CMPSCI 311: Introduction to Algorithms **Spring 2018**

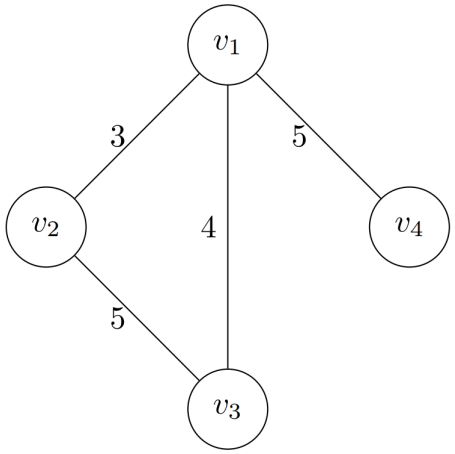
Homework 3 Solutions

Released 2/21/2018 Due 11:59pm 3/7/2018 in Gradescope

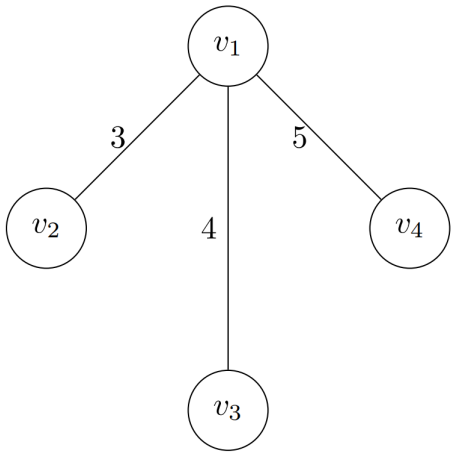
Note: *LaTeX template courtesy of UC Berkeley EECS dept.*

1. **Spanning Trees K&T Ch4.Ex9.**

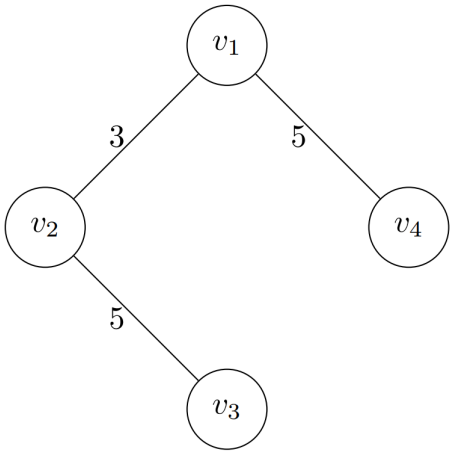
   (a) False. Consider the following graph example:



The MST is:



But a candidate MBST can be:



(b) True. Proof by contraposition: Suppose that T is not an MBST of G. We will work to prove that T is also not an MST, which by contraposition will establish claim. Let us use e to denote the heaviest edge in T and suppose that $e = (u, v)$. Let T' be an MBST of G.

Now, let S be all of the nodes that are reachable from u using only the edges in T, but not passing through v (i.e. reachable using $T \setminus \{e\}$). Let S^C be all the other nodes (these are the nodes reachable from v using only the edges in $T \setminus \{e\}$). We will apply the cut lemma to this cut (S, S^C).

Since T' is an MBST, it is also a spanning tree, so it must have at least one edge crossing this cut, call any one of these edges e'. On the other hand, by construction T has exactly one edge crossing this cut, and it is precisely the edge e, which we know is the heaviest edge in T. Since T' is an MBST and T is not, we know that every edge in T' that crosses this cut, has smaller weight than e, so in particular $w(e') < w(e)$, but $e' \notin T$. By the cut lemma, this implies that T is not an MST.

2. **More Spanning Trees K&T Ch4.Ex10.**

(a) We do this under the assumption that all edge costs are distinct. In this case, we can solve (a) as follows. Let $e = (u, v)$ be the new edge being added. We represent T using an adjacency list, and we find the $u - v$ path P in T in time linear in the number of nodes and edges of T, which is $O(|V|)$. If every node on this path in T has cost less than $w(e)$, then the Cycle Property implies that the new edge $e = (v, w)$ is not in the minimum spanning tree, since it is the most expensive edge on the cycle C formed from P and e, so the minimum spanning tree has not changed. On the other hand, if some edge on this path has cost greater than $w(e)$, then again by the Cycle Property the most expensive edge cannot be in the minimum spanning tree, and so T is no longer the MST.

(b) We replace the heaviest edge on the $u - v$ path P in T with the edge $e = (u, v)$, obtaining a new spanning tree T'. We claim that T' is a minimum spanning tree. To prove this, we consider any edge e' not in T', and show that we can apply the Cycle Property to conclude that e' is not in any MST. So let $e' = (u', v')$. Adding e' to T' gives us a cycle C' consisting of the $u' - v'$ path P' in T', plus e'. If we can show e' is the most expensive edge on C', we are done.

Intuitively, there are two cases. The first case is that e' forms a cycle that is completely independent of the cycle formed by adding the edge e. In this case, we can simply use the fact that T was an MST before e was added to argue that e' should not be included. The second case is that e' forms a cycle that includes the new edge e. This case is trickier.

For the details, consider one further cycle: the cycle K formed by adding e' to T. By the Cycle Property, e' is the most expensive edge on K. So now there are three cycles: C (the cycle formed by adding e to T), C' (the cycle formed by adding e' to T'), and K (the cycle formed by adding e' to T). Edge f is the most expensive edge on C, and edge e' is the most expensive edge on K.

We need to prove that e' is the most expensive edge in C'. Now, if the new edge e does not belong to C', then $C' = K$, so e' is the most expensive edge on K. Otherwise, the cycle K includes f (since C' needed to use e instead), and C' uses a portion of C (including e) and a portion of K. In this case, e' is more expensive than f (since f lies on K), and hence it is more expensive than everything on C (since f is the most expensive edge on C). It is also more expensive than everything else on K, and so it is the most expensive edge on C', as desired.

Note for both parts a and b: If the assumption that edge costs are all distinct is not made, then we need to perturb the values of edges by very small amounts such that all of the edge weights become distinct (as explained in the chapter).

3. **Fast Matrix Multiplication.** Given two $n \times n$ matrices X, Y of integers, their product is another $n \times n$ matrix Z with,

   (a) This proof follows from the fact that i^{th} row of X and j^{th} column of Y are to be multiplied to obtain the value of Z_{ij}. Consider A, B and E, G. While A, B constitute the first $n/2$ rows of X, E, G constitute the first $n/2$ columns of Y. Also since this is a square matrix, and A, B and E, G are equally split, the proof follows. Mathematically, we know that $A_{ij} = X_{ij}, B_{ij} = X_{i,n/2+j}, E_{ij} = Y_{ij}, G_{ij} = Y_{n/2+i,j}$ and similarly for E, G, so we get

$$\begin{aligned}(AE + BG)_{ij} &= \sum_{k=1}^{n/2} A_{ik}E_{kj} + \sum_{k=1}^{n/2} B_{ik}G_{kj} = \sum_{k=1}^{n/2} X_{ik}Y_{kj} + \sum_{k=1}^{n/2} X_{i,n/2+k}Y_{n/2+k,j} \\ &= \sum_{k=1}^{n} X_{ik}Y_{kj} = (XY)_{ij}\end{aligned}$$

   The same logic follows for other 4 blocks of the matrix

   (b) We divide the given matrix into 4 smaller equally sized matrices and recursively call the eight mulitplications on the sub-matrices. The product is calculated using the relation shown in the last section. We use index calculations to avoid copying of the submatrices. Complexity analysis: Total time can be represented as a sum of three parts- one, partitioning time, two, recursive calls, and three, addition of the recursive calls.

$$T(n) = \Theta(1) + 8T(n/2) + \Theta(n^2) \Rightarrow T(n) = 8T(n/2) + \Theta(n^2)$$

   From master's theorem: $T(n) = \Theta(n^3)$ (this is included in the formula from class).

   (c) Recurrence relation:

$$T(n) = \begin{cases} \Theta(1), & \text{if } x = 1 \\ T(n) = 7T(n/2) + \Theta(n^2), & \text{if } x > 1 \end{cases}$$

   This solves to $T(n) = \Theta(n^{\log_2 7}) = \Theta(n^{2.807})$.

4. **Decimal to Binary.**

   (a) $z = \text{pwr2bin}(n/2)$. This produces the binay representation of $10^{n/2}$. Then by returning fastmultiply(z, z), we return $10^{n/2} \times 10^{n/2} = 10^n$.

   Running time: $T(n) = T(n/2) + O(n^{\log_2 3})$, which solves to $T(n) = \Theta(n^{\log_2 3})$.

   (b) return fastmultiply(dec2bin(x_L), pwr2bin($n/2$)) + dec2bin(x_R). This computes $x_L \times 10^{n/2} + x_R = x$ in binary.

   Total time can be represented as a sum of two parts- recursive calls, and subroutine calls. Since we know that fastmultiply and dec2bin take $O(n^{\log_2 3})$ time, we get the recurrence relation: $T(n) = 2T(n/2) + O(n^{\log_2 3}) = O(n^{\log_2 3})$.

5. **(0 points).** How long did it take you to complete this assignment?