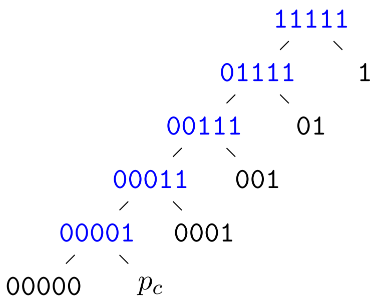
Homework 7

CS41, Spring 2023 Alina Palacios and Nina Zhuo

1. In Huffman coding, we imagine each character as a node in a binary tree where each successive level of the tree will represent a less frequently searched character such that the two least frequently searched characters will be siblings at the lowest level of the tree. Its parent node will be the sum of the binary encoding of both words. Knowing $p_c =$ `00001`, we can build the following tree where the black nodes represent codewords and blue nodes represent the sum of its descendant codewords.

   From here, we can represent the frequency of each codeword as:

   $$1 > 01 > 001 > 0001 > p_c > 00000$$

2. Since we want to find the minimum total weight set F and remove it from G, G will be maximally weighted. We will Kruskal's algorithm on G to find a the minimum set of edges and be left with a maximum spanning tree. The algorithm starts by sorting the edges of G in increasing order of edge weight. Let F be the set of edges that make up the maximum spanning tree. We will set $F = \varnothing$. Then, we isolate each vertex using `MakeSet(u)` for all $u \in V$ such that each vertex belongs to its own singleton set. We add the first edge to F. Using a for loop, we iterate over the set of sorted edges and add the next lightest edge to F by calling `UNION`(u, v) if and only if `FIND`(u) and `FIND`(v) are not equivalent such that adding that edge will not form a cycle in F. If there are no remaining edges, we exit and G is acyclic. We continue this process until there are $V - 1$ edges where V denotes the number of vertices. We stop the algorithm and return set F. The running time is $O(|E|log|V|)$ time; sorting the edges takes $O(|E|log|E|)$ time, and there are $|V|$ `MakeSet` operations, $|V| - 1$ `Union` operations and $2|E|$ `FIND` operations, all of which takes $O(|E|log|E|)$ time. So, the overall running time is $O(|E|log|E|) = O(|E|log|V|)$ time.

3. (a) False; any unique heaviest edge that is not part of a cycle must be in the MST. Suppose G is a graph with one edge. It must be that that edge is the heaviest edge in the set.

   (b) False; building an MST using Kruskal's requires traversing between each vertex using the lightest edge between each vertices, but it could be that a unique shortest $u - v$ path does not use the cheapest edge out from that vertex and therefore is not included in the MST.

(c) False; building an MST using Kruskal's requires traversing between each vertex using the lightest edge between each vertices, but it could be that the unique lightest edge in some cycle in G is not the lightest edge out from that vertex such that e is not necessarily included in every MST.

(d) *Proof.* If we take a greedy algorithm like Kruskal's, we will always start by choosing the edge with the lightest weight to include in an MST. Since e has the minimum weight in G, it will be the lightest edge from any one vertex such that it will always be part of some MST. $\square$