CS4800 - ALGORITHMS - S'18 — NGUYEN
DUE FRI FEB 2, 2018 AT 1:35PM VIA GRADESCOPE

H3.

- The assignment is due at Gradescope on February 2 at 1:35pm. Late assignments will not be accepted. Submit early and often.
- You are permitted to study with friends and discuss the problems; however, *you must write up you own solutions, in your own words*. Do not submit anything you cannot explain. If you do collaborate with any of the other students on any problem, please do list all your collaborators in your submission for each problem.
- Finding solutions to homework problems on the web, or by asking students not enrolled in the class is strictly prohibited.
- We require that all homework submissions are prepared in Latex. If you need to draw any diagrams, however, you may draw them with your hand.

PROBLEM 1 *Why 5 in Median?*

Recall the deterministic selection algorithm:

```
1:  procedure SELECT($A[1,\dots,n],i)$)
2:      $k \leftarrow 5$
3:      if $n \leq 3k$ then
4:          use brute force (sort and pluck)
5:      else
6:          $m \leftarrow \lfloor n/k \rfloor$
7:          for $i \leftarrow 1$ to $m$ do
8:              $M[i] \leftarrow$SELECT($A[k\cdot(i-1)+1,\dots,k\cdot i], \lceil k/2 \rceil$)      ▷ Find medians among groups of size $k$
9:          end for
10:         $MoM \leftarrow$SELECT($M[1\dots,m], \lceil m/2 \rceil$)      ▷ Recursion
11:         ▷ Shuffle $A[]$ so that $MoM$ is at position $r$, $A[1,\dots,r-1] \leq MoM$ and $MoM \leq A[r+1,\dots,n]$
12:         $r \leftarrow$PARTITION($A[1,\dots,n], MoM$)
13:         if $r = i$ then
14:             return $MoM$
15:         else if $r < i$ then
16:             return SELECT($A[r+1,\dots,n], i-r$)      ▷ Recursion
17:         else
18:             return SELECT($A[1,\dots,r-1], i$)      ▷ Recursion
19:         end if
20:     end if
21: end procedure
```

(a) In class, we showed that with $k = 5$, the value of r returned by PARTITION($A[1,\dots,n], MoM$) is at least $3n/10 - 3/2$ and at most $7n/10 + 5/2$. Suppose that instead we use $k = 3$. State an upper and lower bound on the value of r returned by PARTITION($A[1,\dots,n], MoM$). Be as precise as you can.

Solution:

(b) Write a recurrence for $S(n)$, the running time of SELECT($A[1,\dots,n], i$), using the bounds on r you got for $k = 3$.

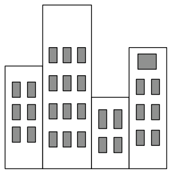
Solution:

(c) Use induction to prove $S(n) = \Omega(n \log n)$ for $k = 3$. Ignore all ceiling/floor for simplicity.

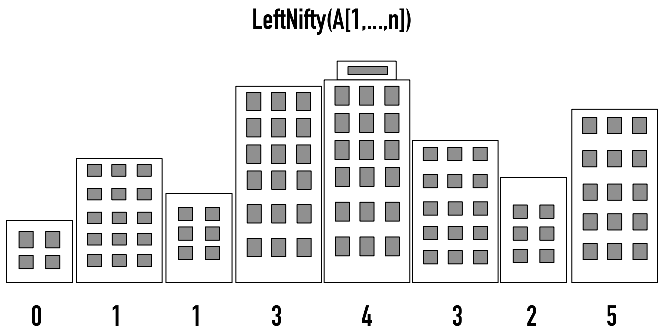
Solution:

PROBLEM 2 *Skyline*

A great skyline must have variations. Define the left skyline function, denoted $n_\ell(s)$, of a skyline s as the total number of times that a building is taller than one of its left neighbors. The right skyline function, $n_r(\cdot)$, is defined analogously. The skyline s below has 4 buildings, and $n_\ell(s) = 3$ and $n_r(s) = 3$ because building 2 is taller than building 1 and building 4 is taller than buildings 3 and 1, and alternatively, building 1 is taller than 3, and building 2 is taller than 3 and 4.



As another example, this skyline has $n_\ell = 19$, the contribution of each building is listed below the building.



We will design and analyze a divide and conquer algorithm that computes the right and left skyline functions of a skyline s with n buildings. The input $A[1, \ldots, n]$ consists of the heights of each building along a street in left to right order; assume all buildings have unique heights.

(a) The left skyline function is the sum of 3 terms: the score of the left half, the score of the right half, and the number of pairs of buildings, one on the left half and one on the right half such that the right one is taller than the left one. In divide and conquer algorithms, we can recursively compute the first two terms so we just need to focus on the third term. Describe how you can compute the third term in $O(n^2)$ time.

Solution:

(b) We need to compute the third term faster. First we sort the buildings on each half in *increasing* heights. Then we merge the two sorted lists into one as in mergesort. If a building B is ranked u in the sorted list of buildings in the right half and ranked v in the sorted list of all buildings, how many buildings are there in the left half that are shorter than B?

Solution:

(c) Design a function LeftSkyline($A[1 \ldots n]$) that returns the sorted list A' (A sorted in increasing heights) and the left skyline function of A. Hint: the algorithm can recursively compute the first two terms and the sorted halves and then use merging to compute the third term and the big sorted list. Your time for computing the third term should be $O(n)$. Hint: you can use the following MergeSort pseudocode as starting point.

```
1: function MergeSort($A[1 \ldots n]$)
2:     if $n = 1$ then return $A$
```

```
    **else**
        $m \leftarrow \lfloor n/2 \rfloor$
        $A1 \leftarrow \textsc{MergeSort}(A[1 \dots m])$
        $A2 \leftarrow \textsc{MergeSort}(A[m+1 \dots n])$
        $A' \leftarrow ()$
        $i \leftarrow 1$
        $j \leftarrow 1$
        **while** $i \leq m$ and $j \leq n - m$ **do**
            **if** $A1[i] < A2[j]$ **then**
                Append $A1[i]$ to $A'$
                $i \leftarrow i + 1$
            **else**
                Append $A2[j]$ to $A'$
                $j \leftarrow j + 1$
            **end if**
        **end while**
        ▷ $A1[i \dots m]$ are larger than all elements in A2[]
        Append $A1[i \dots m]$ to $A'$
        ▷ $A2[j \dots n - m]$ are larger than all elements in A1[]
        Append $A2[j \dots n - m]$ to $A'$
        **return** $A'$
    **end if**
**end function**
```

Solution:

(d) Write a recurrence for the running time of your algorithm and solve it using the master theorem.

Solution:

PROBLEM 3 *Programming Skyline*

In this part, you will implement your algorithm for Skyline and test it on real data. Register and take on the challenge at `https://www.hackerrank.com/contests/cs4800-s18`. Please put here your username and score.

Solution: