CSE 486/586, Assignment 2

Joshua Crail (Crailjc)

Due: Tuesday, September 28, 2021, by 11:59 pm.

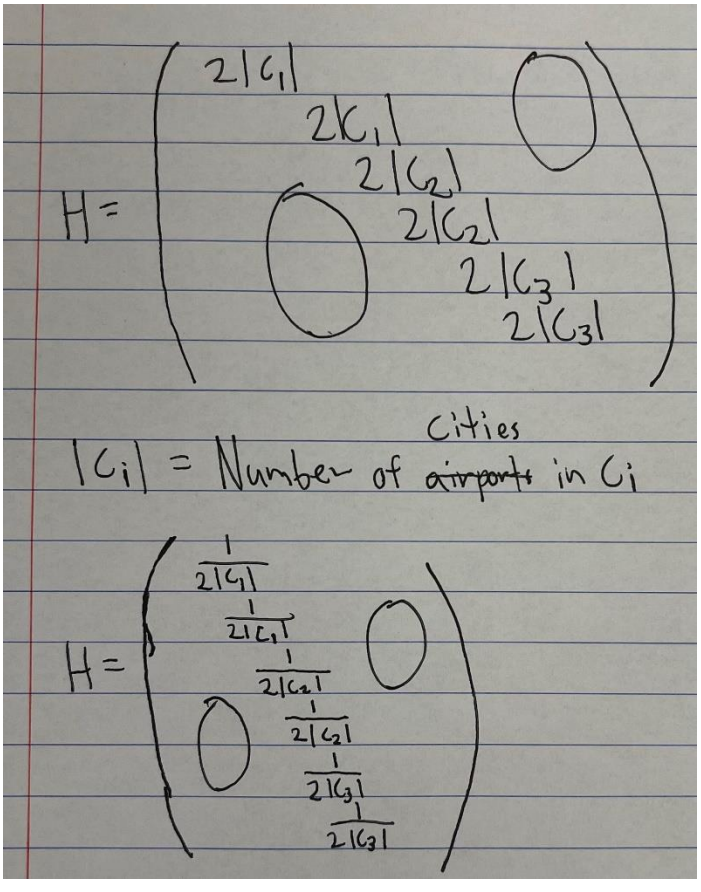
Note 1: The total mark for this assignment is 30. You should *NOT* directly copy anything from slides or other resources. You may get the ideas from slides but what you submit *must be in your own words*. Any help must be acknowledged.

Note 2: Question 8 is only for CSE 586 students (0 point for correct answer, and -5 for wrong answer)

1. Suppose we want to construct three new airports in a given country. The objective is to minimize the distance from each city to its nearest airport. Describe the gradient descent and explain how this technique can help us to solve this problem. Write your solution mathematically as we discussed in class. (8 points)

   We can denote the cities with the nearest airport i as C_i. Then we can make (x_i,y_i) be coordinated of airport i and (x_c,y_c) be the coordinated of the city. We will take an initial value and then use it to continuously get a new value which can be used to better estimate the position of the new airports. Where the equation would be $x \leftarrow x - H_f^{-1}(x)\nabla f(x)$. Where $H_f^{-1}(x)$ is the inverse hessian matrix for our airports where the only nonzero values are on the diagonal, those values will be $\frac{1}{2|c_i|}$. $\nabla f(x)$ is where the gradient vector that is the partial derivative of dx_n, then dy_n for all three airports. Where the final answer will be a minimized x value showing the best spot for each of the three airports.

   Hessian Matrix shown below:



2. Implement the gradient descent for a given scenario. (Optional group problem, 1 EXTRA credit, Due: November 10, 2021, show me your code in person).

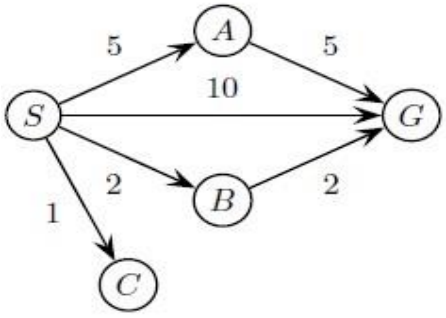
3. Design an AI that plays chess using the minimax algorithm and alpha-beta pruning. (Optional group problem, 2 EXTRA credits, Due: November 10, 2021, show me your code in person).

4. a) Give a set of broad conditions under which A* search reduces to BFS. (2 points)
   When there is a cost constant for the heuristic value and all edge costs are the same.

   b) Does A* always return an optimal solution? Explain. (2 points)
   No, because if our h value is not admissible then A* will not be optimal.

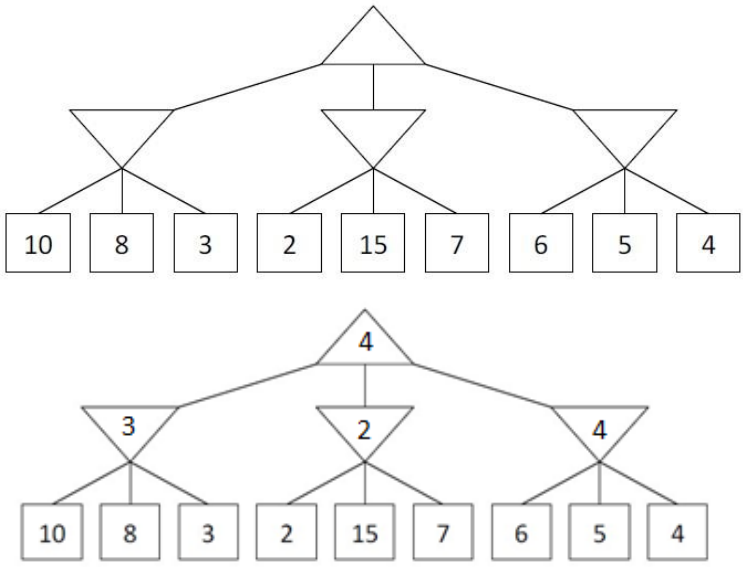
5. a) When would DFS be a better choice than A* search? (3 points)
   When the space complexity is an issue DFS will be linear while A* will not be (exponential). Also, for example if the search required had to do through many depths, then depth first would be better as A* will be going based off of the f-value meaning it will take longer for it to go deeper. In this case when the search has a very large depth DFS would be better suited.

   b) Which path will A* return for the following search problem? (3 points)
   S, B, G Cost: 4



Node	h
S	4
A	3
B	2
C	100
G	0

6. a) Consider the zero-sum game tree shown below. Triangles that point up, such as at the top node (root), represent choices for the maximizing player; triangles that point down represent choices for the minimizing player. Assuming both players act optimally, fill in the minimax value of each node. (4 points)

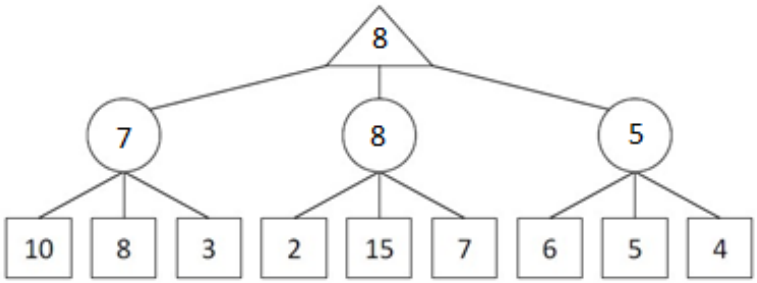


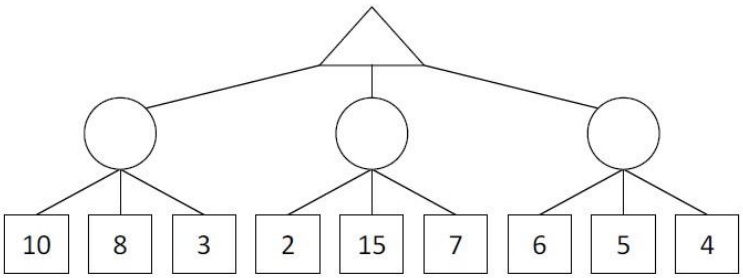
b) Which nodes can be pruned from the game tree above through alpha-beta pruning? If no nodes can be pruned, explain why not. Assume the search goes from left to right; when choosing which child to visit first, choose the left-most unvisited child. (2 points)

Nodes 15,7 can be pruned. 3 is chosen for the left minimum meaning that since the first node looked at for the minimum in the middle is 2 that means that the value chosen will be two or less than two as such 15,7 will be pruned because of this. No nodes on the right most minimum will be pruned since they are all greater than 3.

7. a) Again, consider the same zero-sum game tree, except that now, instead of a minimizing player, we have a chance node that will select one of the three values uniformly at random. Fill in the expectiminimax value of each node. The game tree is redrawn below for your convenience. (4 points)

Left expectiminimax: 10(1/3) + 8(1/3) + 3(1/3) = 7
Middle expectiminimax: 2(1/3) + 15(1/3) + 7(1/3) = 8
Right expectiminimax: 6(1/3) + 5(1/3) + 4(1/3) = 5





b) Which nodes can be pruned from the game tree above through alpha-beta pruning? If no nodes can be pruned, explain why not. (2 points)
No since the nodes are being chosen uniformly at random as such each node it contributing to an average for the left, right, and middle as such if anything was to be pruned the average would be changed meaning that averages changes meaning the result could change. Alpha beta pruning is only meant to speed up the game and not change the final value of the game.

8. The evaluation function of a well-known search algorithm is
$f(n) = (2 - w)g(n) + wh(n)$.

$$f(n) = (2 - w)(g(n) + \frac{w}{2 - w}h(n))$$

For what values of *w* is this search algorithm optimal, assuming that *h* is admissible? What kind of search does this perform for *w* = 0 and *w* = 1? (Only for CSE 586 students)
Since we know h is admissible then the w value should never give a value that will be an overestimate as such w must always be less than or equal to one. If you factor out the (2-w) and just look at $\frac{w}{2-w}$ you can see that when w is less than or equal to one it will always give a value that will not be greater than 1 as such h will not increase only decrease or stay the same meaning that our estimated cost will never go over 1 * h(n) meaning that h will stay admissible making the search optimal.

When w = 0 the search is Uniform cost search (UCS) because there is only the cost of the path from the nodes just like how UCS is g(n). For w = 1 the search is A* since the algorithm for it will be 1g(n) + 1h(n) or g(n) + h(n) just like A*