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 $\{u, v\} \in E$ a cross edge if $u$ and $v$ are not in the same subset (i.e., there is no $i \in [1, t]$ satisfying $u \in V_i$ and $v \in V_i$). Prove: the lightest cross edge 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 the lightest cross edge. Without loss of generality, suppose that $u \in V_i$ and $j \in V_j$ for some distinct $i, j \in [1, t]$. Consider any MST $T$ that does not contain $e$. We now add $e$ to $T$ to produce a cycle $C$. Walk on $C$ by starting from $u$, and passing $v$ as the next vertex, but stop as soon as we have crossed an edge $e'$ that brings us back to a vertex on $C$ that belongs to $V_i$. 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. the edges in $S$ now form an MST

Solution. Set $n = |V| - 1$. 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 the 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 that $e_{x+1}$ is not in $T$. Consider the graph $G' = (V, \{e_1, ..., e_x\})$. Denote by $G_1, ..., G_t$ the connected components (CC) of $G'$. Let us call an edge $e \in E$ a cross edge if it connects two vertices from different CCs.

Since $e_{x+1}$ 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 the weight of $e_{x+1}$ cannot be heavier than that of $e'$. Thus removing $e'$ yields another MST; and this MST contains $e_1, ..., e_{x+1}$, as desired.

Problem 3. Consider $\Sigma$ as an alphabet. Recall that a code tree on $\Sigma$ as a binary tree $T$ satisfying both conditions below:

• $C_1$: Every leaf node of $T$ is labeled with a distinct letter in $\Sigma$; conversely, every letter in $\Sigma$ is the label of a distinct leaf node in $T$. 
• $C_2$: 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 $\sigma$. $T$ produces an encoding where the code word of a letter $\sigma \in \Sigma$ can be obtained by concatenating the bit labels of the edges on the path from the root to the leaf $\sigma$.

Prove:

• The encoding produces by a code tree $T$ is a prefix code.
• Every prefix code is produced by a code tree $T$.

**Solution.** *Proof of the first bullet:* Consider any distinct leaf nodes $\sigma_1, \sigma_2$. Let $u$ be their lowest common ancestor. That the bit strings of $\sigma_1, \sigma_2$ are different follows from the fact that the two edges of $u$ carry different labels.

*Proof of the second bullet:* Let $f$ be the encoding that corresponds to the prefix code that we are given. Define $S = \{ f(\sigma) \mid \sigma \in \Sigma \}$, namely, $S$ collects the codewords of all the letters in $\Sigma$. Grow a binary tree $T$ as follows. At the beginning, $T$ has a single leaf. Then, for each letter $\sigma \in \Sigma$, we add some nodes and edges to $T$ (if necessary) as follows:

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

The final $T$ is a code tree of $f$.

**Problem 4.** Consider the alphabet $\Sigma = \{1, 2, \ldots, n\}$ for some integer $n \geq 1$. Suppose that the frequency of $i$ is strictly less than the frequency of $i + 1$, for any $i \in [1, n - 1]$. Prove: in an optimal prefix code, for any $i \in [1, n - 1]$, the codeword of $i$ cannot be longer than that of $i + 1$.

**Solution.** If this is not true, then swapping the codewords of $i$ and $i + 1$ reduces the average length.