15-210 Assignment 01

Karan Sikka
ksikka@cmu.edu
Section C
September 4, 2012

1: Task 2.1

A dog wearing sunglasses on a mountain with a low cloud. There is an icepick in the ground.

2: Task 2.2

1. They have not broken the policy since notes were not taken, and they waited 4 hours before doing homework. They would have broken course policy if they shared their code during the conversation or if they discussed the solution, took notes, or used the knowledge to do homework within 4 hours of the conversation.
2. The course policy would have been broken since the student took notes. If he hadn't taken notes, and if he waited 4 hours before doing the relevant work, he would not have broken the course policy.
3. Good thing Yelena used toy examples, because this is accepted in the course policy. If Yelena and Abida worked on actual code for the homework, this would be against the course policy.

3: Task 2.3

25 hours is considered 2 days late, which is 25% off your grade twice. If you had a perfect score on your homework (100%), you would have a 56.25% after 25 hours. 56.25% of the total points on the assignment is your highest possible grade.

4: Task 3.2

Work:

$$W_{parenDist}(|S|) = W_{length}(|S|) + W_{flatten}(|S|) + W_{tabulate}(|S|) + W_{map}(|S|^2) \\ + W_{filter}(|S|^2) + W_{map}(|S|^2) + W_{reduce}(|S|^2)$$

$\Longrightarrow$

$$W_{parenDist}(|S|) = O(1) + O(|S| + \sum_{e \in S} |e|) + O(n^2) + O(\sum_{e \in subseq\_indexes} W(subseq(e))) \\ + O(\sum_{e \in subseqs} W(match(e))) + O(\sum_{e \in subseqs} W(length(e))) + O(n^2)$$

$\Longrightarrow$

$$W_{parenDist}(|S|) = O(1) + O(n^2) + O(n^2) + O(n^3) + O(n^3) + O(n^2) + O(n^2)$$

$$\implies$$

$$W_{parenDist}(|S|) = O(n^3)$$

This makes intuitive sense, since there are a quadratic number of subsequences generated, and `match` does linear work on each.

Span:

$$S_{parenDist}(|S|) = S_{length}(|S|) + S_{flatten}(|S|) + S_{tabulate}(|S|) + S_{map}(|S|^2)$$
$$+S_{filter}(|S|^2) + S_{map}(|S|^2) + S_{reduce}(|S|^2)$$

$$\implies$$

$$S_{parenDist}(|S|) = O(1) + O(\log(|S|)) + O(\max_{i=0}^{n} S(tabulate(i))) + O(\max_{e \in subseq\_indexes} S(subseq(i)))$$

$$+O(\log(|subseqs| + \max_{e \in subseqs} S(match(e)))) + O(\max_{e \in subseqs} S(length(e))) + O(\log(|subseqs|))$$

$$+O(\max_{e \in lengths} S(Int.max(e)))$$

$$\implies$$

$$S_{parenDist}(|S|) = O(1) + O(\log(n)) + O(1) + O(1) + O[\log(n^2) + \log^2(n)] + O(1) + O(\log(n^2))$$

$$\implies$$

$$S_{parenDist}(|S|) = O(\log^2(n))$$

This makes intuitive sense, since brute force is generally an extremely highly parallelizable algorithm and therefore, the span is determined by `match`, the divide and conquer step.

5: Task 3.4

1. We know from Recitation 1 that $W(n) \in O(n)$. However, if this is not sufficient, the proof is reproduced below:

   Recurrence:

   $$W(n) = 2W(\frac{n}{2}) + c\lg(n) + k$$

   for n greater than some n_0, and for some constants c, k.

   Using the tree method, we see that

   $$W(n) \leq \sum_{i=1}^{\lg(n)} 2^i (c\lg(\frac{n}{2^i}) + k)$$

   Then we do some algebra:

   $$W(n) \leq \sum_{i=1}^{\lg(n)} 2^i (c(\lg(n) - i) + k)$$

$$W(n) \leq \sum_{i=1}^{\lg(n)} c\lg(n)2^i + k2^i - ci2^i)$$

$$W(n) \leq (c\lg(n) + k)\sum_{i=1}^{\lg(n)} 2^i - c\sum_{i=1}^{\lg(n)} i2^i)$$

Then we use Lemma 1 and Lemma 2 from Recitation 1.

$$W(n) \leq (c\lg(n) + k)(2n-1) - c(2n\lg(n) - (2n-2)))$$

$$W(n) \leq 2cn\lg(n) - c\lg(n) - 2cn\lg(n) + 2cn - 2c + 2kn - k$$

$$W(n) \leq (c+k)(2n-1) - c\lg(n) - c$$

Therefore, $W(n) \in O(n)$.

2. Recurrence:

$$W(n) = 2W(\frac{n}{2}) + cn + k$$

We can expand this recurrence to a tree-like structure where each leaf at height h has a value of $\frac{cn}{2^h} + k$, and there are 2^h leafs at each row. We can express the sum of the nodes in the tree as the sum of the values in each row, which turns out to be constant, times the number of rows. We get the following summation:

$$W(n) \leq \sum_{i=1}^{\lg(n)} cn + 2^i k$$

for n greater than some n_0, and for some constants c, k. Then we do some algebra:

$$W(n) \leq cn\lg(n) + k\sum_{i=1}^{\lg(n)} 2^i$$

By Lemma 1 from recitation:

$$W(n) \leq cn\lg(n) + k(2n-1)$$

Therefore, $W(n) \in O(n\lg(n))$.

3. A balanced tree data structure would make `showt` a constant-time operation. In this data structure, a Tree would be defined as a Leaf, a Node, or null. A Leaf would contain a single element of data, and a Node would contain references to two trees. The function `showt` would return the 2 children of the root in the non-trivial case, a constant work operation.

$$W(n) \in O(1) \subseteq O(lg(n))$$

4. An array data structure could be implemented to make `showt` run in $O(n)$ time. If the length of the array is 0, it returns `NONE`. In the non-trivial cases, it would return two arrays: one of the first half of the original array and one of the second half. Building these arrays takes $O(n)$ work.

6: Task 4.1

The ordering is 6, 3, 5, 1, 2, 4, 7.

7: Task 4.2

1. Proof:
   Let $f \in O(g)$ and $g \in O(h)$. More formally:

$$g(n) \leq c_0 h(n) \qquad \text{for all } n > n_0, \text{for some } c_0 \tag{1}$$
$$f(n) \leq c_1 g(n) \qquad \text{for all } n > n_1, \text{and for some } c_1 \tag{2}$$

   Then we can use (1) to say:

$$c_1 g(n) \leq c_1 c_0 h(n) \qquad \text{for all } n > n_0$$

   And by (2),

$$f(n) \leq c_1 g \leq c_1 c_0 h(n)$$
$$f(n) \leq c_1 c_0 h(n) \qquad \text{for all } n > n_2$$

   If we let $c_2 = c_1 c_0$ and $n_2 = max(n_0, n_1)$, then we have proved that

$$f \in O(h)$$

   which proves that O is a transitive relation on functions.

2. This is false. Counterexample:
   Let $f(n) = 1$ and $g(n) = n$. We see that $f \in O(g)$, where $n_0 = 1, c_0 = 2$. However, $g \notin O(f)$.

   Formal proof (just in case the above was not sufficient):
   It is clear that $g(n) = n \in O(n)$, and $f \in O(1) \subseteq O(n)$. However you cannot say that $g(n) \in O(f)$ since that implies that $g(n) = n \in O(1)$, a false statement since n, the input, can always be one larger than any constant.

3. False. Counterexample: Let $f(n) = 1, g(n) = 2$. Then $f, g \in O(1)$, but $f \neq g$.

8: Task 4.3

$$S = \sum_{i=0}^{n} a^{2i}$$
$$\implies S = 1 + \sum_{i=1}^{n} a^{2i}$$

Now factor out a^2 from every term in the summation and adjust the indexes:

$$\begin{aligned}
&\implies S = 1 + a^2 \sum_{i=0}^{n-1} a^{2i} \\
&\implies S = 1 - a^{2n+2} + a^2 \sum_{i=0}^{n} a^{2i} \\
&\implies S = 1 - a^{2n+2} + a^2 S \\
&\implies S - a^2 S = 1 - a^{2n+2} \\
&\implies S(1 - a^2) = 1 - a^{2n+2} \\
&\implies S = \frac{1 - a^{2n+2}}{1 - a^2} \\
&\implies S = \frac{a^{2n+2} - 1}{a^2 - 1}
\end{aligned}$$