15-210 Assignment Skyline Lab

Roy Sung
roysung@andrew.cmu.edu
Section E
2/3/2014

4: The Pittsburgh Skyline Problem

Task 4.3
Let $P(s)$ = c̈omputeskyline s solves the Skyline problem ¨
We will do structural induction on s.

1. $s =<>$
   If s is the empty sequence then this is the empty case in which it will return $(0, 0)$ signaling that there is no outline which is the correct output for an empty sequence. Thus the empty case holds.

2. For all $b = (l, h, r)$, b representing a single building $s =< p >$
   If s is a singleton sequence then this is the case of ELT(p). In which case it will return $< (l, h), (r, 0) >$, which is the current output to represent a single building, thus the singleton case holds.

3. Let L_1, R_2 be the two sequences returned by the left and right recursive call and assume that they are both valid sequences is in they return the correct output.
   With the two valid sequences we go through and label the two sequence so that we can tell the difference between where the left points came from and where the right points came from. Then we go and mergesort the two together in a order by the x-coordinate of the point. So now we have just a single sequence with both the points from the left and right and we are able to tell which sequence the points originated from. Next we go through and make the new sequence valid by taking away points or creating new points. First we notice that we compare points that are adjacent to each other and not just right next to each other but we compare points from different sequences that are "adjacent" to each other. We do this because comparing points between points that originated from the same sequence is trivial as by our IH we can assume that points from the same sequence create a valid sequence. Also mergesorting together the two sequences will not change the ordering of any of the points relative to theri own repsective sequences. This means that the ith $i+1th$ point from the left sequence WLOG, the ith point will still be ahead of the $i + 1th$ point in the mergedsorted sequences. So lets start with a left point WLOG. This left point will be compared with a right point that came before it on the sequence to either keep,take away, or modify the point.There are two cases for the left point comparing the the latest right point. Let h_1 be the height of the left point and h_2 be the height of the right point

   - if $h_1 > h_2$
     then we keep the left point as this means that the left point is at a higher level than the right point thus it is a valid point in the skyline problem output.
   - if $h_1 \leq h_2$.
     if this case happens that one of two things could happen. We have to check with a left

point that came before the current left point, but after the right comparison point. If there no such point exists then the current left point is not a valid point and we take it away from the sequence, as the right point is clearly the higher point in the placement of the current left point. If there exists some such point, we denote the height of this point with h_3. If $h_3 \leq h_2$ then we can the current left point as it is a redundant point and in our skyline problem we are told to take out these redundant points. If $h_3 > h_2$, then we take the current left point and we give it a height of h_2, this is because since there was a change of height of h_3 to h_1 in the right sequence of buildings at the left point then to make it a valid point at the current left point with the addition of the right buildings h_2 becomes the new max height at the current left point. Thus we change the current left point to make it a valid point in the new sequence.

We do this for all the points in the sequence giving us a new valid sequence that is a combination of the two. Thus since we have shown that the sequence made up of the left and right sequence is a valid sequence.

Since we have shown that the base case and the induction step holds we have proven for all sequence of building inputs.

Task 4.4

So if we go through our code we the base cases (empty and singletons) clearly show that the work is done in constant work compared to $|S| = n$.
So we take the left side and the right side which we assume split the given input sequence in about half. Then we recursively call the function on theses sequences. This would be the $2W(\frac{n}{2})$ in our work recurrence. Once we get back the results. We take the two sequence and we create two new sequences by copying the two sequencs but we add in the information of whether it came from the left side and right side. We do tabulate with length $\frac{n}{2}$ twice which in the end will just mean we have done $O(n)$ work. Then we merge them together with the function merge which has $O(n)$ work since our compare function has constant work. Then call scan on the the newly formed sequence twice which then will still be $O(n)$ work since our compare function that we passed is constant. The final two step is that we we first tabulate and create a new sequence that is has length n and then we use filter on this newly formed sequence. Which in turn again these two function will give us $O(n)$ work twice. If we add up all the work done, we can see that we only used function with $O(n)$ work thus the combine step or $W_{combine}(n) \in O(n)$. So our recurrence looks like

$$W(n) = 2W(\frac{n}{2}) + O(1) + O(n)$$

We already proved in the last homework that when solved says that $W(n) \in O\left(nlog(n)\right)$. Thus we have shown that our algorithm has $O(nlogn)$ work.