3) a) So basically, question asks that if T is a MST of a graph G, and if every edge weight of G is incremented by 1, is T still an MST of G? Our answer is Yes. The simplest proof is that, if G has n vertices, then any spanning tree of G has $(n-1)$ edges. Therefore, incrementing each edge weight by 1 increases the cost of every spanning tree by a constant, $n-1 \Rightarrow$ So, any spanning tree with minimal cost in the original graph also has minimal cost in the new graph.

There are also some alternative ways to paraphrase what we mean by previous proof. For example, assume by the sake of contradiction that T is not an MST of the new graph. Then, there's some other spanning tree of G, call it $T' \neq T$, with lower cost on the new graph. Given a cut (R,S) of G, T has exactly 1 edge e and T' has exactly 1 edge e' crossing the cut. Suppose $e' \neq e$ and e' has lower cost than e. Then, by replacing e with e', we get a new spanning tree for the original graph with lower cost than T, since the ordering of edge weights is preserved when we add 1 to each edge weight. However, this contradicts the assumption that T was an MST of the original graph ✓

b) The answer is no. Suppose, for example, that G consists of an edge from s to t of weight 4, and edges from s to a, a to b, and b to t each of weight 1. Then, the shortest-path is $s \to a \to b \to t$, with cost 3. But, when we increment each edge weight by 1, the shortest path becomes $s \to t$, with cost 5, where $5 < (1+1) + (1+1) + (1+1) = 6$ ■

Scanned with CamScanner

1) In Huffman coding, we build the tree bottom-up by considering pairs of alphabets with the least frequency/probability. So, the 2 least frequent alphabets get assigned the longest code length.

b) In this case, d has the longest code 110, so some other alphabet must also have the code 111, which's not the case

a) Here, c, d have the largest code length, so they have the 2 least probabilities, say $p_c \le p_d$. We start building the tree by forming a dummy vertex labelled v_1 with left-child c, right-child d, and $p_1 = p_c + p_d$. Then, v_1 has the code 11. Now, b has the code 10, i.e. b is the left-child and v_1 is the right-child of a dummy vertex v_2. So, $p_2 = p_b + p_1$ and $p_b \le p_1$. Since out of $\{a, b, v_1\}$, b, v_1 have the longest codes, we also have $p_b, p_1 \le p_a$.

Now, v_2 has the code 1. a has the code 0, i.e. a is the left child and v_2 is the right child of the root vertex R. So, $p_a \le p_2$. So, the constraints we have are:

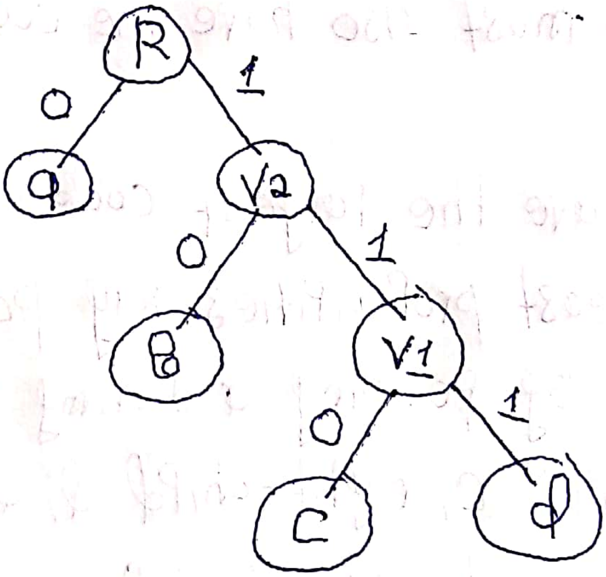
$p_c \le p_d \le p_b \le p_a$

$p_b \le p_c + p_d \le p_a$

$p_a \le p_b + p_c + p_d$

$p_a + p_b + p_c + p_d = 1$

We can check that $\boxed{\begin{array}{l} p_a = 0.41 \\ p_b = 0.29 \\ p_c = 0.15 \\ p_d = 0.15 \end{array}}$ is a solution and it generates the tree in the following picture:

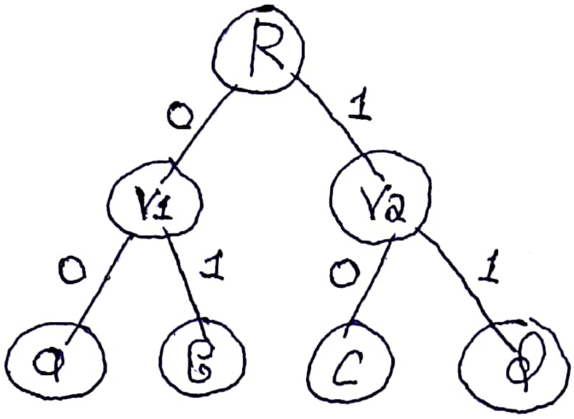


<u>Note:</u> Note that we can talk of probabilities in place of frequencies. If f_i is the frequency of alphabet i, then we assign it the probability $\frac{f_i}{\sum f_i}$ ⟹ The probabilities, so assigned, preserve the relative order among the alphabets based on their frequencies.

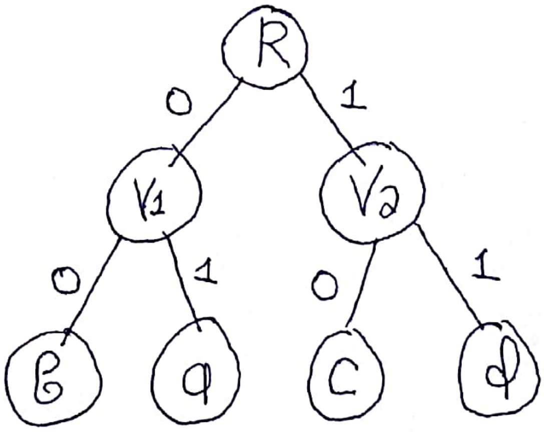
c) In this part, all alphabets have the same code length, so 1 solution is when they are all equiprobable, i.e. $\boxed{p_a = p_b = p_c = p_d = 0.25}$

Scanned with CamScanner

Then, our tree will look like this:



Note that, instead of a, b, c, d we could have b, a, c, d or any other order of the 4 alphabets in the bottom 4 vertices since they have the same probability. But we have to make the tree according to the codes given. The below tree also corresponds to the given probabilities but the code for b is 10, which's not desired ∎



Scanned with CamScanner

Anar Rzayev 2020 Fall CS300 ID: 20190788

Homework # 4

2) Consider any breadth-first spanning tree T of G with x as the root. In T, we know that each node v of G has a level, which is the length of (in other words, # of edges in) a shortest path from x to v. Since each path between x and y has length $\geq l = \frac{n}{2} + 1$, node y occurs at a level $\geq l$

Take levels 1 through $\frac{n}{2} \Rightarrow$ Total amount of nodes in levels 1 through $\frac{n}{2}$ is $\leq (n-2)$ (because we've that nodes x and y don't appear in any of these levels)

If every of those $\frac{n}{2}$ levels has 2 or more nodes, then total # of nodes in G will be greater than n. Thus, there must be a level in the range 1 through $\frac{n}{2}$ containing just 1 node, assuming it is $z \Rightarrow$ clearly, if we delete z, then the resulting graph has no path between x and y ■