Gabriel Shultz
2/9/2022

Homework 2 AI

Q1.)

Search Algorithm	Expanded node order	Solution
DFS (tree search)	S → B → C → G	S → B → C → G
DFS (graph search)	S → B → C → G	S → B → C → G
BFS (tree search)	S → B → E → F → C → F → D → C → E → G	S → B → C → G
BFS (graph)	S → B → E → F → C → D → G	S → B → C → G
UCS (tree)	S → E → F → E → B → C → D → F, D, C, G	S → F → C → G
UCS (graph)	S → E → F → B → C → D → G	S → F → C → G
Greedy (tree)	S → E → D → S E D ...	None
Greedy (graph)	S → E → D → F → C → G	S → F → C → G
A* (tree)	S → E → F → E → C → B → G	S → F → C → G
A* (graph)	S → E → F → C → B → G	S → F → C → G

really hard

Q2.)

DFS (tree)

S → B → A → C → D → A → C → D ...

a.)

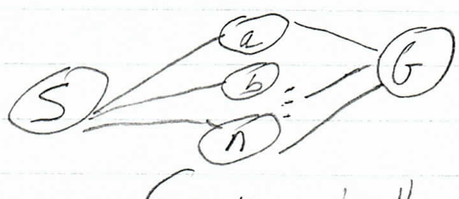
Iterative Deepening

S → B → M → P → S → B → A → M → G

DFS tree search fails to return a solution as it is caught in a cycle. Therefore not optimal.

Iterative Deepening returns the path S → M → G which is optimal and complete. It will return the shortest path if the solution exists. (Complete and optimal in number of actions)

b.)



n different paths

Iterative Deepening would make $O(n)$ expansions where DFS would only make $O(1)$
2 in this case.

$S \to A \leftarrow B \to G$

like

C.) DFS graph search will fail if the goal is not reachable from the start node. (Not fully connected graph)

If there is a solution, DFS graph search will find it.

G1

Q3.) a.) A^* tree and A^* graph both return

~~$S \to B \to M \to D \to P \to G$~~

$S \to D \to P \to G$ as the solution

b.) A^* returns the same solution as UCS in this situation, it is optimal.

C.) UCS tree and graph search also return

$S \to D \to P \to G$

G2

~~$S \to B \to$~~ $P \to D \to E \to G$ ← Expanding node thought process.

a.)

Solution $\Rightarrow S \to B \to P \to G$ is returned by A^*.

b.) This is the optimal solution, so heuristic is fine

C.) $S \to B \to P \to G$ is returned by UCS.

~~So~~ This is the optimal solution which proves the A^* heuristic is okay.

UCS is complete and optimal. A^* tries to make it more efficient.

Q4.) Node expansion

$S \rightarrow F \rightarrow H \rightarrow M \rightarrow G$

Solution returned by UCS

$S \rightarrow H \rightarrow G$

This is not the optimal solution because there is a path with a negative cost. This breaks an assumption made when discussing the optimality of UCS.

Q5.)

a.) The branching factor of this state space would be 4 because each point has 4 successors.

b.) $4^K + 1$ states at depth K.

c.) Breadth first tree search will eventually reach point (X, Y). However, it will become very expensive as the number of graph cycles grow with every level.

The number of operations would be in the order of

$|X| + |Y|$ is roughly the depth to goal

$$\sum_{i=0}^{|X|+|Y|} 4^i$$ is what seems to be the answer. Each expanded node generate 4 more to expand.

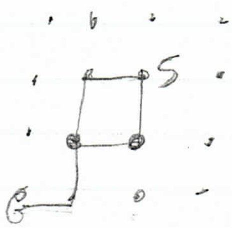
d.) $\underline{(|X| + |Y|) * 4 + 1}$ max number of nodes expanded.

depth

start state

e.) Yes, this heuristic adequately models a distance function. This would steer A* for example well.

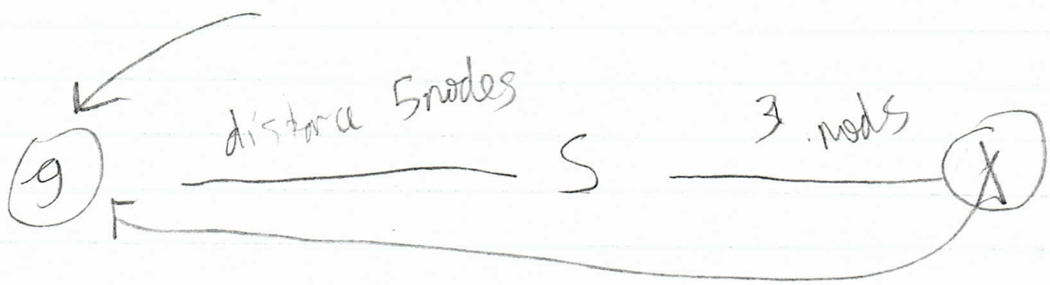
f.)



Number of nodes expanded is distance between start and goal. Or in other words the value of h at start.

g.) Yes, because nodes around the discontinuity will be expanded and the cost for going around such obstacle. Greedy would get stuck, whereas A* would use its BFS like qualities to look further.

h.) /



Unless a situation like the one above occurs, where a node further away from the goal in space links to it. Then the heuristic is still working.

However, the heuristic is not less than the actual cost (# of actions) to get to the goal. Since these "Portals" break the heuristic's assumption of distance, I would say it makes the heuristic not Admissable

Q6 If one wanted to discourage the use of a path, one could make the heuristic in that region inadmissable. This would destroy any gurantees about optimality, but would achieve the goal of discouraging that path. For instance, the user of a navigation app wanting to avoid toll roads.

Q7.) What kind of search does this perform for....?

$w = 0 \Rightarrow$ UCS

$w = 1 \Rightarrow$ Weighted A^*

$w = 4 \Rightarrow$ Greedy $\Rightarrow$ not complete

Algorithm is complete when $0 \leq w < 4$.