CS 188 Spring 2023 Introduction to Artificial Intelligence

Exam Prep 1 Solutions

Q1. SpongeBob and Pacman (Search Formulation)

Pacman bought a car, was speeding in Pac-City, and SpongeBob wasn't able to catch him. Now Pacman has run out of gas, his car has stopped, and he is currently hiding out at an undisclosed location. In this problem, you are on SpongeBob's side, tryin' to catch Pacman!

There are still p of SpongeBob's cars in the Pac-city of dimension m by n. In this problem, **all of SpongeBob's cars can move, with two distinct integer controls: throttle and steering, but Pacman has to stay stationary**. Spongebob's cars can control both the throttle and steering for each step. Once one of SpongeBob's cars takes an action which lands it in the same grid as Pacman, Pacman will be caught and the game ends.

Throttle: $t_i \in \{1, 0, -1\}$, corresponding to {Gas, Coast, Brake}. This controls the **speed** of the car by determining its acceleration. The integer chosen here will be added to the velocity for the next state. For example, if a SpongeBob car is currently driving at 5 grid/s and chooses Gas (1) it will be traveling at 6 grid/s in the next turn.
Steering: $s_i \in \{1, 0, -1\}$, corresponding to {Turn Left, Go Straight, Turn Right}. This controls the **direction** of the car. For example, if a SpongeBob car is facing North and chooses Turn Left, it will be facing West in the next turn.

(a) Suppose you can **only control 1 SpongeBob car**, and have absolutely no information about the remainder of $p - 1$ SpongeBob cars, or where Pacman has stopped to hide. Also, the SpongeBob cars can travel up to 6 grid/s so $0 \leq v \leq 6$ at all times.

(i) What is the **tightest upper bound** on the size of state space, if your goal is to use search to plan a sequence of actions that guarantees Pacman is caught, no matter where Pacman is hiding, or what actions other SpongeBob cars take. Please note that your state space representation must be able to represent **all** states in the search space.

$28mn * 2^{mn}$

There are mn positions in total. At each legal position, there are 7 possible speeds (0, 1, 2, 3, 4, 5, 6), so a factor of 7 is multiplied. In addition, since change of direction depends on orientation of the car, another factor of 4 is multiplied.

The only sequence of actions which guarantees that Pacman is caught is a sequence of actions which visits every location. Thus, we also need to a list of $m * n$ boolean to keep track of whether we have visited a specific grid location, and that is another factor of 2^{mn}

(ii) What is the maximum branching factor? Your answer may contain integers, m, n.

9

3 possible throttle inputs, and 3 possible steering inputs. The list of boolean does not affect the branching factor.

(iii) Which algorithm(s) is/are guaranteed to return a path passing through all grid locations on the grid, if one exists?

- ☐ Depth First Tree Search
- ■ Breadth First Tree Search
- ■ Depth First Graph Search
- ■ Breadth First Graph Search

Please note the list of boolean is in the state space representation, so we can revisit the same grid position if we have to.

(iv) Is Breadth First Graph Search guaranteed to return the path with the shortest number of **time steps**, if one exists?

- ● Yes
- ○ No

Breadth First Graph Search is guranteed to return the path with the shortest amount of time, because each edge here represent moving for 1 unit of time.

(b) Now let's suppose you can control **all** p SpongeBob cars at the same time (and know all their locations), but you still have no information about where Pacman stopped to hide

(i) Now, you still want to search a sequence of actions such that the paths of p SpongeBob cars combined **pass through all $m * n$ grid locations**. Suppose the size of the state space in part (a) was N_1, and the size of the state space in this part is N_p. Please select the correct relationship between N_p and N_1.

○ $N_p = p * N_1$ ○ $N_p = p^{N_1}$ ○ $N_p = (N_1)^p$ ● None of the above

In this question, we only need one boolean list of size mn to keep track of whether we have visited a specific grid location. So the size of the state space is bounded by $N_p = (28mn)^p 2^{mn}$, which is none of the above.

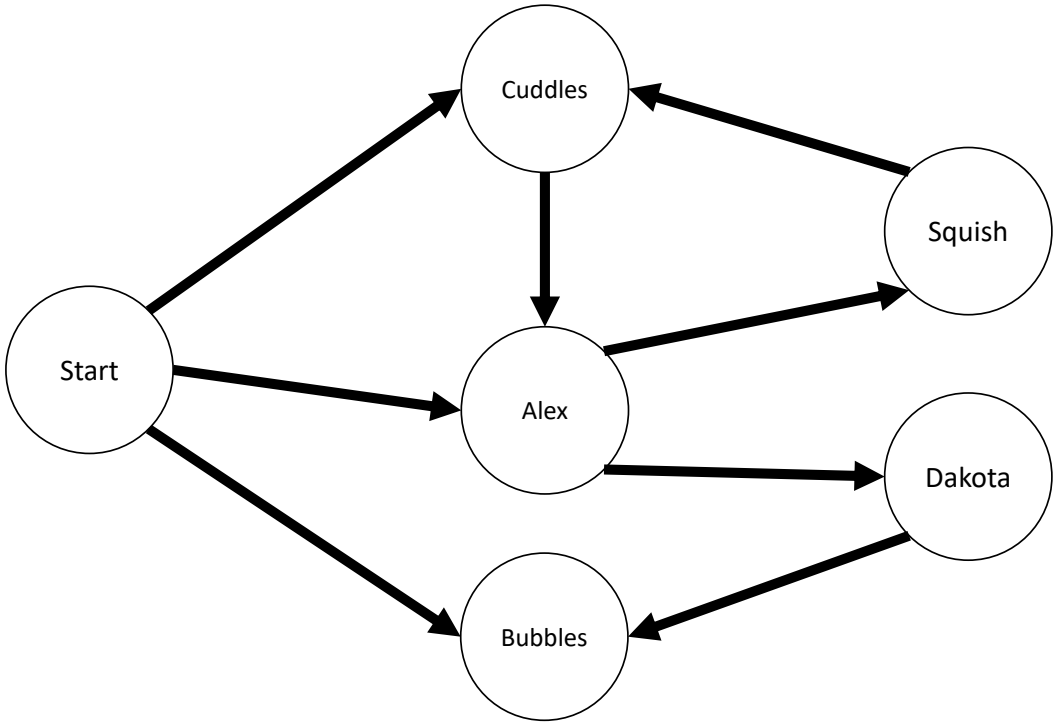
(ii) Suppose the maximum branching factor in part (a) was b_1, and the maximum branching factor in this part is b_p. Please select the correct relationship between b_p and b_1.

○ $b_p = p * b_1$ ○ $b_p = p^{b_1}$ ● $b_p = (b_1)^p$ ○ None of the above

For example, the case of $p = 2$ means two cars can do all 9 options, so the branching factor is $9^2 = 81$. In general, the branching factor is then b_1^p.

Q2. 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

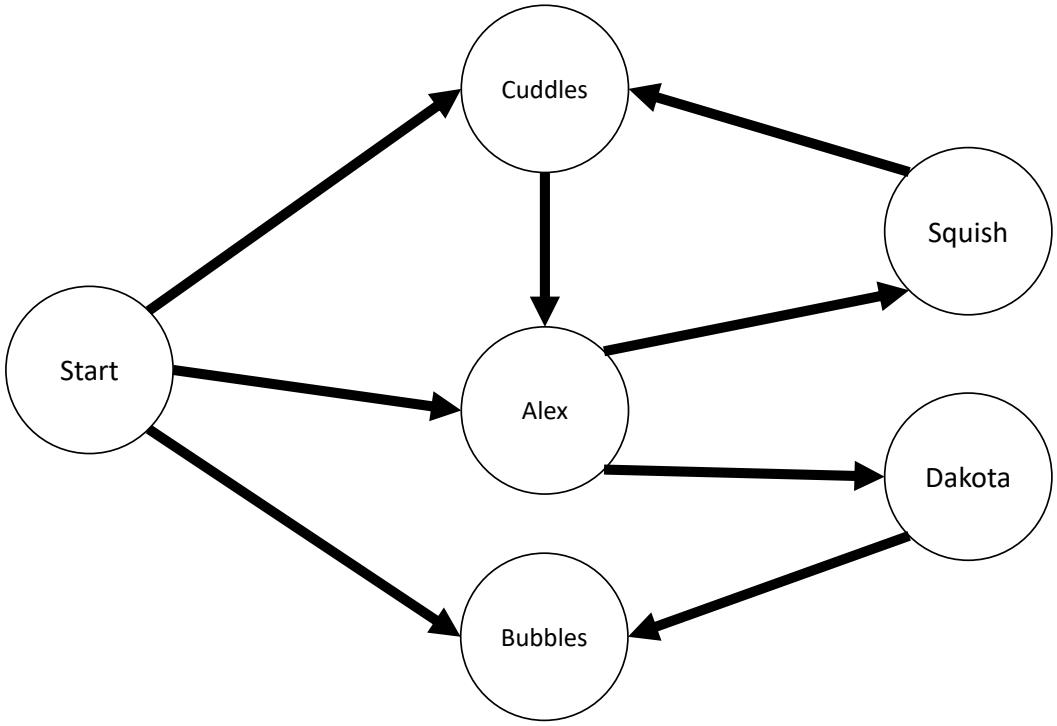
Yes, DFS Graph Search will return the correct path, regardless of the connection from Cuddles → Alex.

(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.