CS 188 Summer 2024 Review: Search/CSP's/Games Solutions

Q1. Search

(a) Rubik's Search

Note: You do not need to know what a Rubik's cube is in order to solve this problem.

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. Note that the state space graph does contain cycles. 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}.

(i) DFS Tree Search

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

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

(ii) DFS graph search

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

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

(iii) BFS tree search

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

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

(iv) BFS graph search

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

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

(v) A* tree search with a perfect heuristic, $h^*(n)$, Best Case

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

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

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

(vii) A* graph search with a perfect heuristic, $h^*(n)$, Best Case

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

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

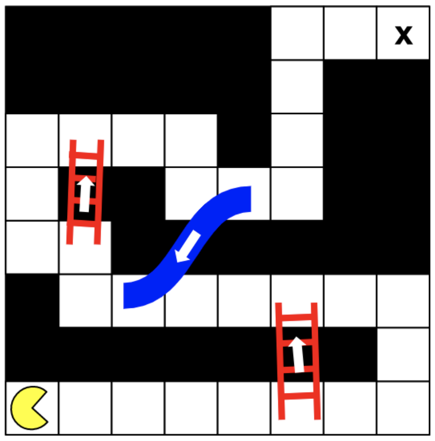
○ 20 [✓ 4.3×10^{19}] ○ 27^{20} ○ ∞ (never finishes)

SID: ______________

Q5. [8 pts] Slides and Ladders

Pacman is in a two-dimensional grid world with ladders and slides. There is a single path to his goal, and the goal is at the end of the path so there are no squares after the goal. Ladders carry Pacman to a square on the path that is closer to his goal, and slides carry him to a square that is farther. Ladders and slides may cross, but each square can only be the beginning or the end of one ladder or slide. (Note: this figure is only for demonstration purposes. For all the questions below, the size of the grid, as well as the locations of ladders and slides, may vary.)

At each timestep, Pacman normally moves forward one step towards the goal along the path. However, if Pacman is at a square where a slide or ladder begins, he can instead choose to get transported to the end of the slide or ladder at the next time step. Pacman wants to minimize the number of time steps it takes to reach the goal.



For each of the following heuristics $h_i(s)$, where s is the input state, indicate whether A* graph search, A* tree search, both algorithms, or neither algorithm will be optimal under the heuristic h_i. The different scenarios below should be viewed independently. Let $(x(s), y(s))$ be Pacman's location in state s, (x_{goal}, y_{goal}) be the goal location, and $d(s)$ be the Manhattan distance between (x_{goal}, y_{goal}) and $(x(s), y(s))$.

(a) Let $B(s)$ and $E(s)$ be lists of squares which contains all the beginnings and ends of the ladders along the path from state s to the goal, respectively. $B(s)[i]$ and $E(s)[i]$ refer to the ith element of $B(s)$ and $E(s)$, respectively. Note that $|X|$ represents the number of elements in the list X.

(i) [2 pts] $h_1(s) = |B(s)|$

○ A* graph search ○ Both
○ A* tree search [✓ Neither]

This heuristic is inadmissible. Consider the case where Pacman is at a ladder that takes him directly to the goal, and there are any number of ladders beginning between.

(ii) [2 pts]

$$h_2(s) = \begin{cases} d(s) & \text{if } |B(s)| = 0 \\ \min_{0\le i<|B(s)|} \text{Manhattan}((x(s), y(s)), B(s)[i]) + \min_{0\le i<|B(s)|} \text{Manhattan}((x_{goal}, y_{goal}), E(s)[i]) & \text{otherwise} \end{cases}$$

○ A* graph search [✓ Both]
○ A* tree search ○ Neither

The heuristic is consistent. While there are ladders ahead, Pacman must move forward by at least the minimum Manhattan distance to a ladder beginning and at least the minimum Manhattan distance between ladder endings and the goal. If there are no more ladders, he must travel at least the Manhattan distance to the goal. The edge forward is not overestimated and there is no edge backward.

(b) [2 pts] Assume that all ladders go straight up (only their y coordinates change) and the length of the longest ladder is l_{max}. If no ladders exist, let $l_{max} = 1$.

$$h_3(s) = \frac{|y(s) - y_{goal}|}{l_{max}} + |x(s) - x_{goal}|$$

○ A* graph search [✓ Both]
○ A* tree search ○ Neither

This heuristic is consistent. The largest step we can take upward is the length l_max, and we must take at least the current distance sideways.

(c) [2 pts] Assume now that Pacman can choose to ignore any slide entrances that he encounters up to Z times and after that, he must transport down the slide if he enters a square containing a slide beginning. Let $C(s)$ and $F(s)$ be lists of squares which contain all the beginnings and ends of the slides along the path from state s to the goal, respectively. Let $l_{max} =$ the max ladder length or 1 if none exist, and z be the number of times that Pacman has already ignored a slide entrance. As a reminder, $d(s)$ is the Manhattan distance between (x_{goal}, y_{goal}) and $(x(s), y(s))$.

$$h_4(s) = \begin{cases} d(s)/l_{\max} & \text{if } |C(s)| = 0 \text{ or } z \le Z \\ d(s)/l_{\max} + \min_{0 \le i < |C(s)|} (\text{Manhattan}(C(s)[i], F(s)[i])) & \text{otherwise} \end{cases}$$

○ A* graph search ○ Both
○ A* tree search [✓ Neither]

This heuristic is inadmissible. While Pacman can avoid falling, he must travel at least his current Manhattan distance to the goal divided by his largest possible single step. Once he starts to fall down slides inevitably, the length of his journey may increase by the Manhattan distance of the shortest slides, but he might also encounter ladders, so optimality is not guaranteed for either algorithm.

Q3. CSPs: Potluck Pandemonium

The potluck is coming up and the staff haven't figured out what to bring yet! They've pooled their resources and determined that they can bring some subset of the following items.

1. Pho
2. Apricots
3. Frozen Yogurt
4. Fried Rice
5. Apple Pie
6. Animal Crackers

There are five people on the course staff: Taylor, Jonathan, Faraz, Brian, and Alvin. Each of them will only bring one item to the potluck.

i. If (F)araz brings the same item as someone else, it cannot be (B)rian.

ii. (A)lvin has pho-phobia so he won't bring Pho, but he'll be okay if someone else brings it.

iii. (B)rian is no longer allowed near a stove, so he can only bring items 2, 3, or 6.

iv. (F)araz literally can't even; he won't bring items 2, 4, or 6.

v. (J)onathan was busy, so he didn't see the last third of the list. Therefore, he will only bring item 1, 2, 3, or 4.

vi. (T)aylor will only bring an item that is before an item that (J)onathan brings.

vii. (T)aylor is allergic to animal crackers, so he won't bring item 6. (If someone else brings it, he'll just stay away from that table.)

viii. (F)araz and (J)onathan will only bring items that have the same first letter (e.g. Frozen Yogurt and Fried Rice).

ix. (B)rian will only bring an item that is after an item that (A)lvin brings on the list.

x. (J)onathan and (T)aylor want to be unique; they won't bring the same item as anyone else.

(a) Which of the listed constraints are unary constraints?

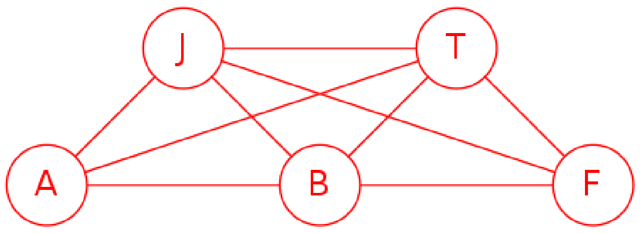
☐ i [✓ ii] [✓ iii] [✓ iv] [✓ v]

☐ vi [✓ vii] ☐ viii ☐ ix ☐ x

(b) Rewrite implicit constraint viii. as an explicit constraint.

(F, J) ∈ { (3, 4), (4, 3),
(2, 5), (5, 2),
(2, 6), (6, 2),
(5, 6), (6, 5),
(1, 1), (2, 2), (3, 3),
(4, 4), (5, 5), (6, 6)
}

(c) How many edges are there in the constraint graph for this CSP?
There are 9 edges in this constraint graph.



(d) The table below shows the variable domains after all unary constraints have been enforced.

A		2	3	4	5	6
B		2	3			6
F	1		3		5	
J	1	2	3	4		
T	1	2	3	4	5	

Following the MRV heuristic, which variable should we assign first? Break all ties alphabetically.

○ A [✓ B] ○ F ○ J ○ T

(e) To decouple this from the previous question, assume that we choose to assign (F)araz first. In this question, we will choose which value to assign to using the Least Constraining Value method.

To determine the number of remaining values, enforce arc consistency to prune the domains. Then, count the total number of possible assignments (**not** the total number of remaining values). It may help you to enforce arc consistency twice, once before assigning values to (F)araz, and then again after assigning a value.

(i) Assigning F = 1 results in 0 possible assignments.

Assigning F = 1 leaves no possible values in J's domain (due to constraint viii).

(ii) Assigning F = 3 results in 5 possible assignments.

Assigning F = 3 leaves J's domain as $\{4\}$. Enforcing arc consistency gives A = $\{2, 3, 5\}$, B = $\{6\}$, and T = $\{1, 2\}$. Therefore, the 5 possible assignments are (A, B, F, J, T) = (2, 6, 3, 4, 1), (3, 6, 3, 4, 1), (5, 6, 3, 4, 1), (3, 6, 3, 4, 2), (5, 6, 3, 4, 2).

(iii) Assigning F = 5 results in 3 possible assignments.

Assigning F = 5 leaves J's domain as $\{2\}$. Enforcing arc consistency gives A = $\{3, 4, 5\}$, B = $\{6\}$, and T = $\{1\}$. Therefore, the 3 possible assignments are (A, B, F, J, T) = (3, 6, 5, 2, 1), (4, 6, 5, 2, 1), (5, 6, 5, 2, 1).

(iv) Using the LCV method, which value should we assign to F? If there is a tie, choose the lower number.

○ 1 ○ 2 [✓ 3] ○ 4 ○ 5 ○ 6