Problem 2-1. Work Stealing Analysis [35 points]

In class, we proved a bound on work stealing when each node has at most 2 children. We will now try to analyze the scheduler when nodes can have up to 3 children — that is, a spawn node can have 2 or 3 children. When a processor executes a node, it can now enable 0, 1, 2 or 3 children. The scheduler operates in the same way as it did when we discussed it in class; however, when a processor enables 3 children, one of the children is assigned and the other two are placed at the bottom of the deque.

Note: There are multiple correct answers to this problem — in particular, depending on how you designed the potential, you will get different constants.

(a) [10 points] Our potential function was $\Phi(u) = 3^{2w(u)}$ where $w(u) = T_\infty - d(u)$. What is the new potential function? Prove that with this new potential function, the potential never increases.

(b) [10 points] Prove the new top-heavy deque lemma. That is, say you have a processor $p$ with total potential (the sum of the potential of all the nodes on the deque as well as the potential of the assigned node) $\Phi(p)$. You must prove that the potential of the node on the top of the deque is at least $c\Phi(p)$ for some constant $c$.

(c) [15 points] Finally, show that if there were $kP$ steal attempts, for some constant $k$, between time $t_1$ and $t_2$, then $\Pr\{\Phi_{t_1} - \Phi_{t_2} \geq c_1\Phi_{t_1}\} \geq c_2$ for some constants $c_1$ and $c_2$. ($\Phi_{t_1}$ and $\Phi_{t_2}$ are the total potentials of the computation at time $t_1$ and $t_2$ respectively.)

Note that once you have proven the above lemma, you can then work through the full proof by just applying standard probabilities.

Problem 2-2. Solve another recurrence [40 points]

The first recurrence is the same as the one we solve for funnel sort in class. The other two are for more practice.

(a) [10 points] Using a recurrence tree, guess the solution to the recurrence $Q(n) = 2n^{2/3}Q(n^{1/2}) + \Theta(n^2)$. Solve this recurrence for two base cases: (1) $Q(n) = c$ for $n = 2$ and some constant $c$; and (2) $Q(n) = n^{3}/L$ for $n \leq \alpha\sqrt{Z}$ for some constant $\alpha$. The second one is the one we did in class.

(b) [15 points] $Q(n) = n^{1/2}Q(n^{1/2}) + \Theta(n)$. 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) [15 points] $Q(n) = 4n^{1/3}Q(n^{1/3}) + \Theta(n^2)$. 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 $\alpha$. 

1
Problem 2-3. Drifted Nodes [25 points]

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.