15-210 Assignment 02

Karan Sikka
ksikka@cmu.edu
Section C
September 20, 2012

1: Task 4.1

For all of the recurrences, we assume that $T(1)$ is a constant.

1. Given the recurrence, we can interpret it to mean:

$$T(n) \leq 3T(\frac{n}{4}) + k_1 n + k_2$$
$$T(n) \geq 3T(\frac{n}{4}) + k_3 n + k_4$$

for all $n \in \mathbb{N}$. Since these inequalities are so similar, we will solve the following recurrence of a stronger statement of the first inequality, and then use the closed form to prove things about both inequalities:

$$T(n) = 3T(\frac{n}{4}) + k_1 n + k_2$$

Then we use the tree method. Let i be the index of the row of the tree:

Value at each node: $(\frac{1}{4})^i(k_1 n + k_2)$

Nodes per row: 3^i

Sum of nodes per row: $(\frac{3}{4})^i(k_1 n + k_2)$

Height of tree: $lg(n)$

Sum of nodes in the tree:

$$T(n) = \sum_{i=0}^{lg(n)} (\frac{3}{4})^i (k_1 n + k_2)$$

We will find a closed form of this expression via algebra.

$$T(n) = (k_1 n + k_2) \sum_{i=0}^{lg(n)} (\frac{3}{4})^i$$

Now we use the formula for the sum of a finite geometric series, and we use this closed form to prove the claim.

$$T(n) \leq (k_1 n + k_2) \frac{1 - (\frac{3}{4})^{lg(n)+1}}{1 - \frac{3}{4}}$$
$$T(n) \geq (k_3 n + k_4) \frac{1 - (\frac{3}{4})^{lg(n)+1}}{1 - \frac{3}{4}}$$

We observe that the numerators of the fractions will always be less than or equal to 1, and it must be greater than .25, so we come up with the following upper and lower bounds:

$$T(n) \leq 4(k_1 n + k_2)1$$
$$T(n) \geq 4(k_3 n + k_4).25$$

Therefore, $T(n)$ is upper-bounded by $O(n)$ and lower-bounded by it as well. Then $T(n) \in \Theta(n)$

2. Given the recurrence, we can interpret it to mean:

$$T(n) \leq 21T(\frac{n-3}{23}) + k_1 n + k_2$$
$$T(n) \geq 21T(\frac{n-3}{23}) + k_3 n + k_4$$

Claim: $T(n) \in \Theta(n)$

The proof is by strong induction on n.

Base case:
$T(1)$ is some constant, so it is in $\Theta(1)$. Since $n = 1$, the claim holds for the base case.

Inductive Hypothesis:
$T(x) \in \Theta(x)$ for all $x < n$.
$T(x) \leq k_5 x + k_6$ for all $x < n$.
$T(x) \geq k_7 x + k_8$ for all $x < n$.

Inductive Step:

(a) Proof of $T(n) \in O(n)$

$$T(n) = 21T(\frac{n-3}{23}) + k_1 n + k_2$$
$$\implies T(n) \leq 21(k_5 \frac{n-3}{23} + k_6) + k_1 n + k_2$$
$$\implies T(n) \leq 21k_5 \frac{n-3}{23} + 21k_6 + k_1 n + k_2$$
$$\implies T(n) \leq 21k_5 \frac{n}{23} - 21k_5 \frac{3}{23} + 21k_6 + k_1 n + k_2$$

Therefore,

$$T(n) \in O(n)$$

(b) Proof of $T(n) \in \Omega(n)$

$$T(n) = 21T(\frac{n-3}{23}) + k_3 n + k_4$$
$$\implies T(n) \geq 21(k_7 \frac{n-3}{23} + k_8) + k_3 n + k_4$$
$$\implies T(n) \geq 21k_7 \frac{n-3}{23} + 21k_8 + k_3 n + k_4$$
$$\implies T(n) \geq 21k_7 \frac{n}{23} - 21k_7 \frac{3}{23} + 21k_8 + k_3 n + k_4$$

Therefore,

$$T(n) \in \Omega(n)$$

(c) Since $T(n) \in O(n)$ and $T(n) \in \Omega(n)$,

$$T(n) \in \Theta(n)$$

3. Given the recurrence, we can interpret it to mean:

$$T(n) \leq 2T(\frac{n}{2}) + k_1\sqrt{n} + k_2$$
$$T(n) \geq 2T(\frac{n}{2}) + k_3\sqrt{n} + k_4$$

Claim: $T(n) \in \Theta(n)$

The proof is by strong induction on n.

Base case:
$T(1)$ is some constant, so it is in $\Theta(1)$. Since $n = 1$, the claim holds for the base case.

Inductive Hypothesis:
$T(x) \in \Theta(x)$ for all $x < n$.
$T(x) \leq k_5x + k_6$ for all $x < n$.
$T(x) \geq k_7x + k_8$ for all $x < n$.

Inductive Step:

(a) Proof of $T(n) \in O(n)$

$$T(n) \leq 2T(\frac{n}{2}) + k_1\sqrt{n} + k_2$$
$$\implies T(n) \leq 2(k_5\frac{n}{2} + k_6) + k_1\sqrt{n} + k_2$$
$$\implies T(n) \leq 2k_5\frac{n}{2} + 2k_6 + k_1\sqrt{n} + k_2$$
$$\implies T(n) \leq k_5n + 2k_6 + k_1\sqrt{n} + k_2$$

Therefore,

$$T(n) \in O(n)$$

(b) Proof of $T(n) \in \Omega(n)$

$$T(n) \geq 2T(\frac{n}{2}) + k_3\sqrt{n} + k_4$$
$$\implies T(n) \geq 2k_7\frac{n}{2} + 2k_8 + k_3\sqrt{n} + k_4$$
$$\implies T(n) \geq k_7n + 2k_8 + k_3\sqrt{n} + k_4$$

Therefore,

$$T(n) \in \Omega(n)$$

(c) Since $T(n) \in O(n)$ and $T(n) \in \Omega(n)$,

$$T(n) \in \Theta(n)$$

4. Given the recurrence, we can interpret it to mean:

$$T(n) \leq \sqrt{n}T(\sqrt{n}) + k_1 n^2 + k_2$$
$$T(n) \geq \sqrt{n}T(\sqrt{n}) + k_3 n^2 + k_4$$

Claim: $T(n) \in \Theta(n^2)$

The proof is by strong induction on n.

Base case:
$T(1)$ is some constant, so it is in $\Theta(1)$. Since $n = 1$, the claim holds for the base case.

Inductive Hypothesis:
$T(x) \in \Theta(x^2)$ for all $x < n$.
$T(x) \leq k_5 x^2 + k_6$ for all $x < n$.
$T(x) \geq k_7 x^2 + k_8$ for all $x < n$.

Inductive Step:

(a) Proof of $T(n) \in O(n^2)$

$$T(n) \leq \sqrt{n}T(\sqrt{n}) + k_1 x^2 + k_2$$
$$\implies T(n) \leq \sqrt{n}(k_5\sqrt{n}^2 + k_6) + k_1 x^2 + k_2$$
$$\implies T(n) \leq \sqrt{n}(k_5 n + k_6) + k_1 x^2 + k_2$$

Therefore,

$$T(n) \in O(n^2)$$

(b) Proof of $T(n) \in \Omega(n^2)$

$$T(n) \geq \sqrt{n}T(\sqrt{n}) + k_3 x^2 + k_4$$
$$\implies T(n) \geq \sqrt{n}(k_7\sqrt{n}^2 + k_8) + k_3 x^2 + k_4$$
$$\implies T(n) \geq \sqrt{n}(k_7 n + k_8) + k_3 x^2 + k_4$$

Therefore,

$$T(n) \in \Omega(n^2)$$

(c) Since $T(n) \in O(n^2)$ and $T(n) \in \Omega(n^2)$,

$$T(n) \in \Theta(n^2)$$

2: Task 5.2

The 2 parallel recursive calls of $|S|/2$ size each are responsible for the first term in the recurrence. The divide step has work of $O(n)$, so it's responsible for the 2nd term in the recurrence. The combine step has work of $O(n)$ for the following reasons. The only functions I call are `map`, `merge`, and `scanI`. All of the `map` calls use a constant time function on a linear input. The merge function has linear work. The scanI function uses a constant time function, so it too is linear in work. Therefore the skyline helper function conforms to the recurrence. The wrapper around skyline

which gets rid of redundant points, does so in linear work as well, since it uses a tabulate, filter, and map with a constant time functions on linear inputs.

`Proof of closed form of recurrence:`

$$W_{skyline}(n) = 2W_{skyline}(\frac{n}{2}) + O(n) + W_{combine}$$
$$\implies W_{skyline}(n) = 2W_{skyline}(\frac{n}{2}) + O(n) + O(n)$$
$$\implies W_{skyline}(n) = 2W_{skyline}(\frac{n}{2}) + k_1 n + k_2$$

This is the mergesort recurrence. But I'll prove that $W(n) \in O(n\log(n))$ anyway. The proof is by induction. Base case: When the length of the sequence is 0 or 1, the correct answer is returned in constant time, which is in $O(n\log(n))$.

Inductive Hypothesis:

$$W_{skyline}(x) \leq k_3 x\log(x) + k_4$$

for natural numbers x less than n.

Inductive Step:

$$W_{skyline}(n) = 2W_{skyline}(\frac{n}{2}) + k_1 n + k_2$$
$$\implies W_{skyline}(n) \leq 2(k_3\frac{x}{2}\log(\frac{x}{2}) + k_4) + k_1 n + k_2$$
$$\implies W_{skyline}(n) \leq k_3 x\log(\frac{x}{2}) + \frac{k_4}{2} + k_1 n + k_2$$
$$\implies W_{skyline}(n) \leq k_3 x\log(\frac{x}{2}) - k_3 x\log(2) + \frac{k_4}{2} + k_1 n + k_2$$
$$\implies W_{skyline}(n) \in O(n\log(n))$$

3: Task 5.3

A building is defined as (l, h, r) where $0 \leq l < r$ and $h > 0$.

Given a sequence of "buildings", the skyline function returns a sequence of "points" sorted by x values, where the x-values correspond exactly to the l and r values of buildings, and the y value is the max height of the buildings which satisfy the condition $l \leq x < r$.

The proof is by structural induction.

Base case:
case of `Seq.empty`:
In this case, the empty sequence is returned, which is the correct output by the definition of the skyline.

case of `Seq.singleton l,h,r`:
In this case, the sequence of $<$`(l,h), (r,0)` $>$ is returned, which is correct.

Inductive Hypothesis:
We assume that skyline is correct for all sequences of length up to $n-1$.

Inductive Step: Consider sequence S where $|S| > 1$. Lets call the values of l and r of all of the elements in S, the set of "x values of S". The problem states that we can assume that the x values of S are unique.

We can partition the sequence of buildings S into $s1$ and $s2$, where each have at most $|S|/2$ elements. We call the skyline function on each, and since $|S|/2 < |S|$, the IH applies. Therefore the results of the skyline calls are correct skyline outputs. Let `skyA = skyline s1` and `skyB = skyline s2`.

By the definition of skyline, `skyA` contains all of the x-values of $s1$, and `skyB` contains all of the x-values of $s2$. Thanks to the IH, the tuples in `skyA` and `skyB` are sorted by x-values. If we merge the x values of $s2$ into `skyA`, the resulting sequence would have all of the x values of S, since the x-values of `skyA` and `skyB` partition the set of x values of S. We merge the x values of $s2$ into `skyA`, and $s1$ into `skyB`, such that each entry from $s2$ in `skyA` will have `NONE` as its corresponding y value, and each each entry in `skyA` not from $s2$ will have `SOME y` as its corresponding y value. Likewise for `skyB` and $s1$. Then for `skyA` and `skyB`, we can perform a copy scan to propagate the height of the skyline at the newly added points to those points– replacing `NONE` with the closest `SOME` on the left. The result, is the same skyline as returned by the skyline function, but with added points that add resolution to the skyline.

Since we merged two sorted sets (`skyA` and x values of $s2$), the result is sorted. The result also contains all of the x-values in S. Both `skyA` and `skyB` do. Therefore, they each have the same length at this point ($|S|$), and tuples at the same indices have the same x-values (since they are distinct and sorted in both $skyA$ and $skyB$). The y-values of the tuples represent the height of the skyline.

Finally we can do a `map2` to return a sequence of points which have the max height of `skyA[i]` and `skyB[i]` for all i.

Since we have the max heights over all the x values in the input sequence, the skyline function is correct in the inductive step.