Assignment 8

1a)

Since a breadth first search will take the biggest number of iterations when the last node of our search tree is the solution and all nodes has the maximum number of children d, the maximum number of iterations are equal to the number of nodes:

where r is the depth of the solution.

b)

BFS requires maintaining all nodes on the current level in memory at the same time while searching, as all paths are explored simultaneously. This can be memory demanding for big trees as this grows exponentially with depth. The maximum of memory required will depend on the number of nodes at level r, . We need to store all nodes at level r:

For DFS we only need to keep track of our r-nodes at a time as we traverse in depth. Since the width almost always is greater than the height of a tree, BFS requires more memory.

2

Both indicated numberings below will cause a depth first search to be caught in a loop since the node labeled 1 will be chosen over 2 every time the choice arises. This is the only place where there exists a choice in this graph.

Chart, line chart

Description automatically generated

To avoid this the DFS can be amended with a memory of previously visited nodes. It is important that nodes are removed again from the memory when the DFS backtracks after exhausting a path which does not lead to a solution. (Not possible to find such paths here). This is to avoid flagging of loops in DAGs when there only are multiple paths to the same node, but not loops. Imagine the graph above with the paths through 1 inverted, now there are 2 paths to the next to last node, but no loops.

3a)

Diagram

Description automatically generated

cost

3b)

Trivial non-trivial solution is just to let the heuristic be the shortest segment:

However, a better solution is to let the heuristic be the shortest of segments covering a remaining topic.

4a)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  | G |  |  |  |  |  |
|  |  | X | x | x | x | x |  |
|  | x | 2 | 1 | 4 |  |  |  |
|  |  | x | s | 3 |  | x |  |
|  |  |  | 5 |  |  |  |  |
|  |  |  |  |  |  |  |  |

4b)

A\*

Chart

Description automatically generated

BFS

Chart

Description automatically generated

Best-first-search

Chart

Description automatically generated

We can see that BFS explores equally in all directions, while both A\* and best-first-search prioritizes where to search. Best-first-search explores from the box with the shortest manhatten distance to the goal, whilst A\* explores from the box with the lowest sum of manhatten distance to target and length of path from start. Its clear that BFS is the slowest as both other algorithms are BFS with some more sophisticated tactics for exploring. Best-first-search can be faster than A\*, but can also not be slower as it can be tricked to explore suboptimal solutions. However best-first-search has the potential to be the fastest.

For this problem all algorithms find the optimal solution, but only A\* and BFS guarantees this. The problem below shows a situation where best-first-search does not find the optimal solution.

Chart

Description automatically generated