Problem 2-1. Work of Funnel Sort
Analyze the work of the funnel sort algorithm we saw in class. You should write the recurrence, solve it using a tree, and then provide an induction proof.

Problem 2-2. Solve another recurrence
We will now practice solving recurrences that do not follow the master method similar to the funnel sort recurrences we saw in class. You need not do the induction proof, but should draw the tree and make a reasonable argument for your solution.

(a) Using a recurrence tree, guess the solution to the recurrence $Q(n) = 2n^2Q(n^{1/2}) + \Theta(n^3)$.
Solve this recurrence for two base cases: (1) $Q(n) = c$ for $n = 2$ and some constant $c$; and (2) $Q(n) = n^4/L$ for $n \leq \alpha \sqrt{Z}$ for some constant $\alpha$.

(b) $Q(n) = n^{1/2}Q(n^{1/2}) + \Theta(n)$. The base cases are (1) $Q(n) = c$ for $n = 2$ and some constant $c$; and (2) $Q(n) = n^2/L$ for $n \leq \alpha \sqrt{Z}$ for some constant $\alpha$.

(c) $Q(n) = 4n^{1/3}Q(n^{1/3}) + \Theta(n^2)$. The base cases are (1) $Q(n) = c$ for $n = 2$ and some constant $c$; and (2) $Q(n) = n^2/L$ for $n \leq \alpha \sqrt{Z}$ for some constant $\alpha$.

Problem 2-3. Drifted Nodes
In class, we saw that for nested-parallel computations, the number of drifted nodes is at most $2S$ where $S$ is the number of steal attempts. Give an example of a non-series parallel computation where the number of drifted nodes can be larger. You should still assume that a node has 0, 1, or 2 children. You should draw a DAG and then sketch out an execution that allows for more drifted nodes. We did an example in class. You can use the same example or come up with your own.

Problem 2-4. Range Additions
You are given an array $A[1..n]$ with $n$ numbers; in addition, you are given another array $T$ with $m$ triples of the form $(\text{start}, \text{end}, \text{value})$. You want to add $\text{value}$ to all elements in the range from $\text{start}$ to $\text{end}$ (that is all elements of $A$ from $A[\text{start}]$ to $A[\text{end}]$). Give an efficient parallel algorithm to do all $m$ operations on array $A$ and analyze work and span.

$\text{value}$ can be a positive or a negative number. As a first step, you can try the case where each $\text{start}$ and $\text{end}$ for all triples in $T$ is unique — meaning if 2 is the start value for any triple in $T$, then it will not appear as either $\text{start}$ or $\text{end}$ for any other triple. You should try to minimize work and span. Is your algorithm work-efficient?

Hint: Use the scan operation.