CS 188
Summer 2024

MT Review Search Solutions

Q1. They See Me Rolling (Search Problem)

Pacman buys a car to start Rolling in the Pac-City! But driving a car requires a new set of controls because he can now travel faster than 1 grid per turn (second). Instead of solely moving [North, South, East, West, Stop], Pacman's car has two distinct integers to control: throttle, and steering.

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 his velocity for the next state. For example, if Pacman is currently driving at 5 grid/s and chooses Gas he will be traveling at 6 grid/s in the next turn.
Steering: $s_i \in \{1, 0, -1\}$, corresponding to {Turn Left, Neutral, Turn Right}. This controls the **direction** of the car. For example, if he is facing North and chooses Turn Left he will be facing West in the next turn.

(a) Suppose Pac-city has dimension m by n, but only $k < mn$ squares are legal roads. The speed limit of Pac-city is 3 grid/s. For this sub-part only, suppose Pacman is a law-abiding citizen, so $0 \leq v \leq 3$ at all time, and he only drives on legal roads.

(i) Without any additional information, what is the **tightest upper bound** on the size of state space, if he wants to search a route (not necessarily the shortest) from his current location to anywhere in the city. Please note that your state space representation must be able to represent **all** states in the search space.

○ $4mn$ ○ $4k$ ○ $12mn$ ○ $12k$ ○ $16mn$ [✓ $16k$] ○ $48mn$ ○ $48k$

Only legal grids count, so there are k legal position. At each legal position, there are 4 possible speed (0, 1, 2, 3), so a factor of 4 is multiplied. In addition, since change of direction depends on orientation of the car, another factor of 4 is multiplied. The size of state space is bounded by $k * 4 * 4 = 16k$

(ii) What is the maximum branching factor? The answer should be an integer. 9
3 possible throttle inputs, and 3 possible steering inputs.

(iii) Which algorithm(s) is/are guaranteed to return a path between two points, if one exists?

☐ Depth First Tree Search [✓ Breadth First Tree Search]
[✓ Depth First Graph Search] [✓ Breadth First Graph Search]

(iv) Is Breadth First Graph Search guaranteed to return the path with the shortest grid distance?

○ Yes [✓ No]

The 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 remove the constraint that Pacman follows the speed limit. Now Pacman's speed is limited by the mechanical constraints of the car, which is 6 grid/s, double the speed limit.

Pacman is now able to drive twice as fast on the route to his destination. How do the following properties of the search problem change as a result of being able to drive twice as fast?

(i) Size of State Space:

[✓ Increases] ○ Stays the same ○ Decreases

At each legal position, there are 7 possible speed (0,1, 2, 3, 4, 5, 6), so a factor of 7 is multiplied. size of state space is now k * 7 * 4 = 28k

(ii) Maximum Branching Factor:

○ Increases [✓ Stays the same] ○ Decreases

Branching factor is independent of top speed

For the following part, **mark all choices that could happen on any graph**

(iii) The number of nodes expanded with **Depth** First Graph Search:

[✓ Increases] [✓ Stays the same] [✓ Decreases]

Anything can happen with Depth First Graph Search. For example, too fast can jump past an intersection that would lead to a shorter path to the goal

(c) Now we need to consider that there are $p > 0$ police cars waiting at $p > 0$ distinct locations trying to catch Pacman riding dirty!! All police cars are stationary, but once Pacman takes an action which lands him in the same grid as one police car, Pacman will be arrested and the game ends.

Pacman wants to find a route to his destination, without being arrested. How do the following properties of the search problem change as a result of avoiding the police?

(i) Size of State Space:

○ Increases [✓ Stays the same] ○ Decreases

The size of statespace is still the same because all the factors in state space calculation is the same

(ii) Maximum Branching Factor:

○ Increases [✓ Stays the same] ○ Decreases

Branching factor is independent of police presense

For the following part, **mark all choices that could happen on any graph**

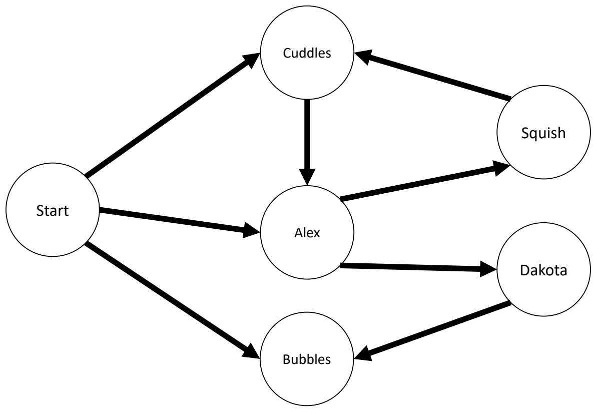
(iii) Number of nodes expanded with **Breadth** First Graph Search:

[✓ Increases] [✓ Stays the same] [✓ Decreases]

Again anything could happen with BFS. Policemen could block out misleading paths to decrease the number of expansion. They could also not affect anything. They could also block the closest goal causing you to explore more nodes.

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

(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 $\rightarrow$ 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.

Q3. Power Pellets

Consider a Pacman game where Pacman can eat 3 types of pellets:

- Normal pellets (n-pellets), which are worth one point.
- Decaying pellets (d-pellets), which are worth $max(0, 5 - t)$ points, where t is time.
- Growing pellets (g-pellets), which are worth t points, where t is time.

The location and type of each pellet is fixed. The pellet's point value stops changing once eaten. For example, if Pacman eats one g-pellet at $t = 1$ and one d-pellet at $t = 2$, Pacman will have won $1 + 3 = 4$ points.

Pacman needs to find a path to win at least 10 points but he wants to minimize distance travelled. The cost between states is equal to distance travelled.

(a) Which of the following must be including for a minimum, sufficient state space?

🟥 Pacman's location
☐ Location and type of each pellet
☐ How far Pacman has travelled
🟥 Current time
☐ How many pellets Pacman has eaten and the point value of each eaten pellet
🟥 Total points Pacman has won
🟥 Which pellets Pacman has eaten

A state space should include which pellets are left on the board, the current value of pellets, Pacman's location, and the total points collected so far. With this in mind:
(1) The starting location and type of each pellet are not included in the state space as this is something that does not change during the search. This is analogous to how the walls of a Pacman board are not included in the state space.
(2) How far Pacman has travelled does not need to be explicitly tracked by the state, since this will be reflected in the cost of a path.
(3) Pacman does need the current time to determine the value of pellets on the board.
(4) The number of pellets Pacman has eaten is extraneous.
(5) Pacman must track the total number of points won for the goal test.
(6) Pacman must know which pellets remain on the board, which is the complement of the pellets he has eaten.

(b) Which of the following are admissible heuristics? Let x be the number of points won so far.

🟥 Distance to closest pellet, except if in the goal state, in which case the heuristic value is 0.
☐ Distance needed to win $10 - x$ points, determining the value of all pellets as if they were n-pellets.
🟥 Distance needed to win $10 - x$ points, determining the value of all pellets as if they were g-pellets (i.e. all pellet values will be t.)
☐ Distance needed to win $10 - x$ points, determining the value of all pellets as if they were d-pellets (i.e. all pellet values will be $max(0, 5 - t)$.
☐ Distance needed to win $10 - x$ points assuming all pellets maintain current point value (g-pellets stop increasing in value and d-pellets stop decreasing in value)
☐ None of the above

(1) Admissible; to get 10 points Pacman will always have to travel at least as far as the distance to the closest pellet, so this will always be an underestimate.
(2) Not admissible; if all the pellets are actually g-pellets, assuming they are n-pellets will lead to Pacman collecting more pellets in more locations, and thus travel further.

(3) Ambiguous; if pellets are n-pellets or d-pellets, Pacman will generally have to go further, except at the beginning of the game when d-pellets are worth more, in which case this heuristic will over-estimate the cost to the goal. However, if Pacman is allowed to stay in place with no cost, then this heuristic is admissable because the heuristic will instead calculate all pellet values as 10. This option was ignored in scoring.
(4) Not admissible; if pellets are n-pellets or g-pellets, Pacman would have an overestimate.
(5) Not admissible; if pellets are g-pellets, then using the current pellet value might lead Pacman to collect more locations, and thus travel further than necesarry.

(c) Instead of finding a path which minimizes distance, Pacman would like to find a path which minimizes the following:

$$C_{new} = a * t + b * d$$

where t is the amount of time elapsed, d is the distance travelled, and a and b are non-negative constants such that $a + b = 1$. Pacman knows an admissible heuristic when he is trying to minimize time (i.e. when $a = 1, b = 0$), h_t, and when he is trying to minimize distance, h_d (i.e. when $a = 0, b = 1$).
Which of the following heuristics is guaranteed to be admissible when minimizing C_{new}?

☐ $mean(h_t, h_d)$ [✓ $min(h_t, h_d)$] ☐ $max(h_t, h_d)$ [✓ $a * h_t + b * h_d$]
☐ None of the above

For this question, think about the inequality $C_{new} = a * t + b * d \geq a * h_t + b * h_d$. We can guarantee a heuristic h_{new} is admissible if $h_{new} \leq a * h_t + b * h_d$
(1) If $a = b$, $0.5 * h_t + 0.5 * h_d$ is not guaranteed to be less than $a * h_t + b * h_d$, so this will not be admissible.
(2) $min(h_t, h_d) = a * min(h_t, h_d) + b * min(h_t, h_d) \leq a * h_t + b * h_d$
(3) $max(h_t, h_d)$ will be greater than $a * h_t + b * h_d$ unless $h_t = h_d$, wo this will not be admissible.
(4) Admissible.