CSCI3160: Regular Exercise Set 5

Prepared by Yufei Tao

Problem 1. Let $G = (V, E)$ be a connected undirected graph where every edge carries a positive integer weight. Divide V into arbitrary disjoint subsets $V_1, V_2, ..., V_t$ for some $t \geq 2$, namely, $V_i \cap V_j = \emptyset$ for any $1 \leq i < j \leq t$ and $\bigcup_{i=1}^{t} V_i = V$. Define an edge $\{u, v\}$ in E as a *cross edge* if u and v are in different subsets. Prove: a cross edge with the smallest weight must belong to a minimum spanning tree (MST).

Solution. Immediate from the "cut property" proved in the Special Exercise List 4. Nevertheless, we give the whole proof below.

Let $e = \{u, v\}$ be a cross edge having the smallest weight. W.l.o.g., suppose that $u \in V_i$ and $j \in V_j$ for some distinct $i, j \in [1, t]$. Consider an arbitrary MST T. If T contains e, we are done. Next, we discuss the case where e is not in T.

Add e to T, which produces a cycle C. Walk on C in the following manner: start from u, cross edge e to reach v, continue in this direction, and stop right after having crossed an edge e' that takes us back to a vertex in V_i. The edge e' must be a cross edge, and hence, must be at least as heavy as e. Deleting e' gives an MST that contains e.

Problem 2* (Kruskal's Algorithm). Let $G = (V, E)$ be a connected undirected graph where every edge carries a positive integer weight. Prove that the following algorithm finds an MST of G correctly:

algorithm
1. $S = \emptyset$
2. **while** $|S| < |V| - 1$
3. find the lightest edge $e \in E$ that does not introduce any cycle with the edges in S
4. add e to S
5. return the tree formed by the edges in S

Solution. Set $n = |V|$. Let $e_1, ..., e_{n-1}$ be the edges picked by the algorithm. We claim that for any $k \in [1, n-1]$, there is an MST that uses $e_1, ..., e_k$. The lemma then follows from the claim at $k = n - 1$. The base case of $k = 1$ is obvious (we proved this in class). Next, assuming correctness at $k = x$ for some integer $x \geq 1$, we will prove the claim for $k = x + 1$.

Let T be an MST that includes $e_1, ..., e_x$. The existence of T is promised by the inductive assumption. If T contains e_{x+1}, we are done; the rest of the proof will focus on the case where e_{x+1} is not in T. Consider the graph $G' = (V, \{e_1, ..., e_x\})$. Denote by $G_1, ..., G_t$ the connected components (CC) of G' for some $t \geq 1$. Let us call an edge $e \in E$ a *cross edge* if it connects two vertices from different CCs.

As e_{x+1} does not introduce any cycle with $e_1, ..., e_x$, we know that e_{x+1} must be a cross edge. Now, add e_{x+1} into T, which gives rise to a cycle. By the same argument as in the solution to Problem 1, we know that the cycle must contain another cross edge e'. By the way e_{x+1} is chosen by the algorithm, we assert that e_{x+1} cannot be heavier than e'. Thus removing e' yields another MST; and this MST contains $e_1, ..., e_{x+1}$, as desired.

Problem 3. Consider Σ as an alphabet. Recall that a *code tree* on Σ is a binary tree T satisfying both conditions below:

- Every leaf node of T is labeled with a distinct letter in Σ; conversely, every letter in Σ is the label of a distinct leaf node in T.
- For every internal node of T, its left edge (if exists) is labeled with 0, and its right edge (if exists) with 1.

Define an *encoding* as a function f that maps each letter $\sigma \in \Sigma$ to a non-empty bit string, which is called the *codeword* of σ. T *produces* an encoding where the code word of a letter $\sigma \in \Sigma$ is obtained by concatenating the bit labels of the edges on the path from the root to the leaf σ. Prove:

- The encoding produces by a code tree T is a prefix code.
- Every prefix code f is produced by a code tree T.

Solution. *Proof of the first bullet:* If the codeword of σ_1 is a prefix of the codeword of σ_2, (by how the codewords are obtained) we can assert that σ_1 is an ancestor of σ_2 in T. But this is impossible because σ_1 needs to be a leaf of T.

Proof of the second bullet: Define $S = \{f(\sigma) \mid \sigma \in \Sigma\}$, namely, S collects the codewords of all the letters in Σ. Grow a binary tree T as follows. Initially, T has only a single leaf. Then, for each letter $\sigma \in \Sigma$, we modify T (if necessary) as follows:

- Initially, set u to the root of T.
- Repeat the following until u is a leaf node:
  - Let ℓ be the level of u.
  - Descend to the left (resp., right) child v of u if the ℓ-th bit of $f(\sigma)$ is 0 (res[., 1). If v does not exist, create it in T, and label its edge with u as 0 (resp., 1).
  - Set u to v.
- Mark the leaf node u with the letter σ.

The final T is a code tree that generates f.

Problem 4. Let T be an optimal code tree on an alphabet Σ (i.e., T has the smallest average height among all the code trees on Σ). Prove: every internal node of T must have two children.

Solution. Let u be any internal node that has a single child v. Let p be the parent of u. Remove u by making v a child of p, and label the edge $\{p, v\}$ appropriately. In the special case where p does not exist (i.e., u is the root), simply make v the new root and delete u. We now have a code tree with strictly smaller average height.

Problem 5* (Textbook Exercise 16.3-7). Consider an alphabet Σ of $n \geq 3$ letters with their frequencies given. The prefix code we construct using Huffman's algorithm is *binary* because each letter $\sigma \in \Sigma$ is mapped to a string that consists of only 0's and 1's. Now, we want the code to be *ternary*, namely, each letter $\sigma \in \Sigma$ is mapped to a string that consists of three possible characters: 0, 1, or 2. As before, the code must be a prefix code. Assuming n to be an odd number, give an algorithm to find an encoding with the shortest average length.

Solution. We define a code tree on Σ as a ternary tree T satisfying:

- There is a one-one correspondence between the leaves of T and the letters in Σ.

- Every internal node u of T has 3 child nodes. The left, middle, and right edges of u carry label 0, 1, and 2, respectively.

For every letter $\sigma \in \Sigma$, the codeword for σ is obtained by concatenating the edge labels from the root of T to the leaf σ.

Let us construct a code tree as follows. Initially, for each character $\sigma \in \Sigma$, create a tree that contains only a single node u, which is labeled with σ. Define the *frequency* of u to be the frequency of σ. In total, there are n trees; collect their roots into a set S. Repeat the following until $|S| = 1$:

- Remove from S the three roots u_1, u_2, and u_3 having the smallest frequencies.
- Create a tree with root u that has u_1, u_2, and u_3 as the child nodes. Define the *frequency* of u as the frequency sum of u_1, u_2, and u_3. This, effectively, combines the three trees — rooted at u_1, u_2, and u_3, respectively — into a new tree, rooted at u. Add u to S.

When $|S| = 1$, we have only one tree left, and this tree is a code tree on Σ. By adapting the argument covered in class, we can prove that Σ generates a prefix code with the shortest average length.