

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**

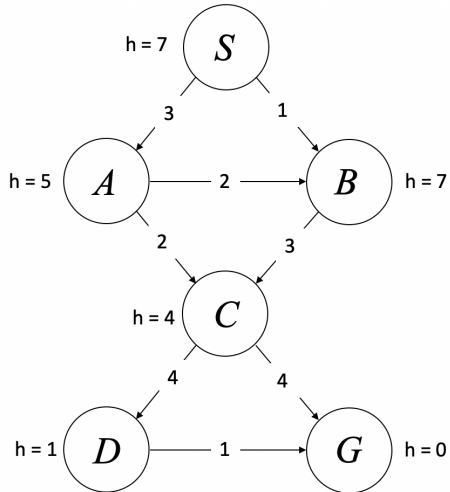
(iii) Number of nodes expanded with **Breadth** First Graph Search:

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 Algorithms Potpourri

- (a) We will investigate various search algorithms for the following graph. Edges are labeled with their costs, and heuristic values h for states are labeled next to the states. S is the start state, and G is the goal state. In all search algorithms, assume ties are broken in alphabetical order.



- (i) Select all boxes that describe the given heuristic values.

admissible consistent Neither

- (ii) Given the above heuristics, what is the order that the states are going to be expanded in, assuming we run A* graph search with the heuristic values provided.

Index	1	2	3	4	5	Not Expanded
S	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
A	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
B	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
C	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>
D	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
G	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>

- (iii) Assuming we run A* graph search with the heuristic values provided, what path is returned?

$S \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow G$
 $S \rightarrow A \rightarrow C \rightarrow G$
 $S \rightarrow A \rightarrow C \rightarrow D \rightarrow G$
 $S \rightarrow B \rightarrow C \rightarrow G$
 $S \rightarrow A \rightarrow C \rightarrow D \rightarrow G$
 $S \rightarrow A \rightarrow C \rightarrow D \rightarrow G$
 $S \rightarrow A \rightarrow B \rightarrow C \rightarrow G$
 None of the above

- (iv) Given the above heuristics, what is the order that the states are going to be expanded in, assuming we run greedy graph search with the heuristic values provided.

Index	1	2	3	4	5	Not Expanded
S	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
A	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
B	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
C	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
D	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
G	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

- (v) What path is returned by greedy graph search?

$S \rightarrow A \rightarrow B \rightarrow C \rightarrow D \rightarrow G$
 $S \rightarrow A \rightarrow C \rightarrow G$
 $S \rightarrow A \rightarrow C \rightarrow D \rightarrow G$
 $S \rightarrow A \rightarrow C \rightarrow D \rightarrow G$
 $S \rightarrow A \rightarrow C \rightarrow D \rightarrow G$
 None of the above

- (b) Consider a complete graph, K_n , the undirected graph with n vertices where all n vertices are connected (there is an edge between every pair of vertices), resulting in $\binom{n}{2}$ edges. Please select the maximum possible depth of the resulting tree when the following **graph** search algorithms are run (assume any possible start and goal vertices).

	1	$\lceil \frac{n}{2} \rceil$	$n - 1$	$\binom{n}{2}$	None of the above
BFS	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
DFS	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>

- (c) Given two admissible heuristics h_A and h_B .

- (i) Which of the following are guaranteed to also be admissible heuristics?

$h_A + h_B$
 $\frac{1}{2}(h_A)$
 $\frac{1}{2}(h_B)$
 $\frac{1}{2}(h_A + h_B)$
 $h_A * h_B$
 $max(h_A, h_B)$
 $min(h_A, h_B)$

- (ii) Consider performing A* **tree** search. Which is generally best to use if we want to expand the fewest number of nodes? **Note this was changed from graph to tree search during the exam**

$h_A + h_B$
 $\frac{1}{2}(h_A)$
 $\frac{1}{2}(h_B)$
 $\frac{1}{2}(h_A + h_B)$
 $h_A * h_B$
 $max(h_A, h_B)$
 $min(h_A, h_B)$

- (d) Consider performing tree search for some search graph. Let $depth(n)$ be the depth of search node n and $cost(n)$ be the total cost from the start state to node n . Let G_d be a goal node with minimum depth, and G_c be a goal node with minimum total cost.

- (i) For iterative deepening (where we repeatedly run DFS and increase the maximum depth allowed by 1), mark all conditions that are guaranteed to be true for every node n that could be expanded during the search, or mark "None of the above" if none of the conditions are guaranteed.

$cost(n) \leq cost(G_c)$
 $cost(n) \leq cost(G_d)$
 $depth(n) \leq depth(G_c)$
 $depth(n) \leq depth(G_d)$
 None of the above

When running iterative deepening we will explore all nodes of depth k , before we explore any nodes of depth $k + 1$. As a result will never explore any nodes that have depth greater than G_d because we would stop exploring once we reached G_d . This also means we would never explore any nodes with depth greater than G_c because G_c has to have depth greater than or equal to the minimum depth goal, G_d

- (ii) What is necessarily true regarding iterative deepening on any search tree?

Complete as opposed to DFS tree search
 Strictly faster than DFS tree search
 Strictly faster than BFS tree search
 More memory efficient than BFS tree search
 A type of stochastic local search
 None of the above