

CS 170 Spring 2008 - Solutions to Homework #7

Problem 5.1 (12 points total, 4 points each)

- (a) The cost of MST is 19.
- (b) There exist two minimum spanning trees for this graph.

	Edge included	Cut
(c)	<i>AE</i>	$\{A, B, C, D\} \& \{E, F, G, H\}$
	<i>EF</i>	$\{A, B, C, D, E\} \& \{F, G, H\}$
	<i>BE</i>	$\{A, E, F, G, H\} \& \{B, C, D\}$
	<i>FG</i>	$\{A, B, E\} \& \{C, D, F, G, H\}$
	<i>GH</i>	$\{A, B, E, F, G\} \& \{C, D, H\}$
	<i>CG</i>	$\{A, B, E, F, G, H\} \& \{C, D\}$
	<i>GD</i>	$\{A, B, C, E, F, G, H\} \& \{D\}$

Problem 5.2 (8 points total, 4 points each)

	Vertex included	Edge included	Cost
(a)	A		0
	B	AB	1
	C	BC	3
	G	CG	5
	D	GD	6
	F	GF	7
	H	GH	8
	E	AE	9

- (b) See Figure ??.

Problem 5.6 (10 points)

We can prove by contradiction. Suppose the graph has two different MSTs T and T' . Then there must exist an edge (u, v) which is present in T but not in T' . The graph obtained by removing the edge (u, v) from T consists of two connected components, which define a cut on G . There must be some other edge (u', v') across this cut in T' . Since all edges have different weights, without



Figure 1: The solution to Prob 5.2(b)

loss of generality, we can assume that $w(u, v) < w(u', v')$. Removing (u', v') from T' and adding (u, v) gives a tree of lower cost than T' . However, this is a contradiction since T' was assumed to be an MST. Therefore, the MST of a graph with distinct edge weights must be unique.

Problem 5.13 (5 points)

Huffman's algorithm assigns codewords of length 1 to T, length 2 to A and length 3 to C and G. So, one possible encoding can be 0 for T, 10 for A, 110 for C and 111 for G.

Problem 5.21 (10 points)

Algorithm: Negate the edge weights of the original graph, run Kruskal (or Prim) and find all MSTs of the connected components of the negated graph. Then go through all the edges of the original graph and output the edges, which do not belong to MSTs obtained in the previous step.

Correctness: Given a connected graph and one of its spanning tree, whenever

an edge, which does not belong to the spanning tree, is added to the spanning tree, a cycle is created. Therefore, all edges, which are not in a spanning tree, forms a feedback arc set of the graph. As the total weight of the graph is fixed, to minimize the feedback arc set weight, we need to maximize the weight of the spanning tree, which is equivalent to minimize the weight of the spanning tree of the graph with the edge weight negated. The MSTs of the connected components of a graph can be obtained by running Kruskal or Prim's algorithm. Time analysis: the running time is the same as Kruskal or Prime ($O(|E| \log |V|)$).

Problem 5.22 (16 points total, 4 points each)

- (a) Consider an MST T , which contains the heaviest edge e on a cycle. Removing e breaks T into two connected components, say S and $V \setminus S$. On the cycle, there must exist another edge, say e' , cross from S to $V \setminus S$. Otherwise it contradicts with the existence of the cycle. Replacing e by e' gives a tree T' such that $cost(T') \leq cost(T)$ as $w(e') \leq w(e)$. Since T' is already an MST, T' has also to be an MST which does not contain e .
- (b) If e is the heaviest edge in some cycle of G , then there is some MST T not containing e as proved in (a). Therefore T is also an MST of $G - e$ and we can simply search for an MST of $G - e$. At every step, the algorithm creates a new graph ($G - e$) such that an MST of the new graph is also an MST of the old graph (G). Hence the output of the algorithm (when the new graph becomes a tree) is an MST of G .
- (c) An undirected edge (u, v) is part of cycle iff u and v are in the same connected component of $G - e$. Since the components can be found by DFS (or BFS), this gives a linear time algorithm.
- (d) The time for sorting is $O(|E| \log |E|)$. Checking for a cycle at every step takes $O(|E|)$ time. Since the final MST has $|V| - 1$ edges, we need to remove $|E| - |V| + 1$ edges. The total running time is $O(|E| \log |E| + (|E| - |V|) * |E|) = O(|E|^2)$.

Problem 5.23 (16 points total, 4 points each)

To solve this problem, we shall use the characterization that T is an MST of G , if and only if for every cut of G , at least one least weight edge across the cut is contained in T .

For the "only if" direction, assume by contradiction that T is a MST, but that the lightest edge e_1 across some cut is not in T . If we add e_1 to T , we create a cycle. Now at least two of the cycle's edges must cross the cut: e_1 and at least one other edge e_2 since e_1 forms a cycle. Thus, if we remove e_2 , we eliminate the cycle and get a new spanning tree of smaller cost than T (since $w(e_2) > w(e_1)$). This contradicts the assumption that T is a MST.

To prove the “if” part, note that at each step in Prim’s algorithm, we include the lightest edge across some cut. But for every possible cut, T contains a lightest edge, so Prim’s algorithm can choose edges from T at each step. Since Prim’s algorithm produces a MST, and T is a possible output of Prim’s (the only possible output if there are no ties), then T must be an MST.

- (a) Since the change only increases the cost of some other spanning trees (those including e) and the cost of T is unchanged, it is still an MST.
- (b) We include e in the tree, thus creating a cycle. We then remove the heaviest edge e' in the cycle, which can be found in linear time, to get a new tree T' . We claim that T' contains a least weight edge across every cut of G and is hence an MST.

Note that since the only changed edge is e , $T \cup \{e\}$ already includes a least weight edge across every cut. We only removed e' from this. However, any cut crossed by e' , must also be crossed by at least one more edge of the cycle, which must have weight less than or equal to e' . Since this edge is still present in T' , it contains a least weight edge across every cut.

- (c) The tree is still an MST if the weight of an edge in the tree is reduced. Hence, no changes are required.
- (d) We remove e from T to obtain two components S_1 and S_2 , and hence a cut. We then include the lightest edge l across the S_1 - S_2 cut to get a new tree T_{new} .

To prove that T_{new} is an MST of the updated graph, we consider two cases: when the updated edge remains in the MST and when it does not.

For the former case, note that if some MST M of the updated graph contains the updated edge e , then $c(M)$ must equal $c(T) + \delta(e)$, where c is the cost function and $\delta(e)$ is the increase in e ’s edge weight. However, $c(T_{new}) \leq c(T) + \delta(e)$, so T_{new} must also be an MST.

For the latter case, suppose to the contrary that there is another spanning tree T' with weight less than T_{new} , i.e. $c(T') < c(T_{new}) = c(T) - c(e) + c(l)$, where c is the cost function and e represents the original edge (before updating). Now consider the spanning tree T'' , where $T'' = T' + e - h$, where h is the heaviest edge on the cycle formed by adding the original edge e to T' . T'' is a spanning tree of the original graph because T' does not contain the updated edge e . Thus, adding the original edge e creates a cycle with at least two edges, e and s crossing the $S_1 - S_2$ cut. Since h is the heaviest edge on this cycle, $c(h) \geq c(s) \geq c(l)$. Therefore, we have $c(T'') = c(T') + c(e) - c(h) \leq c(T') + c(e) - c(l)$. Furthermore, since $c(T') < c(T) - c(e) + c(l)$, we have $c(T'') < (c(T) - c(e) + c(l)) + c(e) - c(l) = c(T)$. This is a contradiction since it was assumed T was a MST of the original graph. Therefore, our assumption of a tree T' of weight less than T_{new} was false, and T_{new} must be a MST of the updated graph.

Problem 5.29 (10 points)

If we consider the binary tree with all strings of length k at level k and the left and right branches representing adding a 0 and 1 respectively, it contains all the binary strings and hence all the strings in the encoding (we only need to have number of levels equal to maximum length of a string). Also, since all intermediate nodes in a path from the root to a node v are prefixes of v , the strings of the encoding must be all at leaves since it is prefix-free.

To argue that the tree must be full, suppose, for contradiction, that a node u corresponding to a string s has only one child v corresponding to $s0$. Since a codeword is a leaf in the subtree rooted at u iff it has s , and hence also $s0$ as a prefix, replacing $s0$ by s in all these codewords gives a better encoding. However, this is a contradiction since we assumed our encoding to be minimal.