CS 188
Summer 2023

Discussion 1C Solutions

Q1. Searching with 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 (a), (b) and (c). You may want to solve Questions (a) and (b) at the same time.

(a) 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

(i) 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 ○ $C = h^*(S)$ ○ $C > h^*(S)$ ● $C \geq h^*(S)$
4. If we define the set $K(X)$ for a node X as all its neighbor nodes Y satisfying $h^*(X) > h^*(Y)$, and the following always holds

$$h'(X) \leq \begin{cases} \min_{Y \in K(X)} h'(Y) - h(Y) + h(X) & \text{if } K(X) \neq \emptyset \\ h(X) & \text{if } K(X) = \emptyset \end{cases}$$

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

5. If K is the same as above, we have

$$h'(X) = \begin{cases} \min_{Y \in K(X)} h(Y) + \mathit{cost}(X, Y) & \text{if } K(X) \neq \emptyset \\ h(X) & \text{if } K(X) = \emptyset \end{cases}$$

where $\mathit{cost}(X, Y)$ is the cost of the edge connecting X and Y,
then, ● $C = h^*(S)$ ○ $C > h^*(S)$ ○ $C \geq h^*(S)$

6. If $h'(X) = \min_{Y \in K(X)+\{X\}} h(Y)$ (K is the same as above), ● $C = h^*(S)$ ○ $C > h^*(S)$ ○ $C \geq h^*(S)$

(ii) 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 □ 4 □ 5 □ 6

(b) Suppose h is a **consistent** heuristic, and we conduct A* **graph search** using heuristic h' and finally find a solution.

(i) Answer exactly the same questions for each conditions in Question (a)(i).

1. ● $C = h^*(S)$ ○ $C > h^*(S)$ ○ $C \geq h^*(S)$
2. ● $C = h^*(S)$ ○ $C > h^*(S)$ ○ $C \geq h^*(S)$
3. ○ $C = h^*(S)$ ○ $C > h^*(S)$ ● $C \geq h^*(S)$
4. ● $C = h^*(S)$ ○ $C > h^*(S)$ ○ $C \geq h^*(S)$
5. ● $C = h^*(S)$ ○ $C > h^*(S)$ ○ $C \geq h^*(S)$
6. ○ $C = h^*(S)$ ○ $C > h^*(S)$ ● $C \geq h^*(S)$

(ii) In which of the conditions above, h' is still **consistent** and for sure to dominate h? Check all that apply.
□ 1 ■ 2 □ 3 □ 4 ■ 5 □ 6

Grading for Bubbles: 0.5 pts for a1 a2 a3 a6 b1 b2. 1 pts for a4 a5 b3 b4 b5 b6.

Explanations:

All the $C > h^*(S)$ can be ruled out by this counter example: there exists only one path from S to G.

Now for any $C = h^*(S)$ we shall provide a proof. For any $C \geq h^*(S)$ we shall provide a counter example.

a3b3 - Counter example: SAG fully connected. cost: SG=10, SA=1, AG=7. h*: S=8, A=7, G=0. h: S=8, A=7, G=0. h': S=16, A=14, G=0.

a4 - Proof: via induction. We can have an ordering of the nodes $\{X_j\}_{j=1}^n$ such that $h^*(X_i) \geq h^*(X_j)$ if $i < j$. Note any $X_k \in K(X_j)$ has $k > j$.

X_n is G, and has $h'(X_n) \leq h(X_n)$.

Now for j, suppose $h'(X_k) \leq h(X_k)$ for any $k > j$ holds, we can have $h'(X_j) \leq h'(X_k) - h(X_k) + h(X_j) \leq h(X_j)$ ($K(X_j) = \emptyset$ also get the result).

b4 - Proof: from a4 we already know that h' is admissible.

Now for each edge XY, suppose $h^*(X) \geq h^*(Y)$, we always have $h'(X) \leq h'(Y) - h(Y) + h(X)$, which means $h'(X) - h'(Y) \leq h(X) - h(Y) \leq cost(X, Y)$, which means we always underestimate the cost of each edge **from the potential optimal path direction**. Note h' is not necessarily to be consistent ($h'(Y) - h'(X)$ might be very large, e.g. you can arbitrarily modify h'(S) to be super small), but it always comes with optimality.

a5 - Proof: the empty K path: $h'(X) \leq h(X) \leq h^*(X)$. the non-empty K path: there always exists a $Y_0 \in K(X)$ such that Y_0 is on the optimal path from X to G. We know $cost(X, Y_0) = h^*(X) - h^*(Y_0)$, so we have $h'(X) \leq h(Y_0) + cost(X, Y_0) \leq h^*(Y_0) + cost(X, Y_0) = h^*(X)$.

b5 - Proof:

First we prove $h'(X) \geq h(X)$. For any edge XY, we have $h(X) - h(Y) \leq cost(X, Y)$. So we can have $h(Y) + cost(X, Y) \geq h(X)$ holds for any edge, and hence we get the dominace of h' over h. Note this holds only for consistent h.

We then have $h'(X) - h'(Y) \leq h(Y) + cost(X, Y) - h'(Y) \leq cost(X, Y)$. So we get the consistency of h'.

Extension Conclusion 1: If we change K(X) into {all neighbouring nodes of X} + {X}, h' did not change.

Extension Conclusion 2: h' dominates h, which is a better heuristics. This (looking one step ahead with h') is equivalent to looking two steps ahead in the A* search with h (while the vanilla A* search is just looking one step ahead with h).

a6 - Proof: $h'(X) \leq h(X) \leq h^*(X)$.

b6 - counter example: SAB fully connected, BG connected. cost: SA=8, AB=1, SB=10, BG=30. h*: A=31, B=30 G=0. h=h*. h': A=30, B=0, C=0.

Q2. Iterative Deepening Search

Pacman is performing search in a maze again! The search graph has a branching factor of b, a solution of depth d, a maximum depth of m, and edge costs that may not be integers. Although he knows breadth first search returns the solution with the smallest depth, it takes up too much space, so he decides to try using iterative deepening. As a reminder, in standard depth-first iterative deepening we start by performing a depth first search terminated at a maximum depth of one. If no solution is found, we start over and perform a depth first search to depth two and so on. This way we obtain the shallowest solution, but use only O(bd) space.

But Pacman decides to use a variant of iterative deepening called **iterative deepening A***, where instead of limiting the depth-first search by depth as in standard iterative deepening search, we can limit the depth-first search by the f value as defined in A* search. As a reminder $f[node] = g[node] + h[node]$ where $g[node]$ is the cost of the path from the start state and $h[node]$ is a heuristic value estimating the cost to the closest goal state.

In this question, all searches are tree searches and **not** graph searches.

(a) Complete the pseudocode outlining how to perform iterative deepening A* by choosing the option from the next page that fills in each of these blanks. Iterative deepening A* should return the solution with the lowest cost when given a consistent heuristic. Note that cutoff is a boolean and new-limit is a number.

```
function ITERATIVE-DEEPENING-TREE-SEARCH(problem)
    start-node ← MAKE-NODE(INITIAL-STATE[problem])
    limit ← f[start-node]
    loop
        fringe ← MAKE-STACK(start-node)
        new-limit ← [ (i) ]
        cutoff ← [ (ii) ]
        while fringe is not empty do
            node ← REMOVE-FRONT(fringe)
            if GOAL-TEST(problem, STATE[node]) then
                return node
            end if
            for child-node in EXPAND(STATE[node], problem) do
                if f[child-node] ≤ limit then
                    fringe ← INSERT(child-node, fringe)
                    new-limit ← [ (iii) ]
                    cutoff ← [ (iv) ]
                else
                    new-limit ← [ (v) ]
                    cutoff ← [ (vi) ]
                end if
            end for
        end while
        if not cutoff then
            return failure
        end if
        limit ← [ (vii) ]
    end loop
end function
```

A_1	$-\infty$	**A_2**	0	**A_3**	∞	**A_4**	*limit*
B_1	True	**B_2**	False	**B_3**	*cutoff*	**B_4**	not *cutoff*
C_1	*new-limit*	**C_2**	*new-limit* + 1	**C_3**	*new-limit* + f[*node*]	**C_4**	*new-limit* + f[*child-node*]
C_5	MIN(*new-limit*, f[*node*])	**C_6**	MIN(*new-limit*, f[*child-node*])	**C_7**	MAX(*new-limit*, f[*node*])	**C_8**	MAX(*new-limit*, f[*child-node*])

(i)	○ **A_1**	○ **A_2**	● **A_3**	○ **A_4**
(ii)	○ **B_1**	● **B_2**	○ **B_3**	○ **B_4**
(iii)	● **C_1**	○ **C_2**	○ **C_3**	○ **C_4**
	○ **C_5**	○ **C_6**	● **C_7**	● **C_8**
(iv)	○ **B_1**	○ **B_2**	● **B_3**	○ **B_4**
(v)	○ **C_1**	○ **C_2**	○ **C_3**	○ **C_4**
	○ **C_5**	● **C_6**	○ **C_7**	○ **C_8**
(vi)	● **B_1**	○ **B_2**	○ **B_3**	○ **B_4**
(vii)	● **C_1**	○ **C_2**	○ **C_3**	○ **C_4**
	○ **C_5**	○ **C_6**	○ **C_7**	○ **C_8**

The cutoff variable keeps track of whether there are items that aren't being explored because of the limit. If cutoff is false and the algorithm has exited the while, no nodes were cutoff (not added to the fringe because of the limit). This scenario suggests that there is no solution.

In order to ensure that iterative deepening A* obtains the lowest cost solution efficiently, we want to increase the limit as much as we can while guaranteeing optimality. Setting new-limit to the smallest f cost of nodes that were cutoff achieves this. When nodes aren't cutoff (part iii), the new-limit should not change. Hence C1, C7, C8, or a combination of the three were accepted as answers.

(b) Assuming there are no ties in f value between nodes, which of the following statements about the number of nodes that iterative deepening A* expands is True? If the same node is expanded multiple times, count all of the times that it is expanded. If none of the options are correct, mark None of the above.

- ● The number of times that iterative deepening A* expands a node is greater than or equal to the number of times A* will expand a node.
- ○ The number of times that iterative deepening A* expands a node is less than or equal to the number of times A* will expand a node.
- ○ We don't know if the number of times iterative deepening A* expands a node is more or less than the number of times A* will expand a node.
- ○ None of the above

Iterative deepening A* runs depth first search multiples at different limit values. This causes iterative deepening A* to expand certain nodes multiple times.