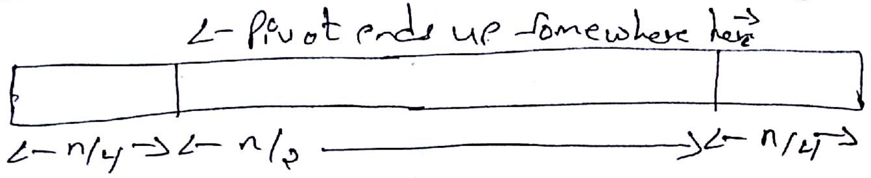
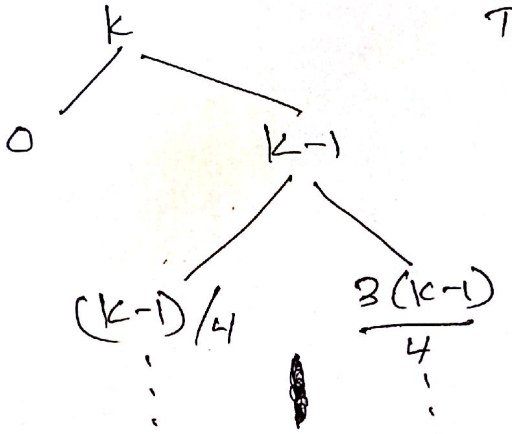
(4) Let us suppose our pivot could 1place end up in anywhere in n-element subarray after Partesioning.



<- Pivot ends up somewhere here ->
<- n/4 -><- n/2 -><- n/4 ->

It is likely that there is 50% of getting a bad split and the other half a good split.

So, The tree would look like with alternating good and bad splits:-



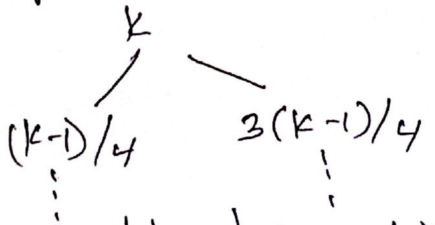
Total partisioning time:-

ck

c(k-1)

c(k-1)

For 100% good pivot calls it would be:-



Total Partisioning time:-

ck

c(k-1)

So, ~~ana~~ analysing both tree we can find that, even if we got the worst case split half of the time and good case split the other half, the running time would be about twice the running time of good case split.

And since that is just a constant factor, we can write the running time as ~~O(nlgn)~~ O(nlogn).