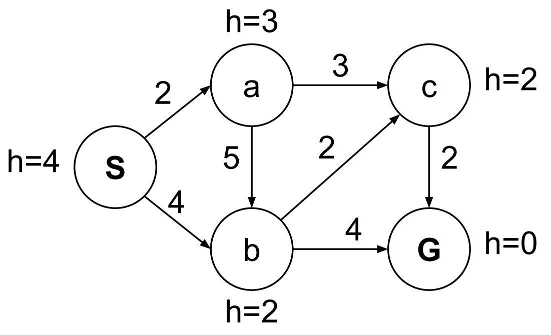
Final Exam Practice

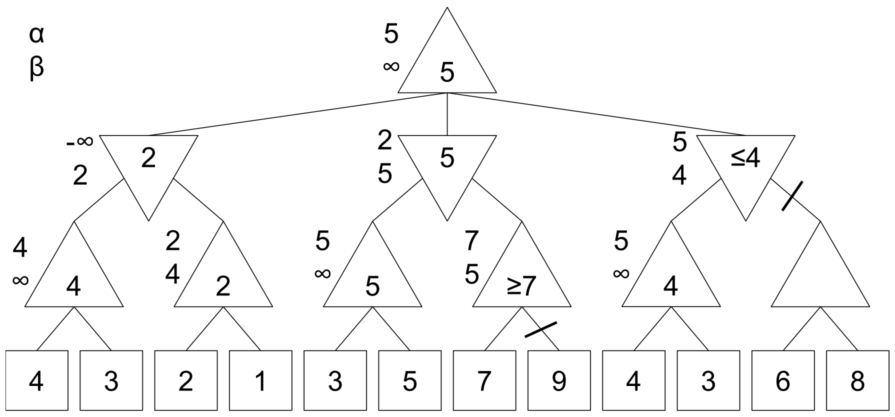
1. Simulate A* graph search on the following problem, starting at **S** and ending at **G**. What nodes are expanded in what order? Is the heuristic **h** consistent?



Simulation:
Expand **S** (f=4). Fringe {a (f=5), b (f=6)}
Expand a (f=5). Fringe {b (f=6), b (f=9), c (f=7)}
Expand b (f=6). Fringe {b (f=9), c (f=7), **G** (f=8)}
Expand c (f=7). Fringe {b (f=9), **G** (f=8), **G** (f=7)}
Expand **G** (f=7). Done.

The heuristic **h** is consistent. Check for each arc x→y that h(x) - h(y) ≤ c(x→y). (Proof omitted)

2. Simulate alpha-beta pruning on the following game tree. Assume the nodes are evaluated from left to right. Mark all branches that are pruned, and show the computed value of each node that wasn't pruned. If only an upper or lower bound is known, write ≤x or ≥x, where x is the value of that bound. Next to each non-terminal node, write the final values of alpha and beta used in evaluating that node.



3. Why split a dataset into training, validation, and test subsets? What is each one used for?

The training subset is used to update the parameters of the model. The validation subset is used to test for overfitting and optimize hyperparameters. The test set is for final model evaluation. The splits are made to prevent overfitting and result engineering.

4. Describe what a BatchNorm layer does. Why is this useful?

A BatchNorm layer calculates the mean and standard deviation of a batch of inputs, and renormalizes them so that they have a new mean and standard deviation (these new ones are learnable parameters). This helps deep networks prevent inputs to the activation functions from being stuck in "dead" regions with vanishing gradient.

5. How many parameters does a neural network have that takes **5** inputs, has a hidden layer of size **7**, and an output layer with **4** output neurons? Remember that each neuron also has a bias.

(**5** inputs)*(**7** weights) **+** (**7** biases) **+** (**7** inputs)*(**4** weights) **+** (**4** biases) **= 35 + 7 + 28 + 4 = 74**

6. You are playing the following game on a board with six squares arranged like so:

0	**3**	**2**	**4**	**7**	**5**

Your piece starts on the left-most square (marked **0**). In each round you have a choice of two actions: **STOP**, or **MOVE**. If you choose **STOP**, you are awarded the points marked on the square your piece is on and the game is over. If you choose **MOVE**, your piece moves 1, 2, 3, or 4 squares to the right chosen uniformly at random. If it moves past the end of the board, you are awarded **0** points, and the game is over. Model this as an MDP, and calculate the optimal values **V*** and policy **π*** for each of the four states.

s	**0**	**3**	**2**	**4**	**7**	**5**
V*(s)	5	5	4	4	7	5
π*(s)	MOVE	MOVE	MOVE	STOP	STOP	STOP

7. Perform Q-learning updates resulting from the observed transitions in the table below. From all states the possible actions are →, ↑, ←, and ↓. All Q values are initialized to **0** at the start. Use a learning rate of **α = ½** and a discount factor of **γ = ½**.

State	**Action**	**Reward**	**Next State**
A	→	8	B
B	↓	-4	C
C	←	6	A
A	→	-8	C

$Q(A,\rightarrow) = \frac{1}{2}Q(A,\rightarrow) + \frac{1}{2}(8 + \frac{1}{2}\max Q(B,x)) = 4$

$Q(B,\downarrow) = \frac{1}{2}Q(B,\downarrow) + \frac{1}{2}(-4 + \frac{1}{2}\max Q(C,x)) = -2$

$Q(C,\leftarrow) = \frac{1}{2}Q(C,\leftarrow) + \frac{1}{2}(6 + \frac{1}{2}\max Q(A,x)) = 4$

$Q(A,\rightarrow) = \frac{1}{2}Q(A,\rightarrow) + \frac{1}{2}(-8 + \frac{1}{2}\max Q(C,x)) = -1$

8. Write pseudocode for generating a sequence of outputs given an initially empty context as we did in Makemore. For this purpose, suppose that you have a model **M** that takes some context as input and produces a probability distribution as output.

```
for i = 0 .. num_examples:
    context = empty_context
    output = []
    loop:
        probabilities = M(context)
        result = random_output(probabilities)
        output.append(result)
        if is_terminator(result):
            break loop
        context = context[1:] + [result]
    print(output)
```