CS 170 Dis 5

Released on 2017-02-21

1 Minimum Spanning Trees

For each of the following statements, either prove or supply a counterexample. Always assume $G = (V, E)$ is undirected and connected. Do not assume the edge weights are distinct unless specifically stated.

1. Let e be any edge of minimum weight in G. Then e must be part of some MST.

2. If e is part of some MST of G, then it must be a lightest edge across some cut of G.

3. If G has a cycle with a unique lightest edge e, then e must be part of every MST.

4. For any $r > 0$, define an r-path to be a path whose edges all have weight less than r. If G contains an r-path from s to t, then every MST of G must also contain an r-path from s to t.

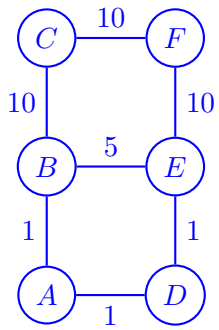
Solution:

1. True, e will belong to the MST produced by Kruskal.

2. True, suppose (u, v) is the edge. Let one side of the cut be everything reachable in the MST from u without using the edge (u, v). If this cut has an edge lighter than (u, v) then we could add this edge to the MST and remove (u, v). We know this edge is not already in the MST because otherwise both its endpoints would be reachable from u without using (u, v).

3. False. Let e be also the heaviest edge of a different cycle; then, we know that e can't be part of the MST. Concretely, in the following graph, edge (B, E) satisfies this condition, but will not be added to the graph.

4. True. Let $v_1, v_2, \ldots, v_n$ denote the r path in G from $s = v_1$ to $t = v_n$. Consider the greatest i such that the MST has an r path from s to v_i and assume for contradiction that $i < n$. Then the edge (v_i, v_{i+1}) is not in the MST. Adding this edge to the MST



forms a cycle, which must have edges of length less than r since otherwise we could replace one of them with (v_i, v_{i+1}). Thus there is an r path from v_i to v_{i+1} and hence from s to v_{i+1}, contradicting our initial assumption.

2 Divide and Conquer for MST?

Is the following algorithm correct? If so, prove it. Otherwise, give a counterexample and explain why it doesn't work.

procedure FindMST(G: graph on n vertices)
If $n = 1$ return the empty set
$T_1 \leftarrow$ FindMST(G_1: subgraph of G induced on vertices $\{1, \ldots, n/2\}$)
$T_2 \leftarrow$ FindMST(G_2: subgraph of G induced on vertices $\{n/2 + 1, \ldots, n\}$)
$e \leftarrow$ cheapest edge across the cut $\{1, \ldots, \frac{n}{2}\}$ and $\{\frac{n}{2} + 1, \ldots, n\}$.
return $T_1 \cup T_2 \cup \{e\}$.

Solution: This algorithm does not work; multiple edges of the MST could cross this particular cut. Another way to see this is that the MSTs of the subgraph needn't also be part of the MST of the whole graph.
As a concrete counterexample, consider a wide rectangle and the horizontal cut between the top two vertices and the bottom two. Both edges on this cut should be in the MST.

3 Huffman Proofs

1. Prove that in the Huffman coding scheme, if some character occurs with frequency more than $\frac{2}{5}$, then there is guaranteed to be a codeword of length 1. Also prove that if all characters occur with frequency less than $\frac{1}{3}$, then there is guaranteed to be no codeword of length 1.

2. Under a Huffman encoding of n symbols with frequencies $f_1, f_2, \ldots, f_n$, what is the longest a codeword could possibly be? Give an example set of frequencies that would produce this case, and argue that it is the longest possible.

Solution:

1. Suppose all codewords have length at least 2 – the tree must have at least 2 levels. Let the weight of a node be the sum of the frequencies of all leaves that can be reached from that node. Suppose the weights of the level-2 nodes are (from left to right for a tree rooted on top) a, b, c, and d. Without loss of generality, assume A and B are joined first, then C and D. So, $a, b \leq c, d \leq a + b$.

   If a or b is greater than $\frac{2}{5}$, then both c and d are greater than $\frac{2}{5}$, so $a+b+c+d > \frac{6}{5} > 1$ (impossible). Now suppose c is greater than $\frac{2}{5}$ (similar idea if d is greater than $\frac{2}{5}$). Then $a + b > \frac{2}{5}$, so either $a > \frac{1}{5}$ or $b > \frac{1}{5}$ which implies $d > \frac{1}{5}$. We obtain $a + b + c + d > 1$ (impossible).

   For the second part suppose there is a codeword of length 1. We have 3 cases. Either the tree will consist of 1 single level-1 leaf node, 2 level-1 leaf nodes, or 1 level-1 leaf node, and 2 level-2 nodes with an arbitrary number of leaves below them in the tree. We will prove the contrapositive of the original statement, that is, that if there is a codeword of length 1, there must be a node with frequency greater than $\frac{1}{3}$

   In the first case, our leaf must have frequency 1, so we've immediately found a leaf of frequency more than $\frac{1}{3}$. If the tree has two nodes, one of them has frequency at least $\frac{1}{2}$, so condition is again satsified. In the last case, the tree has one level-1 leaf (weight a), and two level-2 nodes (weights c and d). We have: $b, c \leq a$. So $1 = a + b + c \leq 3a$, or $a \geq \frac{1}{3}$. Again, our condition has been satisfied.

2. The longest codeword can be of length $n - 1$. An encoding of n symbols with $n - 2$ of them having probabilities $1/2, 1/4, \ldots, 1/2^{n-2}$ and two of them having probability $1/2^{n-1}$ achieves this value. No codeword can ever by longer than length $n - 1$. To see why, we consider a prefix tree of the code. If a codeword has length n or greater, then the prefix tree would have height n or greater, so it would have at least $n + 1$ leaves. Our alphabet is of size n, so the prefix tree has exactly n leaves.

4 Horn Formula Practice

Find the variable assignment that solves the following horn formulas:

1. $(w \wedge y \wedge z) \Rightarrow x, (x \wedge z) \Rightarrow w, x \Rightarrow y, \Rightarrow x, (x \wedge y) \Rightarrow w, (\bar{w} \vee \bar{x}, \vee \bar{y}), (\bar{z})$

2. $(x \wedge z) \Rightarrow y, z \Rightarrow w, (y \wedge z) \Rightarrow x, \Rightarrow z, (\bar{z} \vee \bar{x}), (\bar{w} \vee \bar{y} \vee \bar{z})$

Solution:

1.
- Set everything to false initially
- x must be true since we have the statement $\Rightarrow x$
- y must be true since $x \Rightarrow y$
- w must be true since $(x \wedge y) \Rightarrow w)$
- Not all negative clauses are satisfied at this point, so there is no satisfying assignment.

2.
- z must be true since we have the statement $\Rightarrow z$.
- w must be true since $z \Rightarrow w$
- x and y need not be changed, as all our implications are satisfied.
- All negative clauses are now satisfied, so we've found our satisfying assignment.