CS41 Homework 6

This homework is due at 11:59PM on Wednesday, October 26. Write your solution using LaTeX. Submit this homework in a file named `hw6.tex` using **github**.

This is a partnered homework. You should primarily be discussing problems with your homework partner. It's ok to discuss approaches at a high level with others. However, you should not reveal specific details of a solution, nor should you show your written solution to anyone else. The only exception to this rule is work you've done with a lab teammate *while in lab*. In this case, note (in your **homework submission poll**) who you've worked with and what parts were solved during lab.

The main **learning goals** of this homework assignment are to practice solving recurrence relations and to work with divide and conquer algorithms.

1. **Recurrence Relations.** Solve the following recurrence relations.

   (a) $T(n) = 4T(n/2) + 3n$,
   $T(1) = 4$

   (b) $W(n) = 3W(n/3) + n$,
   $W(1) = 1$

   Note: You must use Substitution Method for one of the recurrences and Recursion Tree for the other.

2. **Divide and conquer minimum spanning trees?**

   Joshua has a really cool idea for a divide and conquer algorithm which will find a MST. Given a connected, undirected graph $G = (V, E)$ with weighted edges, Joshua's algorithm does the following:

   - Divides the graph into two pieces, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. ($V_1 \cup V_2 = V$ and V_1 and V_2 are disjoint. E_1 is the edges in E with both endpoints in V_1, and E_2 is the edges in E with both endpoints in V_2.)
   - Recursively finds the MSTs M_1 for G_1 and M_2 for G_2.
   - Finds the lowest-weight edge $e = (u, v)$ with $u \in V_1$ and $v \in V_2$.
   - Returns the minimum spanning tree $M_1 \cup M_2 \cup \{e\}$.

   Unfortunately, this algorithm does not work. Give an example input graph G with weights and describe a run of this algorithm where the algorithm does not return a minimum spanning tree on G.

3. **(extra challenge) Divide and conquer for minimum spanning trees (V2.0)**

   Is it possible to "patch" Joshua's algorithm to work, if the vertex partition is chosen cleverly? That is, can we do a little bit of conquering *before* the divide step(s), which will make this divide-and-conquer MST algorithm work?

   If YES, then describe how to fix this divide and conquer algorithm to be correct. If NO, then argue why no rule for dividing G can make the algorithm correct.