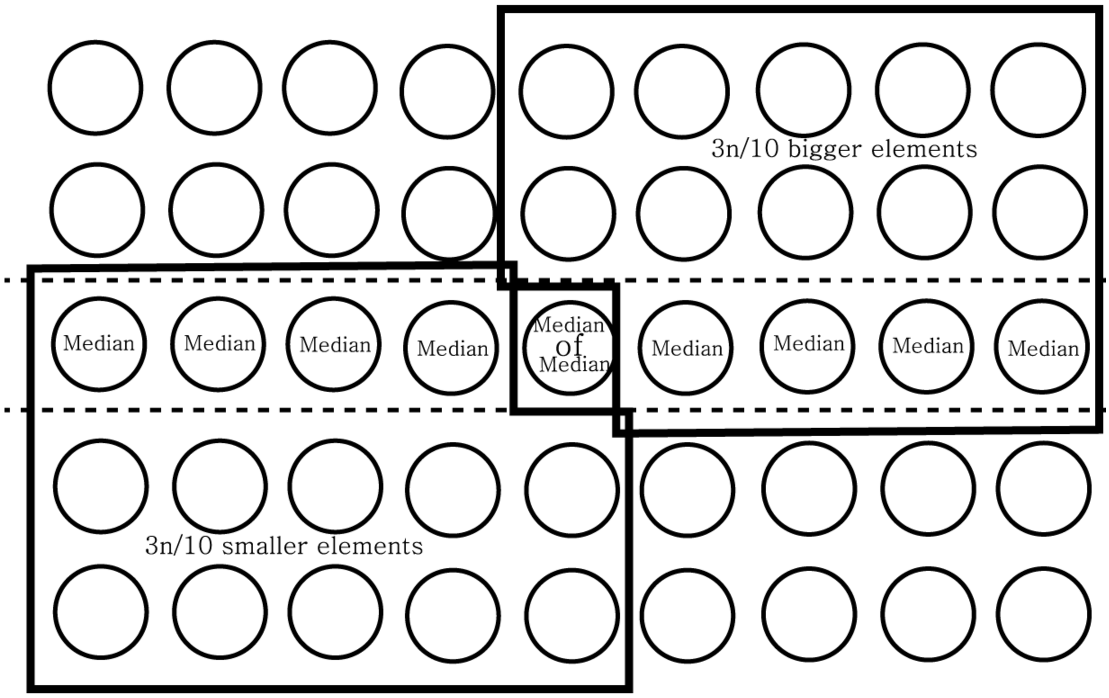
Porf. Helmut Alt
Team 31. 20140174 Hyungrok Kim
20150710 Kanghee Cho

1. (a) Algorithm to find the median in linear time is as follows.
assume the time that takes for this is T(n).
first, divide by the appropriate size. (generally, the size is five.)
here, it takes O(n) long.
Second, find the median for each group. this takes O(n) too because it takes constant time for each 5 sized array and there are n/5 of them.
Third, Compare the values of the medians thus found to find the median.
this takes T(n/5).



Use this median, divide to two parts, smaller elements and bigger elements. like the picture above, we already have 3n/10 smaller elements and 3n/10 bigger elements. After dividing rest of elements, assume pivot P's index is i and median's index k
if i=k, return P
if i>k, find kth element in partition of smaller elements.
if i<k, find k-ith element in partition of larger elements.(here and above, execute same deterministic selection algorithm recursively.

by the regulation from the picture, a partition array's Min size is $3n/10$, Max size is $7n/10$ for each group, L and H.
This proves that

$$T(n) \leq cn + T(n/5) + T(7n/10)$$

from here, by induction,
assume that

$$T(n) = O(n)$$
$$T(n) \leq an$$

in $T(n) \leq cn + T(n/5) + T(7n/10)$

$$T(n) \leq cn + an/5 + 7an/10 = O(n)$$

So, $T(n)$is $O(n)$.

(b) At randomized select pivot Quicksort, when pivot is minimum or maximum elements, it has worst case $O(n^2)$. Because it recursively does same process just without pivot. So, if we set the pivot to a moderate value, it can be worst case $O(n)$. On the other hand, best case is to set the pivot median, so it $O(nlogn)$ Because each step, it takes $O(n)$ divide process and whole is $O(logn)$.
let's say algorithm A to find median from array of size n takes time of an.
then we set the median as a pivot to recursively execute the algorithm on two of (n/2) size array.

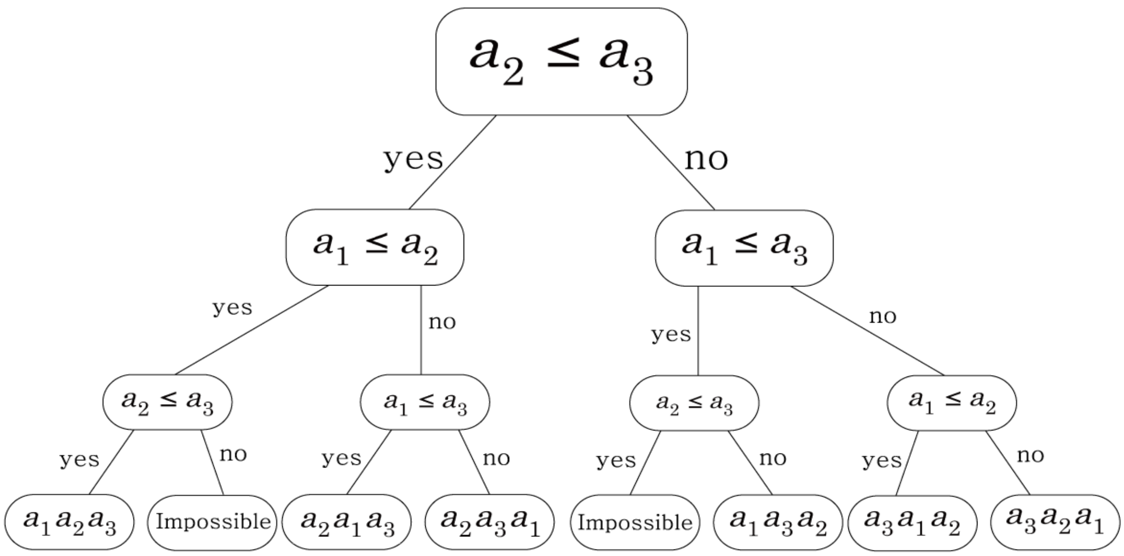
$$T(n) = an + 2T(n/2)$$
$$= an + 2 * a(n/2) + 4T(n/2)$$

⋮

$$= an * log_2(n) + nT(1) = O(nlogn)$$

In Quicksort, it called 'good call' when pivot divide the sequence that the sizes of L and G are each less than $3S/4$.
So When we select median by deterministic selection algorithm, Quicksort's worst case is best case of randomized Quicksort $O(nlogn)$.

3. (a) assume that the array is $a_1a_2a_3$ (whose first element is a_1, second is a_2 and last is a_3. the sorting algorithm will be mergesort which splits the array into subsequences of sizes ⌊n/2⌋ and ⌈n/2⌉ , here , when the size is 3, subsequences' sizes will be 1 and 2. then the subsequences will be $a_1$and a_2a_3. and in a_2a_3 it will be splited to $a_2$and $a_3$again, and will make new sorted array of $a_2$and $a_3$by comparing them, and that new array will be compared with $a_1$to make the final sorted array. To show it with a comparison tree



(b) if you see the tree, from leaf to root there are 3 comparisons to find kth(size) element from unsorted a1a2a3. let's say the number of comparisons to the leaf is Depth.
now when there are n number of elements in given array, to search kth element

from the array. from the process of proving when we deal with mergesort(which has the fastest worst T(n)), spliting the array recursively until the last subsequences' size become 1, has the fastest worst T(n)= $\Omega(nlogn)$. those cases, the Depth is ⌈logn⌉ but for searching(not sorting), we don't need to check every nodes on the kind of tree, we just need to go through the tree and arrive at a leaf to completely specify the element of kth index because like in problem 1-a after comparison we can judge what side(of 2) we should compare with.

then, in the worst case, we have to check at least ⌈logn⌉ times to check to the leaf.