CS440: Intro to Artificial Intelligence
Homework #2

Jonathan Friedman(jif24) and David Schwab (dbs12)

30. September 2018

Problem 1. Trace the following search algorithms:

Breadth First Search:

1. $q = [A]$
2. A, $q = [B, C, D]$
3. B, $q = [C, D, E, F]$
4. C, $q = [D, E, F]$
5. D, $q = [E, F]$
6. E, $q = [F]$
7. F, $q = []$

Depth First Search: *(We are assuming duplicate nodes we have already visited will not be added to our queue)*

1. $q = [A]$
2. A, $q = [B, D]$
3. B, $q = [C, F]$
4. C, $q = [D, E, F]$
5. D, $q = [E]$
6.. E, $q = [F]$
7. F, $q = []$

Uniform Cost Search:

1. $q = [A(0)]$
2. A, $q = [B(3), D(4), C(5)]$
3. B, $q = [C(4), D(4), F(9)]$
4. C, $q = [D(4), E(6), F(9)]$
5. D, $q = [E(6), F(9)]$ *(backtrack to D)*
5. E, $q = [F(8)]$
6. F, $q = []$

A*:

1. $q = [A(0)]$
2. A, $q = [B(3+4), C(5+3), D(4+4)]$
3. B, $q = [C(4+3), D(4+4), F(9+0)]$
4. C, $q = [E(6+1), D(4+4), F(9+0)]$
5. E, $q = [D(4+4), F(8+0)]$
6. D, $q = [F(8+0)]$

F, $q = []$

Problem 2. Provide an estimate of how many nodes will be expanded in the following scenarios:

A. Cycle Graph:

1.*Breadth First Search:*

a. A BFS tree search will traverse $O(2^k)$ nodes in order to find its goal. Since it doesn't remember previous states, it will be adding its two neighbors regardless of whether or not they've been visited.

b. A BFS graph search on a cycle graph, where nodes are not visited twice, will only need to expand (n) nodes to discovers its goal.

2. *Depth First Search:*

a. A DFS tree search will need to expand approximately $O(n)$ nodes to reach its goal. It will either start looking in the right direction and expand only k nodes or it will go the wrong direction and need to expand $n - k$ nodes.

b. A DFS graph search will be the same situation - $O(n)$ nodes expanded in the worst case. Since DFS dives deep down a child branch, and our graph is one large cycle, remembering past states doesn't provide any improvement to the DFS algorithm.

B. Infinite Diamond Strip:

1.*Breadth First Search:*

a. A tree search implementation of BFS will expand $O(4^{2m})$ nodes in the worst case. $2m$ corresponds to the distance we need to travel to get to our goal. Since we don't remember visited neighbors, every node will result in an additional 4 nodes being added to our queue, $2m$ times.

b. A graph search implementation of BFS on an IDS will perform much better, since it's only adding nodes we haven't visited yet. Consequently, this will result in the expansion of $O(4m)$ nodes (expanding $2m$ nodes in either direction).

2.*Depth First Search:*

a. A tree search DFS will either find the goal in $O(2m)$ nodes (assuming the goal is on the bottom left of where we started. Otherwise it will be infinite, as DFS will have an infinite amount of children nodes to explore in one direction.

b. Since DFS will be exploring an infinite sequence of children nodes in one direction, a graph search approach in which we remember repeated states won't experience any improvement. It will still either find the goal node in $O(2m)$ nodes, or continue searching infinitely.

Problem 3. Prove that consistent heuritstics are always admissible. For a heuristic function $h(x_g) = 0$, prove that $h(x)C^*x$ where C^*x is the optimal cost from x to x_G.

Solution: We can prove this using induction.

Base Case: Let's define $k(n)$ to be the cost of the cheapest path from some node n to the goal (ie; an admissible function). Assume we are at our goal node, x_G. Then $h(x_G) = 0 \leq k(x_G)$. In this case, our consistent function is also admissible.

Inductive Step: Consider some node n that is k steps away from our goal. It must have some successor n' generated by action a that would put us on the optimal path to our goal. We can then say that n' must be $k-1$ steps away from our goal. This implies:

$$h(n) \leq c(n, an') + h(n')$$

By our inductive hypothesis, $h(n\prime) \leq k(n\prime)$, therefore:

$$h(n) \leq c(n, a, n') + k(n') = k(n)$$

which is a result of the fact that n' is on an optimal path to our goal x_G. This concludes our inductive step, proving that any consistent heuristic is also admissible.

4. Which of the following are admissible, given admissible heuristics h_1, h_2? Which of the following are consistent given consistent heuristics h_1, h_2? For your justification, prove starting with definitions of admissible and consistent heuristics.

(i)$h_{min}(n) = min\{h_1(n), h_2(n)\}$

Admissible: *True.* Since $h_1(n), h_2(n) \leq h^*(n)$, it follows that $min\{h_1(n), h_2(n)\}$ will be less than $h^*(n)$ as well.

Consistent: *True.* Since $h_1, h_2 \leq c(n, a, n') + h(n')$, it also follows that the minimum of the two heuristics will be consistent, as it will satisfy

the necessary triangle inequality.

(ii)$h_{max}(n) = max\{h_1(n), h_2(n)\}$

Admissible: *True.* Since $h_1(n), h_2(n) \leq h^*(n)$, $max\{h_1(n), h_2(n)\}$ will still be less than $h^*(n)$ as well.

Consistent: *True.* Since $h_1, h_2 \leq c(n, a, n') + h(n')$, it also follows that the max of the two heuristics will be consistent, as it will satisfy the necessary triangle inequality.

(iii) $h_{lin}(n) = wh_1(n) + (1 - w)h_2(n), 0 \leq w \leq 1$

Admissible: *True.* Since $h_1(n), h_2(n) \leq h^*(n)$, and there is no combination of the two using weights $0 \leq w \leq 1$ and $1 - w$ that will result in an $h_3(n)$ with values greater than either h_1 or h_2, this is an admissible function.

Consistent: *True.* By the same logic for the admissibility of h_3, since there is no combination of h_1 and h_2 (given some weights $0 \leq w, (1 - w) \leq 1$) that can result in an h_3 bigger than h_1 or h_2, and snce those are both consistent heuristics, h_3 is at most equal to the greater of the two, so it follows that the the heuristic h_3 is also consistent.

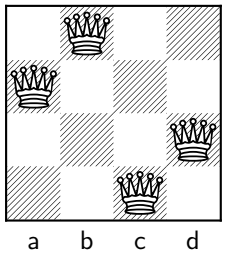
5. Consider an informed best-first search algorithm with an evaluation function $f(n) = (2 - w)g(n) + wh(n), 0 \leq w \leq 2$. For what values of w will this be optimal? What kind of search will it perform at $w = 0, 1, 2$?

   Solution: When $w = 0$, $f(n) = 2g(n)$, meaning we don't take the heuristic values into consideration, and our search algorithm will default to a uniform-cost search in which we only consider the $g(n)$ values (at twice their normal values). At $w = 1$, $f(n) = g(n) + h(n)$. In other words, the algorithm performs like a regular A^* algorithm, and at $w = 2$, $f(n) = 2h(n)$, and we don't consider $g(n)$ values at all, so our algorithm performs in a greedy best-first search manner.

   In every case where $w > 1$, our algorithm will behave in an increasingly greedy fashion. This means we will lose our guarantee of optimality. In order to guarantee optimality we must use either uniform-cost search, A^*, or some combination of both. This will only happen when $w \leq 1$.

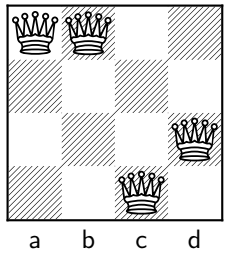
6. Apply hill-climbing to the 4-Queens problem until you can no longer make any moves. The objective function is equal to the number of attacking pairs (without considering blocking). Each queen is only allowed to move in the column she belongs.

Iteration 0: Move Queen a3



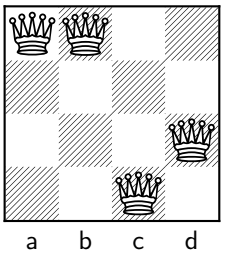
Objective Function Values:
[3|Q|4|3]
[Q|3|3|3]
[3|4|4|Q]
[3|4|Q|4]

Iteration 1: Move Queen b4



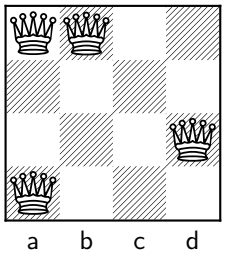
Objective Function Values:
[Q|Q|3|3]
[3|2|3|3]
[3|2|4|Q]
[3|2|Q|4]

Iteration 2: Move Queen c1



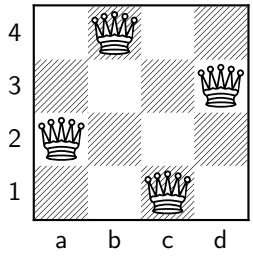
Objective Function Values:
[Q|4|3|3]
[3|Q|3|3]
[4|3|3|Q]
[2|2|Q|3]

Iteration 3: Move Queen d2



Objective Function Values:
$[Q|4|4|4]$
$[4|Q|4|3]$
$[4|4|3|Q]$
$[Q|4|3|4]$

Carrying on with these tie-brekaing methods we end up with a far greater number of iterations than necessary. In specific, not considering blocked pairs throws away valuable information, leading to a slower convergence to a global minimum. The desired final state looks as follows:



7. Consider the 8-puzzle problem and the heuristics h_1 and h_2 from class. Recall that h_1 counts the number of misplaced tiles to their target distance and h_2 the Manhattan distance. Prove that both heuristics are admissible.

   Solution:
   We can easily show that h_2 is admissible by proving its consistency. To prove its consistency we can rely on the function $a^2+b^2 = c^2$. In a Manhattan distance heuristic, a^2 can be thought of as the equivalent of $c(n, a, n')$, b^2 as $h(n')$ and c^2 as $h(n)$. Because it is consistent, it is always admissible.

   To prove that h_1 is admissible, consider that every tile out of place must be moved at least once in order to reach our goal state. Therefore, it is impossible to come up with a scenario in which $h_1(n) > h^*(n)$