CS 188 Spring 2022 — Search

Q1. 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

(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

(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

Q2. Rubik's Search

A Rubik's cube has about 4.3×10^{19} possible configurations, but any configuration can be solved in 20 moves or less. We pose the problem of solving a Rubik's cube as a search problem, where the states are the possible configurations, and there is an edge between two states if we can get from one state to another in a single move. Thus, we have 4.3×10^{19} states. Each edge has cost 1. Since we can make 27 moves from each state, the branching factor is 27. Since any configuration can be solved in 20 moves or less, we have $h^*(n) \leq 20$.

For each of the following searches, estimate the approximate number of states expanded. Mark the option that is closest to the number of states expanded by the search. Assume that the shortest solution for our start state takes exactly 20 moves. Note that 27^{20} is much larger than 4.3×10^{19}.

(a) DFS Tree Search

(i) Best Case: ○ 20 ○ 4.3×10^{19} ○ 27^{20} ○ ∞ (never finishes)

(ii) Worst Case: ○ 20 ○ 4.3×10^{19} ○ 27^{20} ○ ∞ (never finishes)

(b) DFS graph search

(i) Best Case: ○ 20 ○ 4.3×10^{19} ○ 27^{20} ○ ∞ (never finishes)

(ii) Worst Case: ○ 20 ○ 4.3×10^{19} ○ 27^{20} ○ ∞ (never finishes)

(c) BFS tree search

(i) Best Case: ○ 20 ○ 4.3×10^{19} ○ 27^{20} ○ ∞ (never finishes)

(ii) Worst Case: ○ 20 ○ 4.3×10^{19} ○ 27^{20} ○ ∞ (never finishes)

(d) BFS graph search

(i) Best Case: ○ 20 ○ 4.3×10^{19} ○ 27^{20} ○ ∞ (never finishes)

(ii) Worst Case: ○ 20 ○ 4.3×10^{19} ○ 27^{20} ○ ∞ (never finishes)

(e) A* tree search with a perfect heuristic, h*(n), Best Case

○ 20 ○ 4.3×10^{19} ○ 27^{20} ○ ∞ (never finishes)

(f) A* tree search with a bad heuristic, h(n) = 20 - h*(n), Worst Case

○ 20 ○ 4.3×10^{19} ○ 27^{20} ○ ∞ (never finishes)

(g) A* graph search with a perfect heuristic, h*(n), Best Case

○ 20 ○ 4.3×10^{19} ○ 27^{20} ○ ∞ (never finishes)

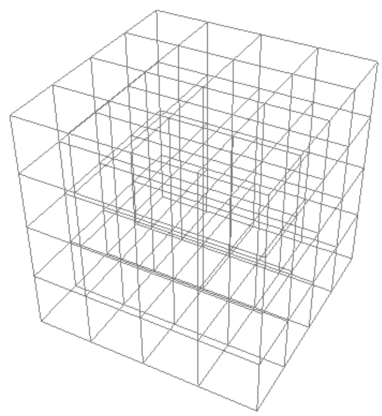
(h) A* graph search with a bad heuristic, h(n) = 20 - h*(n), Worst Case

○ 20 ○ 4.3×10^{19} ○ 27^{20} ○ ∞ (never finishes)

Q3. **[Optional]** 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?

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

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

(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?

(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 $h1$.