CSC 226 - Algorithms and Data Structures II

Assignment 1

Ben Wolfe — V00205547

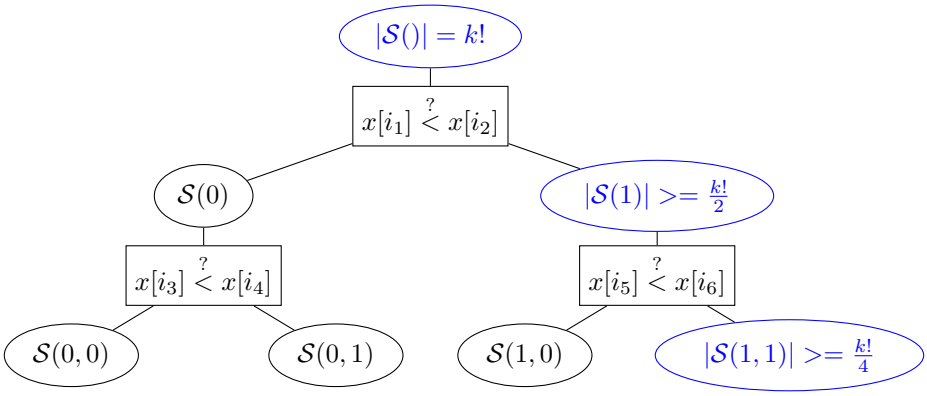
Problem 1

Consider an input of n elements. The input is broken up into $\frac{n}{k}$ groups therefore each group contains k elements:

$$(x_1...x_k), (x_{k+1}...x_{2k}), (x_{2k+1}...x_{3k}), ..., (x_{n-k}...x_n)$$

The arrangement of groups is already established so we only need to consider sorting each of the $\frac{n}{k}$ subgroups. Since each group contains k elements, there are $k!$ possible permutations of the correct order. Using a decision tree, we can determine the lower-bound on time complexitiy to sort each group.

Let $\mathcal{S}$ be a set representing all possible permutations.



Given any decision in the tree, the advisary will always select the branch with more permutations. The advisary's decisions are shown in blue. We can see that with each step we are reducing the number of possible permutations:

$$k! \rightarrow \frac{k!}{2} \rightarrow \frac{k!}{4} \rightarrow ... \rightarrow \frac{k!}{2^h}$$

In final term above, h represents the height of the tree (and the number of decisions made).
Let s represent the advisaries sequence of decisions until there is only one possible permutation left:

$$|\mathcal{S}(s)| = 1 \geq \frac{k!}{2^h}$$
$$2^h \geq k!$$
$$h \geq log(k!)$$

Now consider, $log(k!)$:

$$log(k!) = log(1) + log(2) + ... + log(k-1) + log(k)$$

Upper Bound

$$\begin{aligned} log(1) + log(2) + ... + log(k-1) + log(k) &\leq log(k) + log(k) + ... + log(k) \\ &= klog(k) \end{aligned}$$

Lower Bound

$$\begin{aligned} log(1) + log(2) + ... + log(k-1) + log(k) &\geq log(\frac{k}{2}) + ... + log(k) \\ log(k!) &\geq log(\frac{k}{2}) + ... + log(\frac{k}{2}) \\ &= \frac{k}{2}log(\frac{k}{2}) \end{aligned}$$

Using the lower bound determined above for $klog(k!)$, $h \geq log(k!) \geq \frac{k}{2}log(\frac{k}{2})$. This is equivalent to an asymptotic time complexity of $\Omega(klogk)$.

Since there are $\frac{n}{k}$ subgroups that need to be sorted, the time compexity of the entire operation is:

$$klog(k) \times \frac{n}{k} = nlog(k) \ \square$$

Problem 2

There are four actions to consider when analyzing the time complexity of QuickSelect:

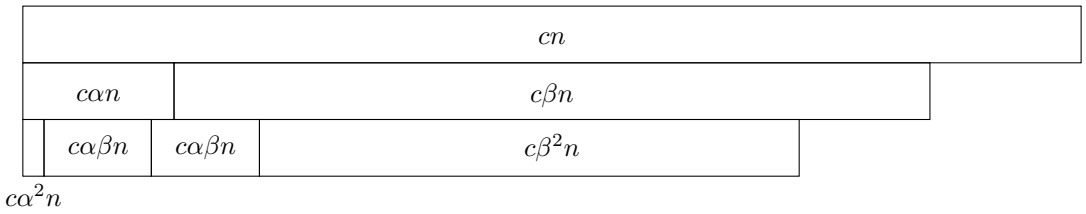
1. Splitting the input into subgroups groups of size 7 and sorting $O(n)$
2. Finding the median of subgroups $T(\frac{n}{7})$
3. Using the pivot to split on the appropriate pieces $O(n)$
4. Recursing on the appropriate piece *(Found Below)*

If there are $\frac{n}{7}$ groups then in at least $\lceil (n/7)/2 \rceil$ of them (those groups whose median $\leq$ to $m*$ where $m*$ is the median of medians) at least 4 elements are $\leq m*$. Therefore, the total number of elements $\leq m*$ is at least $4\lceil (n/7)/2 \rceil = \lceil 4n/14 \rceil$. By symmetry, the same number of elements $\geq m*$. As a result, in the **worst case**, there will be $\frac{10n}{14}$ elements for each block that we recurse on.

Therefore, the final equation for the recursion is:

$$T(n) = cn + T(n/7) + T(10n/14)$$

To analyze the recursive expression, we'll use a stack of bricks with $\alpha = \frac{1}{7}$ and $\beta = \frac{10}{14}$:

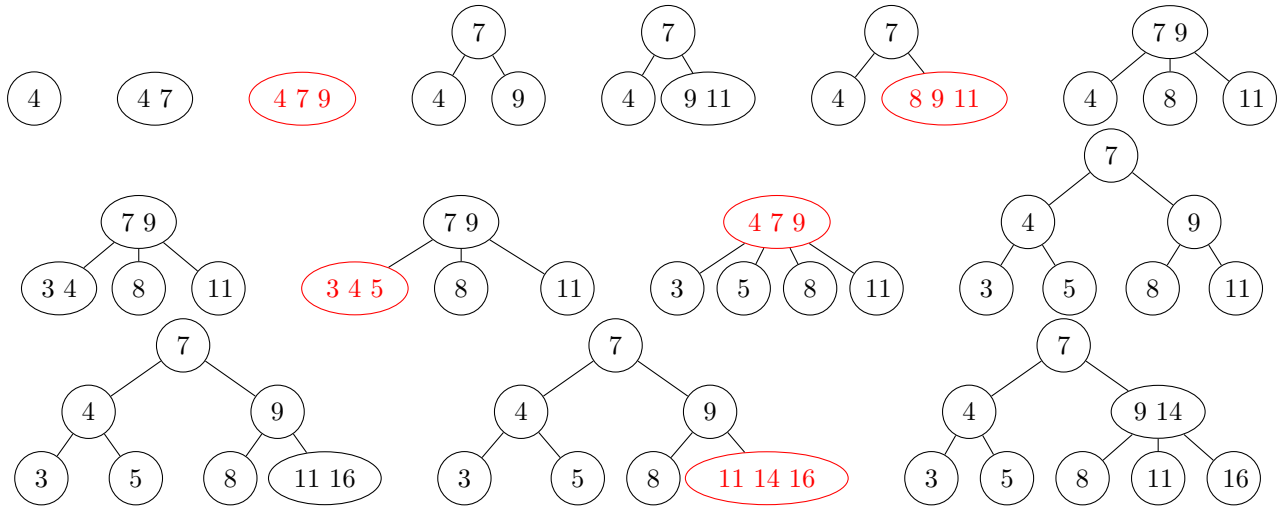


The total sum, and thus the worst case, will only ever be:

$$cn\sum_{i=0}^{\infty}(\alpha+\beta)^i = \frac{cn}{1-(\alpha+\beta)} = \frac{cn}{1-(\frac{1}{7}+\frac{10}{14})} = 7cn = O(n) \ \square$$

Problem 3

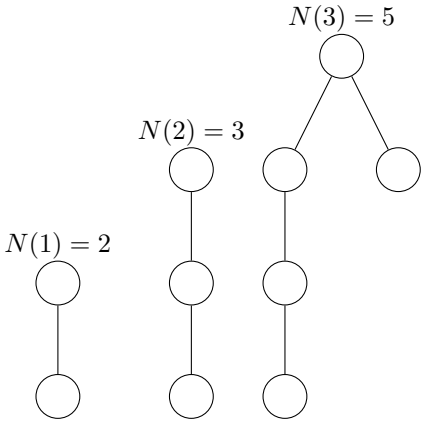
The following diagram shows the insertion sequence for elements 4, 7, 9, 11, 8, 3, 5, 16, 14 into a 2-3 tree. All stages of insertion are shown including temporary 4 nodes (represented in red).



Problem 4

Claim: $N(h) = \Omega(k^h)$ for some constant $k > 1$ using "relaxed" AVL trees.

Basis:



$N(1) = 2 \geq k^h$ if $log(2) \geq log(k)$
$N(2) = 3 \geq k^h$ if $log(3) \geq 2log(k)$
$N(3) = 5 \geq k^h$ if $log(5) \geq 3log(k)$

Referring to the trees above and because of the "relaxed" tree conditions, the trees above are represented by the recurisve expression: $N(h) = 1 + N(h-1) + N(h-3)$

Induction Hypothesis: Suppose that "relaxed" AVL trees have a minimum number of nodes, k^h, for some $h > 3$.

Induction Step:

$$\begin{aligned} N(h) &= 1 + N(h-1) + N(h-3) \\ &\geq 1 + ck^{h-1} + ck^{h-3} \ \textit{Induction Hypothesis} \end{aligned}$$

We wanted to show that $N(h) \geq k^h \therefore$ it sufficies to show the following:

$$\begin{aligned} 1 + k^{h-1} + k^{h-3} &\geq k^h \text{ Divide through by } k^{h-3} \\ \frac{1}{k^{h-3}} + k^2 + 1 &\geq k^3 \text{ *See note} \\ k^2 + 1 &\geq k^3 \end{aligned}$$

* - The first term is dropped since it becomes exponentially small as h increases.

The inequality will hold as long as k is less than the solution to $k^3 = k^2 + 1$. Using an polynomial solver, $k \approx 1.46$. Therefore, the claim holds.

Conclusion:
By proof of induction, a "relaxed" AVL tree has a minimum number of nodes represented by $N(h) = \Omega(k^h)$ $\square$