Q1. Word Ladder (30 points)

Wikipedia describes the rules for the word game Word Ladder as: The player is given a start word and an end word (of equal length). In order to win the game, the player must change the start word into the end word progressively, creating an existing word at each step. Each step consists of replacing a single letter. For example, the following is a solution to the Word Ladder puzzle between words "Cold" and "Warm":

$$\text{COLD} \rightarrow \text{CORD} \rightarrow \text{CARD} \rightarrow \text{WARD} \rightarrow \text{WARM}$$

1. (10 pts) Formulate the problem of solving word ladders as a search problem.

Solution. Since we're given both the starting word and ending word, this saves us from having to search the entire alphabet at each step. Thus, employing the rules that we choose letters alphabetically unless we have a preference and we check at each step if the change is in fact a word we can employ the following search:

1. Check each letter to see if changing to the goal letter results in a word.
2. If so, repeat for other letters until this fails.
3. If not, choose a letter alphabetically and attempt to change it to desired letter.
4. Repeat all steps until desired word is attained.

Therefore our nodes are each word, our fringe is the currently accessed words that we could change to from a certain step but haven't yet changed to, and the edges are the connections we have between these words. The steps in the fringe would look something like the following:

S	$COLD$
$S \rightarrow C$	
$S \rightarrow O$	
~~$S \rightarrow L$~~	$COLD \rightarrow CORD$
$S \rightarrow D$	
$S \rightarrow L \rightarrow C$	
~~$S \rightarrow L \rightarrow O$~~	$COLD \rightarrow CORD \rightarrow CARD$
$S \rightarrow L \rightarrow D$	
~~$S \rightarrow L \rightarrow O \rightarrow C$~~	$COLD \rightarrow CORD \rightarrow CARD \rightarrow WARD$
$S \rightarrow L \rightarrow O \rightarrow D$	
$S \rightarrow L \rightarrow O \rightarrow C \rightarrow D$	$COLD \rightarrow CORD \rightarrow CARD \rightarrow WARD \rightarrow WARM$

This of course follows the example above where we know a step-by-step solution to get to the goal. However, since this is one of the ideal routes, another fringe could include more steps and more letters. For instance, one could go from COLD to WIND via some number of steps, then eventually via some other number of steps, get to WARM. An example would be going from COLD → CORD → CORE → ... → WARM, provided each step is actually a word.

This problem becomes generalized when we consider the entire alphabet, words of any size, etc. One would need to extend each letter choice to include every letter of the alphabet and check at each step whether or not the letter change results in a word.

2. (10 pts) Consider a restriction where the words can only use the first seven letters (a-g). Draw the state space graph for converting "bee" to "dad".

Solution. I make the assumption that since "bed" has two edges, one could find their way back to bed via the path not taken. Hence in all except the goal "dad," we have double arrows to represent paths they could take. Unfortunately, this creates infinite loops in the search.

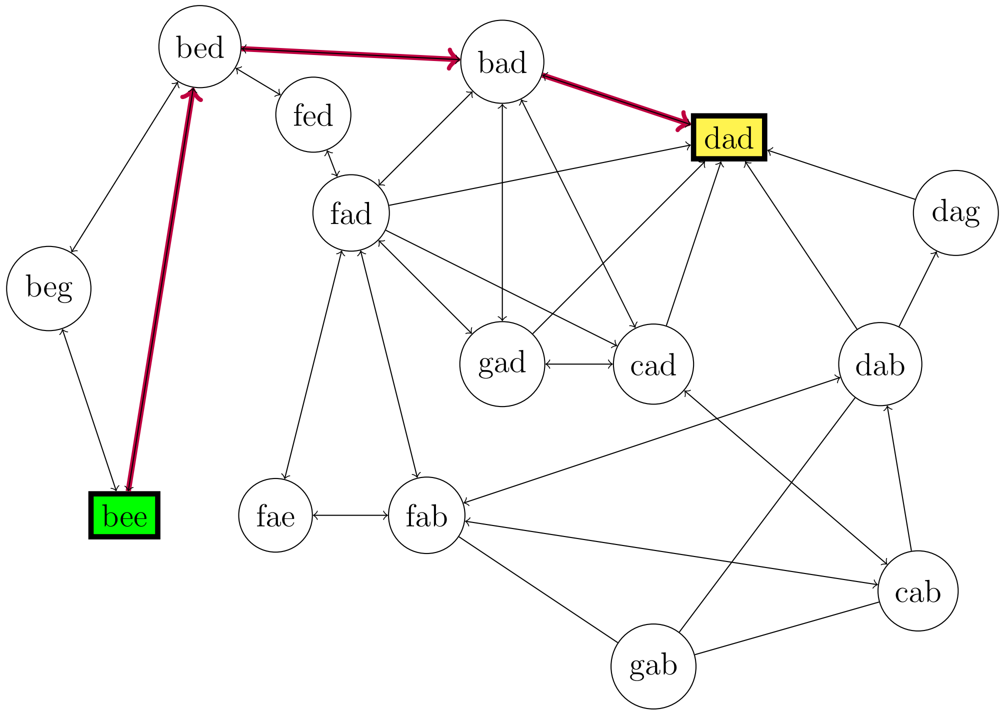


Figure 1: State space diagram for the word ladder bee → dad using only the first 7 letters. The ideal path is highlighted in violet. Since no rule is explicitly stated that a word can't be used over, endless loops are prevalent in the diagram (a rule that could be fixed easily in application).

3. (10 pts) (Tree Search) Which part of the search tree is expanded by DFS?

Solution. I suppose I'm to assume that we break ties alphabetically. Also, I'm going to try to ignore possible endless loops such as bad → gad → fad → bad. In this case, the tree would expand in the following order first:

$$\text{bee} \to \text{bed} \to \text{bad} \to \text{cad} \to \text{cab} \to \text{dab} \to \text{dad}$$

The steps in the fringe can be seen in Figure 2. It's worth noting that with these rules, some nodes will never be accessed, such as "beg" and "dag." One could argue for different rules for tie-breakers to include these words if necessary.

bee

~~*bee → bed*~~

bee → beg

~~*bee → bed → bad*~~

bee → bed → fed

~~*bee → bed → bad → cad*~~

bee → bed → bad → dad

bee → bed → bad → fad

bee → bed → bad → gad

~~*bee → bed → bad → cad → cab*~~

bee → bed → bad → cad → dad

bee → bed → bad → cad → fad

bee → bed → bad → cad → gad

~~*bee → bed → bad → cad → cab → dab*~~

bee → bed → bad → cad → cab → fab

bee → bed → bad → cad → cab → gab

✓~~*bee → bed → bad → cad → cab → dab → dad*~~

bee → bed → bad → cad → cab → dab → dag

Figure 2: Fringe steps for the case where the tie-breaker is the edges in alphabetical order. This neglects the ability to reuse a word, avoiding endless loops.

Q2. Graph Search (50 points)

Q2.1. Search Algorithms. (30 points) Consider the following graph. The start node is S and the goal node is G. For the following sub-questions, ties are broken in alphabetical order.

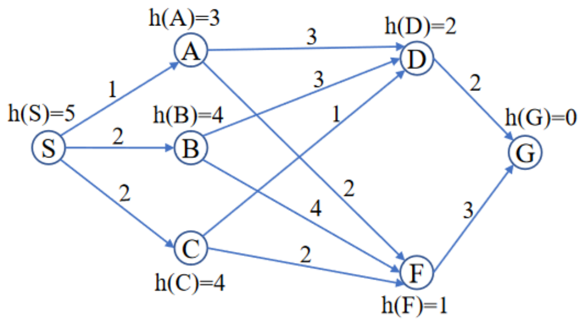


Figure 3: A given state space graph.

1. (10 pts) In what order are states expanded by UCS? What path would UCS return?

Solution. A UCS always chooses the current expanded path that has the cheapest cumulative cost in the current fringe. The first returned path will be $S \to C \to D \to G$. I think it'd be easiest to describe how this expands if I draw the search tree and comment on the order in the caption:

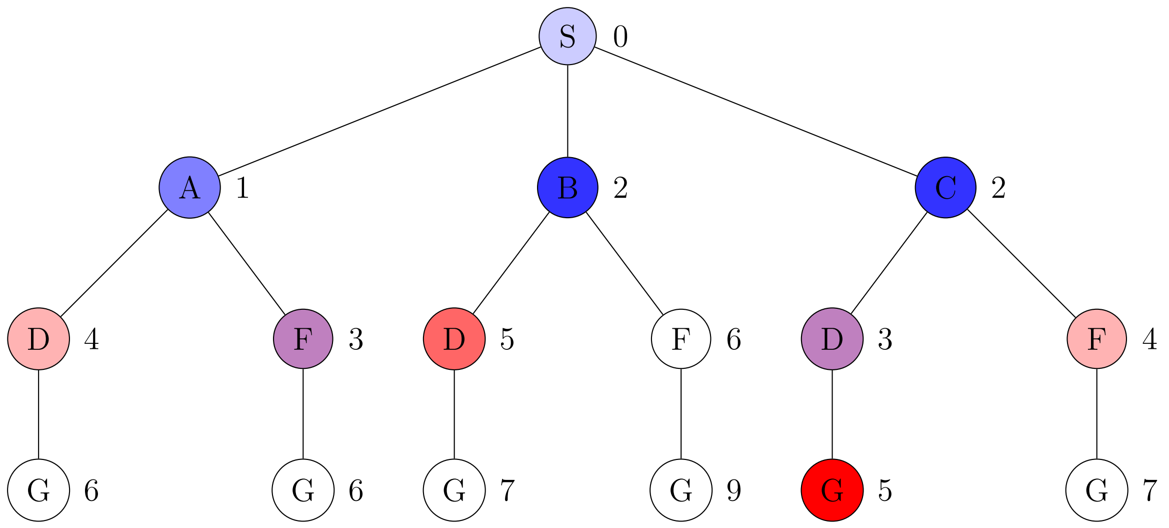


Figure 4: Search tree for a UCS given the state space in Figure 3. We first expand A since it has the cheapest cost of the three choices. Next, B and C are expanded in that order due to the alphabetical tiebreaker. Next, $A \to F$ then $C \to D$ are expanded, again due to the tiebreaker. Then we expand $A \to D$ then $C \to F$, followed by $B \to D$, and finally $C \to D \to G$.

2. (10 pts) In what order are states expanded by Greedy? What path would Greedy return?

Solution. A Greedy search looks for the node that *seems the closest* to the goal based on the heuristic alone, not taking into account accumulated cost so far. The smallest heuristic in step 1 is $h(A) = 3$ so we begin with $S \to A$. Next, the two competing heuristics are $h(D) = 2$ and $h(F) = 1$ so we now have $S \to A \to F$ (since $h(F) \leq h(B) = h(C)$. Finally $S \to A \to F \to G$ since $h(G) = 0$.

S

~~$S \to A$~~

$S \to B$

$S \to C$

$S \to A \to D$

~~$S \to A \to F$~~

~~$S \to A \to F \to G$~~

Figure 5: A fringe representing a Greedy expansion of the state space given in Figure 3. Greedy searches choose the edge in the fringe that is *closest to the goal* then chooses that as the next step. This fringe is identical to the fringe for Part 3 given that the current heuristic is not admissible.

3. (10 pts) In what order are states expanded by A^*? What path would A^* return?

Solution. A^* expands states based on a combination of the total cost and immediate cost to progress to the next node. That is, it uses an estimated heuristic $h(n)$ added with the actual cost to determine where to step next. At this second node, the actual cost is stored and the same process determines the next step until the cheapest *estimated* path is found. Thus, it's an informed search that makes a guess based on a heuristic as to what will *probably* be the next best course of action to quickly find the cheapest total cost to the goal.

At depth 1, A would be chosen because the estimated cost given the heuristics for A, B, and C are 4, 6, and 6 respectively. From A we move to F since the total accumulated cost and the estimated cost to move to F from A is $1 + 2 + 1 = 4 \leq 6$. Finally, from F we can only move to G or reconsider the initial moves from S to B or C. The step of moving from F to G, including previous cost, has an estimated cost of $3 + 3 + 0 = 6 \leq 6$, therefore it would be a tie. Since our ties are broken in alphabetical order, we choose the path

$$S \to A \to F \to G.$$

I'm a little unsure on this but I'm considering the alphabetical tiebreaker to follow dictionary rules as if the path is a string. Therefore $SAFG$ would alphabetically lead SB and SC.

Q2.2.Admissibility/Consistency. (20 points)

- (10 pts) Is the above heuristic function h admissible? Explain your answer.

Solution. The heuristic function $h(n)$ is *not* admissible. A heuristic function is admissible if at node n, $0 \leq h(n) \leq h^*(n)$ where $h^*(n)$ is the true cost to the nearest goal (from Lecture 4). We see that since $h(C) = 4 \geq h^*(C) = 3$ we can conclude that we don't have admissibility.

- (10 pts) For which TWO states (S, A, B, C, D, F, or G) could you change the heuristic values to make everything consistent? What range of values are possible to make this correction?

Solution. Clearly we would need to change the heuristic $h(C)$ as we discussed in the previous part. This would make the state space admissible (but not necessarily consistent). We could make $h(C) = a \in [0, 3)$ but ideally we'd make $h(C) = 3$ since that's the actual cost to the goal via D.

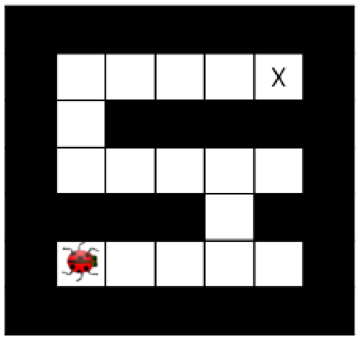
For consistency, we would also need to alter node F such that $h(F) = 2$. Therefore,

$$h(A) - h(F) = 1 \leq 2 = \mathit{cost}(A, F), \ \checkmark$$
$$h(B) - h(F) = 2 \leq 4 = \mathit{cost}(B, F), \ \checkmark$$
$$h(C) - h(F) = 1 \leq 2 = \mathit{cost}(C, F), \ \checkmark$$
$$h(F) - h(G) = 2 \leq 3 = \mathit{cost}(F, G). \ \checkmark$$

We could also let $h(F) = 3$ and still obtain consistency based on the definitions given during lecture (I didn't see anything about $h(n) - h(m)$ needing to be positive semi-definite or not).

Q3. Hive Mind: Lonely Bug. (20 points)

You control a single insect in a rectangular maze-like environment with dimensions M × N . Figure below is an example of such environment. The insect must reach a designated target location X, also known as the hive. There are no other insects moving around. At each time step, an insect can either (a) move into an adjacent square if that square is currently free, or (b) stay in its current location. Squares may be blocked by walls, but the map is known. Optimality is always in terms of time steps; all actions have cost 1 regardless of the number of insects moving or where they move.



1. (10 pts) Provide a minimal correct state space representation? What is the state-space size?

Solution. I think what you're asking for is an idea of what the minimum information we need for a state and what the bug can do, maybe some sort of pseudo-code? Here goes!

1. Check to see bugs position, orientation, and distance from the goal (based possibly on heuristics below). Suppose its state is stored as $((x, y), ['direction'], h)$.
2. Check to see which directions the bug can move, maybe by checking what walls are adjacent. These could be given like in the project as successors: $((x, y), ['direction'], c)$ where c is cost. This would have to incorporate the knowledge of walls in the maze which may be predetermined by the world state? I'm unsure how this is incorporated but obviously it can be done as Pacman deals with it.
3. We know there are various ways that h could be calculated but possibly some function here to calculate that (especially if we know where the goal is).

Therefore the state space representation includes just the most important information we need to get an idea of what the world state is.

If we let w be the number of "filled" grid points to represent walls and assume the bug can face 4 different directions then

$M \times N - w$	Bug positions given walls, w
up, down, left, right $= 4$	Bug facing directions
$M \times N - w - 1$	Hive position given walls, w
$(M \times N - w) \times 4 \times (M \times N - w - 1)$	State-space size

2. (10 pts) Provide two heuristics which are admissible.

Solution. Since this is an $M \times N$ grid, one heuristic that will *always* be admissible will be Euclidean distance to the goal. This can be proven using the triangle inequality since for any three sides of a triangle A, B, and C, $|C| \leq |A| + |B|$. Thus if we define our Euclidean distance at any time to be C, it will always be less than or equal to any path taken to reach the goal.

Another heuristic would be the total distance given only x and y translations ignoring walls. Similar to the triangle inequality from the first heuristic, this distance will always be equal to or shorter than the path distance obeying the walls. TL;DR - the Manhattan distance.

These heuristics are admissible but not necessarily consistent since consistency $\Rightarrow$ admissibility but admissibility $\not\Rightarrow$ consistency.