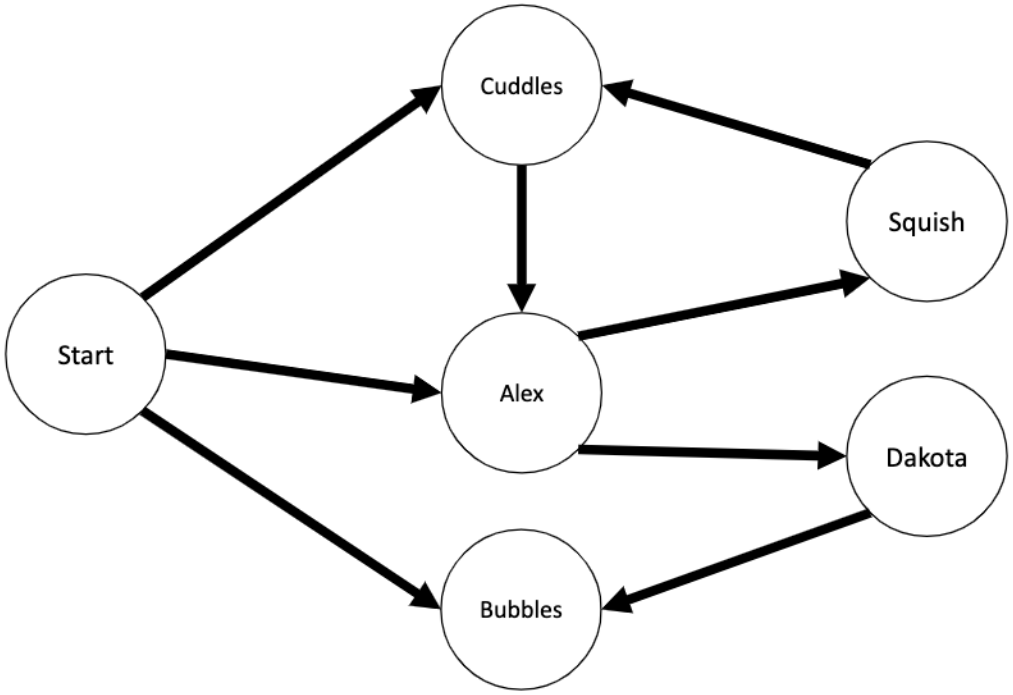
CS 188 Spring 2021 Introduction to Artificial Intelligence

Final Review Search Solu

Q1. Search: Snail search for love

Scorpblorg the snail is looking for a mate. It can visit different potential mates based on a trail of ooze to nearby snails, and then test them for chemistry, as represented in the below graph, where each node represents a snail. In all cases, nodes with equal priority should be visited in alphabetical order.



(a) Simple search

In this part, assume that the only match for Scorpblorg is Squish (i.e. Squish is the goal state). Which of the following are true **when searching the above graph**?

(i) BFS Tree Search expands more nodes than DFS Tree Search ● True ○ False

DFS Tree Search expands the path Alex, then Dakota, then Bubbles, then Squish. In contrast, BFS Tree Search expands Alex, Bubbles, Cuddles, Alex, and Dakota before opening Squish.

(ii) DFS Tree Search finds a path to the goal for this graph ● True ○ False

DFS Tree Search does not get stuck in any loops on this graph and does return the solution path.

(iii) DFS Graph Search finds the shortest path to the goal for this graph ● True ○ False

DFS Graph Search does return the shortest solution path.

(iv) If we remove the connection from Cuddles → Alex, can DFS Graph Search find a path to the goal for the altered graph? ● Yes ○ No

Yes, DFS Graph Search will return the correct path, regardless of the connection from Cuddles → Alex.

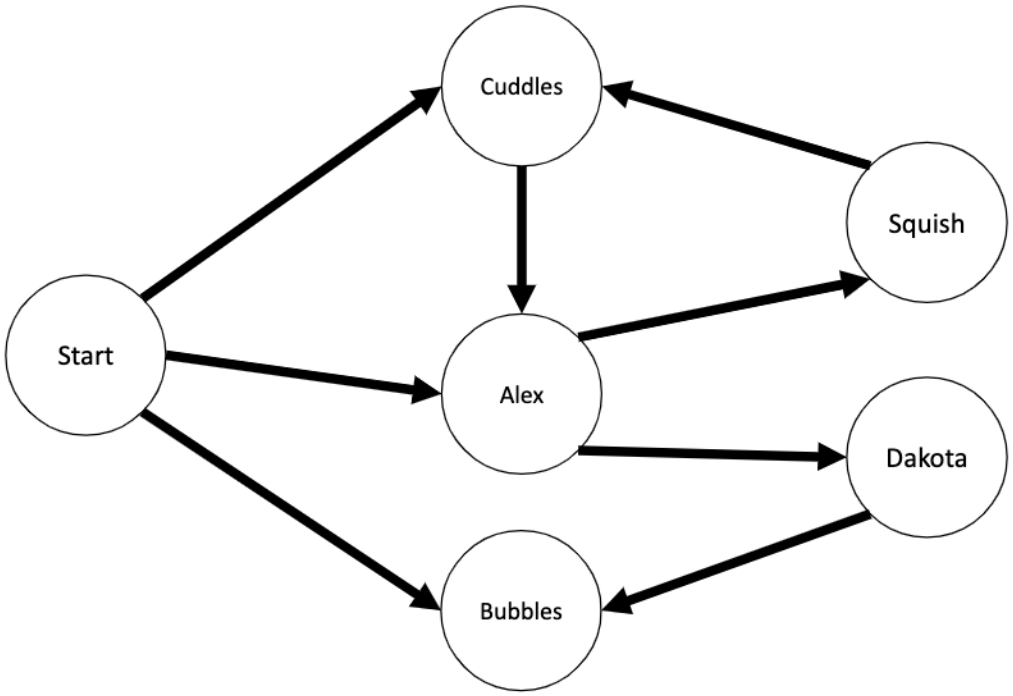
(b) Third Time's A Charm

Now we assume that Scorpblorg's mate preferences have changed. The new criteria she is looking for in a mate is that she has **visited the mate twice before** (i.e. when she visits any state for the third time, she has found a path to the goal).

(i) What should the most simple yet sufficient new state space representation include?

- ■ The current location of Scorpblorg
- ☐ The total number of edges travelled so far
- ☐ An array of booleans indicating whether each snail has been visited so far
- ■ An array of numbers indicating how many times each snail has been visited so far
- ☐ The number of distinct snails visited so far

The current location is needed to generate successors. The array of number indicating how many times each snail has been visited so far is needed for the goal test. A list of boolean is insufficient because we need to revisit more than once. Other information is redundant



(The graph is copied for your convenience)

(ii) DFS Tree Search finds a path to the goal for this graph ● True ○ False

DFS Tree Search does not get stuck in any loops on this graph and does return the solution path.

(iii) BFS Graph Search finds a path to the goal for this graph ● True ○ False

Revisiting a location is allowed with BFS Graph search because the "visited" set keep track of the augmented states, which means revisiting any location is right

(iv) If we remove the connection from Cuddles → Alex, can DFS Graph Search finds a path to the goal for the altered graph? ○ Yes ● No

Meeting three time requires the Alex, Cuddles, Squish cycle. Since it is the only cycle, removing it will prevent Scorpblorg from meeting any mate three times

We continue as in part (b) where the goal is still to find a mate who is visited for the third time.

(c) Costs for visiting snails

Assume we are using Uniform cost search and we can now add costs to the actions in the graph.

(i) Can one assign (non-negative) costs to the actions in the graph such that the goal state returned by UCS (Tree-search) changes? ● Yes ○ No

Yes, if the costs are all equal, UCS will return the same goal state as BFS (Tree-search): Alex. However, putting a very large cost on the path from Cuddles to Alex will change the goal state to Cuddles. Other Examples exist.

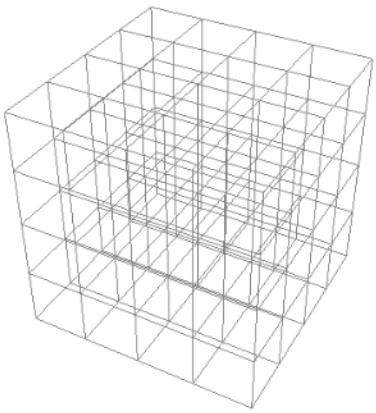
(ii) Can one assign (potentially negative) costs to the actions in the graph such that UCS (Tree-search) will never find a goal state? ○ Yes ● No

No, regardless of the costs on the graph, eventually a state will be re-visited, resulting in a goal state.

Q2. Search in 3D Maze

Imagine you are the Spider man. Your friend Superman was captured by the evil Spider man somewhere in this mystical 3D maze (Fig 1), (x_t, y_t, z_t). This maze is a *infinite* 3D grid world. You are located at $(0, 0, 0)$ right now and want to come up with a plan to rescue the Superman. Even though you are the Spider man, you can only travel along the wires, not through the space.

3D Maze



(a) What is the branch factor b in this space?

6

(b) How many distinct states can you reach at depth k?

$4k + 4k(k-1) + 2 = 4k^2 + 2$

(c) If you run BFS-tree search, how many nodes would you have expanded up to the goal state? What about BFS-graph search?

Without checking repeated states, the BFS will expend exponentially many nodes on the order of $6^{x_t+y_t+z_t}$. On the other hand, BFS-graph search would expand on the order of $(x_t + y_t + z_t)^3$ many states.

(d) Assume each edge has a cost of 1. Let the state be (u, v, w). Which of the following heuristics are admissible? Select all that apply.

- ■ $h1(u, v, w) = \sqrt{(x_t - u)^2 + (y_t - v)^2 + (z_t - w)^2}$
- ■ $h2(u, v, w) = |x_t - u| + |y_t - v| + |z_t - w|$
- ■ $h3(u, v, w) = \sqrt{|x_t - u|} + \sqrt{|y_t - v|} + \sqrt{|z_t - w|}$
- ☐ $h1(u, v, w) = (x_t - u)^2 + (y_t - v)^2 + (z_t - w)^2$

(e) Approximately how many nodes would you expand if you use heuristic $h1$?

○ $|x_t y_t z_t|$ ○ $(x_t)^2 + (y_t)^2 + (z_t)^2$ ● $\sqrt{(x_t - u)^2 + (y_t - v)^2 + (z_t - w)^2}$ ○ $|x_t| + |y_t| + |z_t|$

What about $h2$?

● $|x_t y_t z_t|$ ○ $(x_t)^2 + (y_t)^2 + (z_t)^2$ ○ $\sqrt{(x_t - u)^2 + (y_t - v)^2 + (z_t - w)^2}$ ○ $|x_t| + |y_t| + |z_t|$

(f) If the evil Spider man destroys half of the links in this grid, would the heuristics h_1 and h_2 be admissible?

Yes, they are, because removing links only cause more detours.

(g) In expectation (assume random tie-breaking), how many nodes would you expand before hitting the goal if you use UCS tree search, DFS graph search, or greedy search instead of A* search? Assume each path has cost 1 and heuristic $h1$.

UCS tree search would explore $6^{x_t+y_t+z_t}$ many nodes. DFS graph search could go to infinity because this is an infinite grid world. Greedy search would have the same complexity as A* in this case, which is $\sqrt{(x_t-u)^2+(y_t-v)^2+(z_t-w)^2}$