Recitation 2 — Recurrences

Parallel and Sequential Data Structures and Algorithms, 15-210 (Fall 2012)

September 5, 2012

1 Announcements

- HW1 is due tonight at 23:59EST. Hopefully you have all started by now; if not, now would be a good time.
- HW2 will be out tomorrow or at the latest on Friday, and is due next Wednesday. There will be two main programming problems, which we will introduce to you towards the end of today's recitation.
- If you are not able/want to use Piazza to contact the course staff, you may send email to

  `15210-staff@lists.andrew.cmu.edu`.

- Questions from lecture, homework, or life?

2 Recurrences

Let's start by solving a recurrence which should be familiar to all of you as a warmup:

$$W(n) = 2W(n/2) + O(n)$$

Suppose $W(1) \in O(1)$. We claim $W(n) \in O(n)$. Is this true? Let's try to prove it, by induction.

Base case: Given.

Inductive hypothesis: For all i<n, $W(i) = O(i)$.

Inductive case:

$$\begin{aligned} W(n) &= 2W(n/2) + O(n) \\ &= 2\left[O(n/2)\right] + O(n) \\ &= 2O(n) + O(n) \\ &= O(n) \end{aligned}$$

So, we proved that $W(n) \in O(n)$. Or did we?

2.1 A Closer Look

What went wrong? Let's take a closer look at the definition of Big-O.

Recall from lecture:

Lemma 2.1. *If $f = O(n)$, there exist constants k_1, k_2 so that $f(n) \leq k_1 n + k_2, n \geq 0$*

Proof. By the definition of Big-O, $f = O(n)$, there exist c and n_0 so that $f(n) \leq cn$ for $n > n_0$. Then $k_1 = c, k_2 = \max(f(i) : 0 \leq i < n_0)$ works. □

When we say $W(n) \in O(n)$, we mean there exists some n_0, c such that for all $n > n_0$, $W(n) \leq c \cdot n$. That is to say, n_0 and c must both be fixed. Specifically, we want to show $W(n) \leq c_1 \cdot n + c_2$. This isn't the case in our proof of the inductive case:

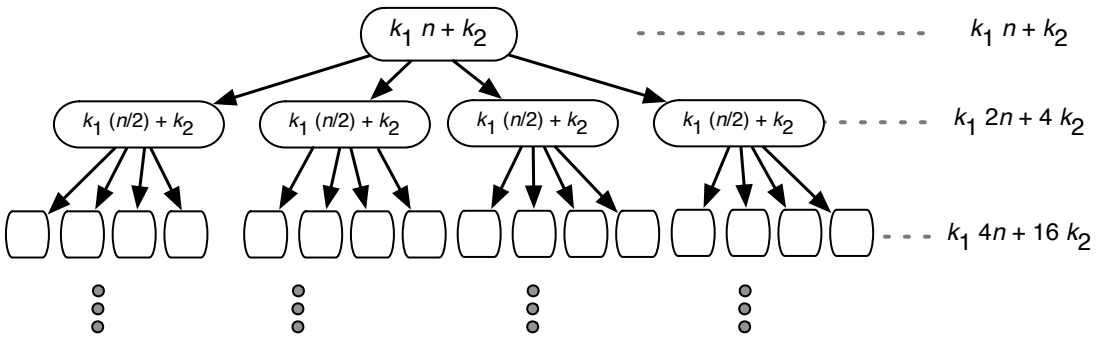
$$\begin{aligned} W(n) &\leq 2W(n/2) + k_1 \cdot n + k_0 \\ &\leq 2\left[c_1 \cdot n/2 + c_2\right] + k_1 \cdot n + k_0 \\ &\leq (c_1 + k_1) \cdot n + 2 \cdot c_2 + k_0 \qquad \not\leq c_1 \cdot n + c_2 \end{aligned}$$

Suddenly, we have a constant factor of n that is more than c_1, so we cannot conclude that $W(n) \in O(n)$.

2.2 Brick Method

Yesterday in lecture we went over the brick method for determining if a recurrence is root-dominated, leaf-dominated, or balanced. It's a good way to get started when solving a recurrence.

- For $W(n) = 4W(n/2) + O(n)$, the recursion tree is:



That is, we have at level i:

Problem Size	$n/2^i$
Node Cost	$\leq k_1(n/2^i) + k_2$
Number of Nodes	4^i

So the cost at each level is bounded by

$$4^i \cdot \left(k_1(n/2^i) + k_2\right) = k_1 \cdot 2^i \cdot n + 4^i \cdot k_2$$

This gives us a stack of bricks which is dominated at the leaves because the cost at level i geometrically *increases* by more than a constant factor of 2. So $W(n) = O(\text{number of leaves}) = O(n^2)$, since the leaves are at level $\log_2 n$ and there are $4^{\log_2 n} = n^2$ of them.

- For $W(n) = W(3n/4) + O(n)$, we have at level i:

Problem Size	$(3/4)^i n$
Node Cost	$\leq k_1(3/4)^i n + k_2$
Number of Nodes	1

The cost at each level is bounded by

$$1 \cdot \left(k_1(3/4)^i + k_2\right) = k_1 \cdot (3/4)^i \cdot n + k_2$$

This gives us a stack of bricks which is dominated at the root node because the cost at level i geometrically *decreases* by a constant factor of $3/4$. So $W(n) = O(\text{cost at root}) = O(n)$.

- For $W(n) = 2W(n/2) + O(n)$, we have at level i:

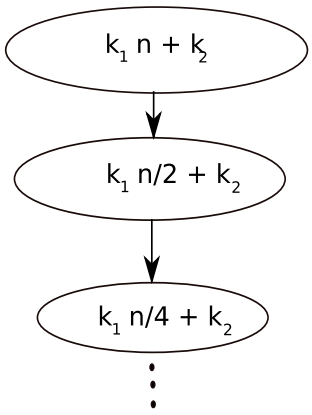
Problem Size	$n/2^i$
Node Cost	$\leq k_1(n/2^i) + k_2$
Number of Nodes	2^i

The cost at each level is bounded by

$$2^i \cdot \left(k_1(n/2^i) + k_2\right) = k_1 \cdot n + 2^i \cdot k_2$$

This gives us a stack of bricks which is balanced throughout because the cost at every level is the same, within a constant factor. So $W(n) = O(\text{height of tree} * \text{work at each level}) = O(n \log(n))$.

- For $W(n) = W(n/2) + O(n)$, we have at level i:



Problem Size	$(1/2)^i n$
Node Cost	$\leq k_1(1/2)^i n + k_2$
Number of Nodes	1

The cost at each level is bounded by

$$1 \cdot \left(k_1(1/2)^i + k_2\right) = k_1 \cdot (1/2)^i \cdot n + k_2$$

This gives us a stack of bricks which is dominated at the root node because the cost at level i geometrically *decreases* by a constant factor of $1/2$. So $W(n) = O(\text{cost at root}) = O(n)$.