CS 188 Fall 2026

Intro to Artificial Intelligence Exam Prep Discussion 1

Q1 *They See Me Rolling*

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.

Q1.1 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.

(a) 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.

- ○ $4mn$
- ○ $16mn$
- ○ $4k$
- ● $16k$
- ○ $12mn$
- ○ $48mn$
- ○ $12k$
- ○ $48k$

Solution: 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 $k * 4 * 4 = 16k$

(b) What is the maximum branching factor? The answer should be an integer.

9

Solution: 3 possible throttle inputs, and 3 possible steering inputs.

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

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

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

- ○ Yes
- ● No

Solution: 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.

Q1.2 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?

(a) Size of State Space:

● Increases ○ Stays the same ○ Decreases

Solution: 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 $k * 7 * 4 = 28k$

(b) Maximum Branching Factor:

○ Increases ● Stays the same ○ Decreases

Solution: Branching factor is independent of top speed.

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

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

■ Increases ■ Stays the same ■ Decreases

Solution: 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.

Q1.3 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?

(a) Size of State Space:

○ Increases ● Stays the same ○ Decreases

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

(b) Maximum Branching Factor:

○ Increases ● Stays the same ○ Decreases

Solution: Branching factor is independent of police presense.

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

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

■ Increases ■ Stays the same ■ Decreases

Solution: 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 *Heuristics*

Consider the A* searching process on the connected undirected graph, with starting node S and the goal node G. Suppose the cost for each connection edge is **always positive**. We define $h^*(X)$ as the shortest (optimal) distance to G from a node X.

Answer Questions 2.1 and 2.2. **Question 2.3 is optional.**

Q2.1 Suppose h is an **admissible** heuristic, and we conduct A* **tree search** using heuristic h' and finally find a solution. Let C be the cost of the found path (directed by h', defined in part (a)) from S to G.

Choose **one best** answer for each condition below.

1. If $h'(X) = \frac{1}{2}h(X)$ for all Node X, then

   ● $C = h^*(S)$ ○ $C > h^*(S)$ ○ $C \geq h^*(S)$

2. If $h'(X) = \frac{h(X)+h^*(X)}{2}$ for all Node X, then

   ● $C = h^*(S)$ ○ $C > h^*(S)$ ○ $C \geq h^*(S)$

3. If $h'(X) = h(X) + h^*(X)$ for all Node X, then

   ○ $C = h^*(S)$ ○ $C > h^*(S)$ ● $C \geq h^*(S)$

Q2.2 In which of the conditions above, h' is still **admissible** and for sure to dominate h? Check all that apply. Remember we say h_1 dominates h_2 when $h_1(X) \geq h_2(X)$ holds for all X.

☐ 1 ■ 2 ☐ 3

Q2.3 [Optional] Suppose h is an **admissible** heuristic, and we conduct A* **tree search** using heuristic h' and finally find a solution.

If $\varepsilon > 0$, and X_0 is a node in the graph, and h' is a heuristic such that

$$h'(X) = \begin{cases} h(X) & \text{if } X = X_0 \\ h(X) + \varepsilon & \text{otherwise} \end{cases}$$

- Alice claims h' can be inadmissible, and hence $C = h^*(S)$ does not always hold.
- Bob instead thinks the node expansion order directed by h' is the same as the heuristic h'', where

$$h''(X) = \begin{cases} h(X) - \varepsilon & \text{if } X = X_0 \\ h(X) & \text{otherwise} \end{cases}$$

Since h'' is admissible and will lead to $C = h^*(S)$, and so does h'. Hence, $C = h^*(S)$ always holds.

The two conclusions (underlined) apparently contradict with each other, and **only exactly one of them are correct and the other is wrong**. Choose the **best** explanation from below - which student's conclusion is wrong, and why are they wrong?

- ○ Alice's conclusion is wrong, because the heuristic h' is always admissible.
- ○ Alice's conclusion is wrong, because an inadmissible heuristics does not necessarily always lead to the failure of the optimality when conducting A* tree search.
- ○ Alice's conclusion is wrong, because of another reason that is not listed above.
- ○ Bob's conclusion is wrong, because the node visiting expansion ordering of h'' during searching might not be the same as h'.
- ○ Bob's conclusion is wrong, because the heuristic h'' might lead to an incomplete search, regardless of its optimally property.
- ● Bob's conclusion is wrong, because of another reason that is not listed above.

Solution: Choice 4 is incorrect, because the difference between h' and h'' is a constant. During searching, the choice of the expansion of the fringe will not be affected if all the nodes add the same constant to the heuristics.

Choice 5 is incorrect because there will never be an infinite loop if there are no cycle has negative COST sum (rather than HEURISTICS). If there is a cycle, such that its COST sum is positive, and all the nodes in the cycle have negative heuristics, when we do g+h, g is getting larger and larger, while h remains a not-that-large negative value. Soon, the search algorithm will be favoring other paths even if the h in there are not negative.

The true reason: h'' violate a property of admissible heuristic. Since h is admissible, we have $h(G) = 0$. If $X_0 = G$, we have a negative heuristic value at $h''(G)$, and it is no longer admissible. If $X_0 \neq G$, then it is indeed that the optimality holds - the only change is that more nodes will be likely to be expanded for h' and h'' compared to h.