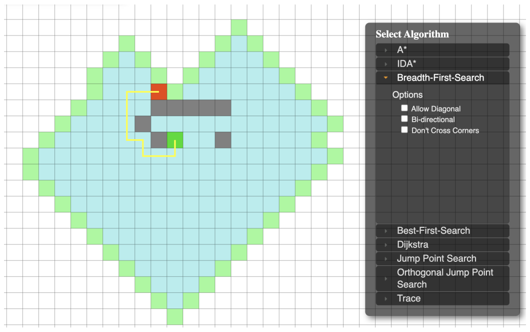

Figure 5: BFS algorithm

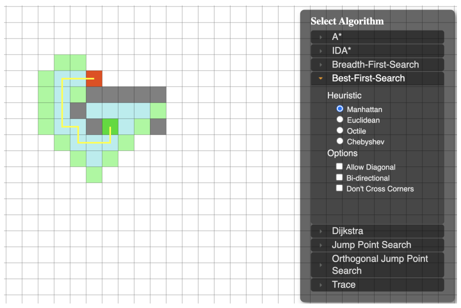

Figure 6: Best-First-Search algorithm

c

We constructed a mazes, which BFS finds the shortest path rather Best-First-Search algoritms failed. Figure 7 and figure 8 showed the different path found by Best-First-Search and BFS, respectively cost 42 and 24 step.
The reason for this is Best-First-Search algorithm failed to estimate the cost according to the heuristic function. When algorithm stand in the cross, he failed to estimate the actual cost of reaching the goal, causing the Best-First-Search to deviate from the shortest path. If the heuristic is inaccurately estimated, the Best-First-Search may follow what the heuristic considers to be a good direction, but a shorter path may actually exist.

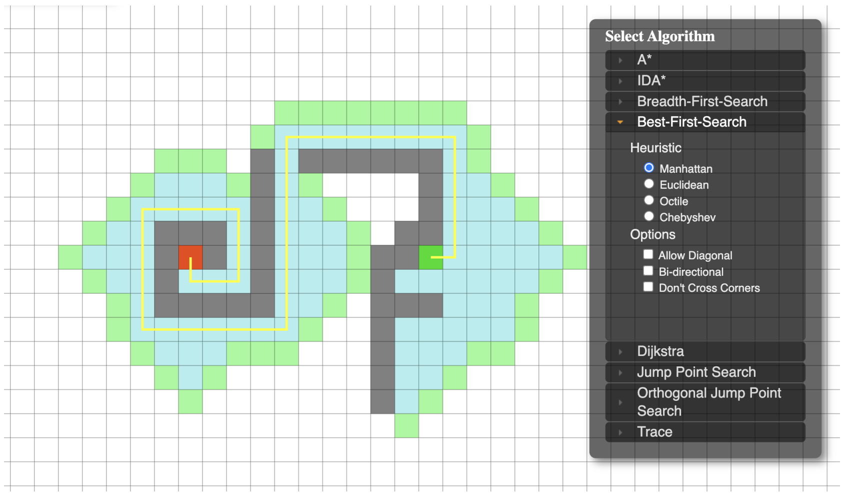

Figure 7: A* algorithm

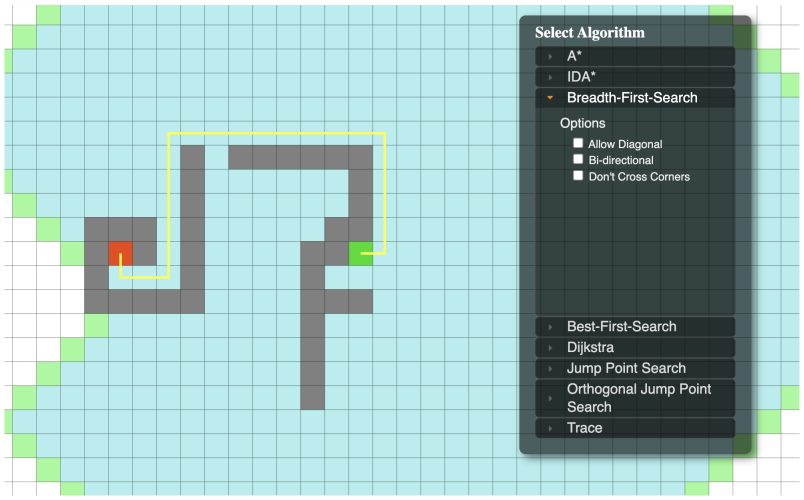

Figure 8: BFS algorithm