Aditya Shirode, Rhythm Shah, Shrenuj Gandhi

2. [6 points] *Quicksort.*

Recall that the most sophisticated Quicksort partitioning implementation used by Bentley and McIlroy, given pivot p, puts elements with keys $< p$ in the left part of the array, those with keys $= p$ in the middle and those with keys $> p$ to the right. Call this implementation Eq-Partition; it requires exactly $n - 1$ comparisons independent of input.

(a) Assuming that Eq-Partition is used for partitioning and no recursive calls are done for elements with keys equal to the pivot, what is the worst-case number of comparisons required to sort an array whose elements have keys that are entirely 0's and 1's? Your answer should give the *exact* number of comparisons in the worst case and describe what an array for the worst case looks like. Note that the algorithm is comparison-based, i.e., is unaware of the actual keys.

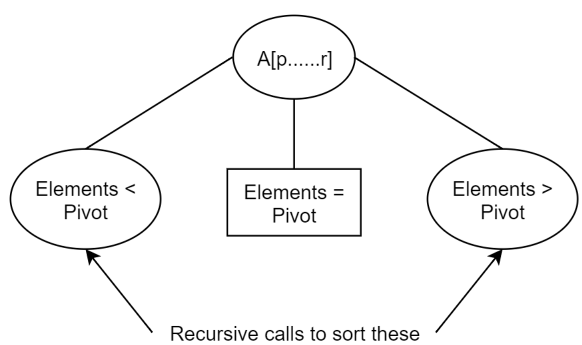
(b) Give a Θ bound for the worst case number of comparisons when the keys are from the set $\{1, \dots, k\}$. Again your bound should assume that Eq-Partition is used. Your bound should be in terms of both n and k. In this situation a Θ bound would be defined as:

> $f(n, k)$ is $\Theta(g(n, k))$ if there exist four positive constants c_1, c_2, n_0 and k_0 so that $c_1 g(n, k) \le f(n, k) \le c_2 g(n, k)$ for all $n \ge n_0$ and $k \ge k_0$.

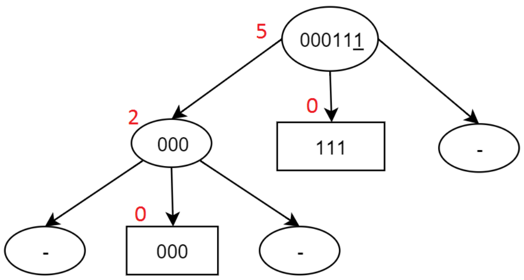
For the lower bound, you should describe what a general worst-case example looks like.

Answer:

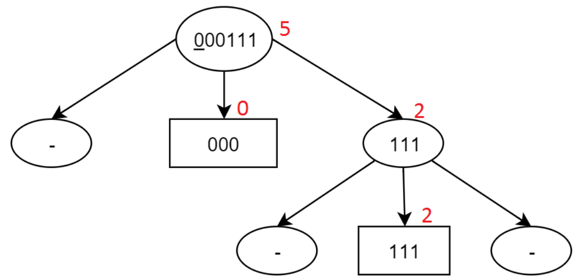
(a) The 3-way Quicksort partitioning implemented by *Bentley and McIlroy* can be represented as follows -



If we were sorting an array of 0's and 1's, we would have either 0 or 1 as a pivot, and would need to make a recursive call to sort 1s (in case pivot = 0) or 0s (in case pivot = 1).
This could be represented as :



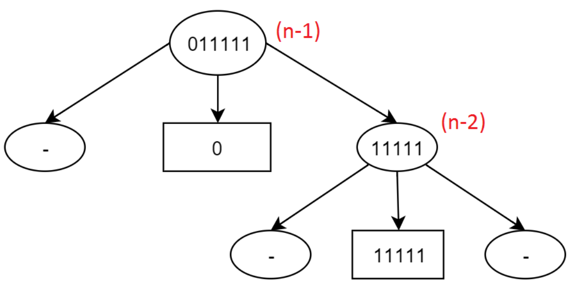
OR



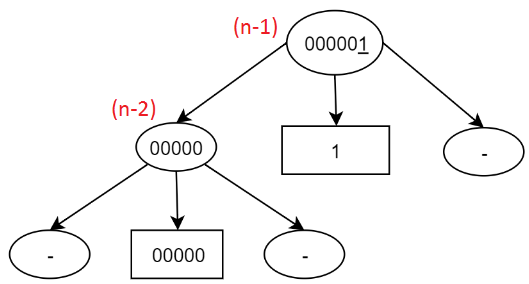
Thus we have $(n-1)$ comparisons initially, and $(n-k)$ comparisons in the subsequent stage, where k = either #0's or #1's, depending on the pivot.

To depict the worst case, we should maximize $(n-k)$. In other ords, after removing k pivot elements, the remaining array size should be maximum, which will yield maximum comparisons in the second stage. Thus, there should be $(n-1)$ remaining elements, which would give $(n-2)$ comparisons.

Hence, the total number of comparisons would be
$= (n-1) + (n-2)$
$= 2n - 3$ comparisons $\in \mathrm{O}(n)$



OR



The array will contain one element of one type (either 0 or 1) and (n-1) elements of other type.

(b) Given that the keys are from the set $\{1 \ldots k\}$, we can say that there would be k calls to the partition algorithm in the worst case. Since the partition algorithm makes $(n-1)$ comparisons, a good upper bound would be $(n-1)k = kn - k$.

$$\text{Thus } \; T(n,k) \in \mathrm{O}(nk) \tag{1}$$

Also, from the previous example (a), we can say that there would be $(n-1)$ comparisons at the first level, $(n-2)$ comparisons at the second level, $(n-3)$ comparisons at the third level, till $(n-k)$ comparisons at the k^{th} level.
Here, we assume that no element in array is equal to the pivot till the k^{th} level, and all other elements are either smaller (or bigger) than pivot at every stage, till k^{th} stage.

Thus, we have -

$$\begin{aligned}
T(n,k) &= \sum_{i=1}^{k}(n-i) \\
&= n\sum_{i=1}^{k}1 - \sum_{i=1}^{k}i \\
&= nk - \frac{k(k+1)}{2} \\
&= nk - \frac{k^2}{2} - \frac{k}{2}
\end{aligned}$$

$$\text{If } \; n \geq k, \text{ then - } \; T(n,k) \in \Omega(nk) \tag{2}$$

And in this case, the array will look like single occurences of elements from 1 to $(k-1)$, and $(n-k+1)$ occurences of k^{th} element.

Thus, from (1) and (2), we can say that - $T(n,k) \in \Theta(nk)$