15-210 Assignment Parenlab

Roy Sung
roysung@andrew.cmu.edu
Section E
01/27/14

4: The Parenthesis Distance Problem

Task 4.3
Work: So the brute force algorithm first takes all contiguous substrings of the string. So parenDist first uses the function contiguousElem which goes and grabs all contiguous substrings. If We assume subseq to take only $O(1)$ then contiguous substring will take $O(n^2)$. Then it takes the sequence of all contiguous substrings and filters out results that don't start with a open parenthesis and a close parenthesis. Since there are $O(n^2)$ elements, filter's work is the sum of all the work done by the function passed onto filter. This function is realPmatch, it's work is also O(1). So this means that filter done on $O(n^2)$ elements will take $O(n^2)$. We go through again and use filter again with parensMatchSeq. parensMatchSeq has a work of linear time and this is done on $O(n^2)$ elements which in this means that this call to filter will be have work of $O(n^3)$. We find the biggest one with map and a function that is passes has work of constant time (since length has work done in constant time). Thus this call to map has work of $O(n^2)$. After which we just find the largest substring with iter and this also has work of $O(n^2)$. So if we add all this up, we find that work done by parenDist is $O(n^3)$.

Span: The expanation should not be different but all the time complexity should be different. So if we look we see that we only used the function filter, map. The span of filter is the max span done by the function and this is the same with map as well. So when we take all contiguous strings which has the span of $O(n^2)$ since there are $O(n^2)$ elements. When filter is done the first time it has a span of constant time. The second time filter is done the function passed is parensMatchSeq which has a span of $O(log^2 n)$. Then map is done which is also has a span of constant time since the function passed is has a span of constant time. iter also has constant time for the same reason. So we if add all these up we see that that span will be $O(n^2)$. This is because we still have to get all contiguous substrings. So teh span of parenDist is $O(n^2)$.

Task 4.6

1. So we have that $W_{showt} \in O(logn)$ since $W_{showt}(n) \in \theta(logn)$. So we get
   $W(n) \leq 2 \cdot W(n/2) + c_1 logn + c_2 + c_3$.
   $W(n/2) = 2 \cdot W_{n/4} + W_{showt}(n/2) + O(1) \leq 2W(n/4) + c_4 log(n/2) + c_5 + c_6$. When we break it down more, it will make a tree with $logn$ levels and at each level the work added up will be $2^k \cdot W_{showt}(n/k) + O(1)$, where k is the level of the tree. So we have to sum up all the work done at each level essentially doing $\sum_{i=0}^{logn} 2^i \cdot log(n/2^i) = \sum_{i=0}^{logn} 2^i \cdot (logn - ilogn)$. If we take out the constants because they are trivial we get $\sum_{i=0}^{logn} 2^i logn - 2^i$. If we reduce this (getting the closed form from lecture notes) we get that it is $(2n-1)logn - 2nlogn - (2n-2)$. Which we can simplify and say that the final form is an element of $O(n)$. So that is the upper bound for the work if we assume that $W_{showt}(n) \in O(logn)$. This also means that the lower bound is also linear because we are given that $W_{showt}(n) \in \theta(logn)$. We can do

this out formally by going through the whole substitution part but with just saying that $W(n) \geq c \cdot W(n/2) + c \cdot W_{showt}(n) + c \cdot O(1)$ for some constant c..

2. So we have that $W_{showt} \in \Theta(n)$.
   So using the same process as before we split the work go down the tree with $W(n/2)$ and then this will go and give us $logn$ levels with each level doing $k \cdot n$ work, with k being the level of the tree. If we add up all the work done at each level it will give us $2^k \cdot n/k = n$ amount of work at the kth level. Since there are $logn$ levels we just do $n \cdot logn$ to get the total amount of work done. This will give us that $W(n) \in O(nlogn)$. So the upper bound is $nlogn$ and for the lower bound same thing as above but we just switch the inequality sign which we can do because we are given that $W_{showt} = \theta(n)$ and we get that the lower bound is n which then tells us that $W(n) \in \theta(nlogn)$.

3. A list where when showt is called upon it, it turns the list into a balanced tree. So that way showt has to create logn levels for the tree created by the list. So when showt is called it creates two actual tree structures with logn levels, that way showt has to have time complexity of logn.

4. A linked list with the pointer node always starts at the beginning of the list. This way each time the list has to go and pick an element from the elements that is stored within it, it has to traverse the list to get to that element and then spit it back out. So showt will give the first half of the list which will take linear time and then the second half of the list which will also take linear time to take from the list.