31/01/2022

Order Statistics

Problem Statement

Input: An unsorted Array A, an integer k

Output:- Find the Kth- smallest number in A.

(Array A has distinct numbers)

$K=1$: minimum element in A

$K=n$: maximum element in A

To find minimum / maximum you just scan the array.

How much time this will take? $O(n)$.

$K=\frac{n}{2}$: median element in A.

Suppose you are allowed to use sorting.

→ Sort A (merge-sort)

→ return $A[\frac{n}{2}]$.

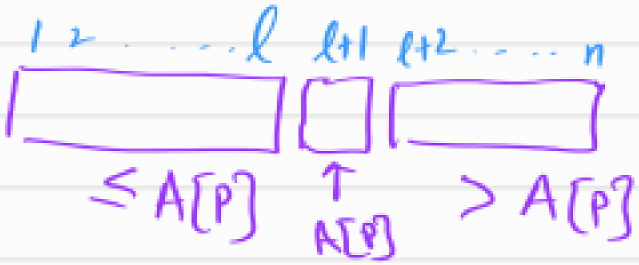
How much time the above algo. will take?

$O(n \log n)$.

Today we will see a linear time algo.

Recall the subroutine that returns the rank of a element in an array A.

Partition $(A[1,\dots,n], P)$ $\xrightarrow{\text{output}}$ how many elements are smaller than $A[P]$ + 1.



if $k \le l$ $\longrightarrow$ recurse in the left part

if $k > l$ $\longrightarrow$ recurse in the right part

if $k = l$ $\rightarrow$ return $A[l+1]$

Select-V1 $(A[1,\dots n], k)$:

if $n = 1$
return $A[n]$

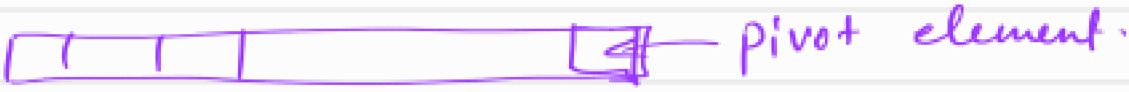
else choose a pivot element $A[P]$

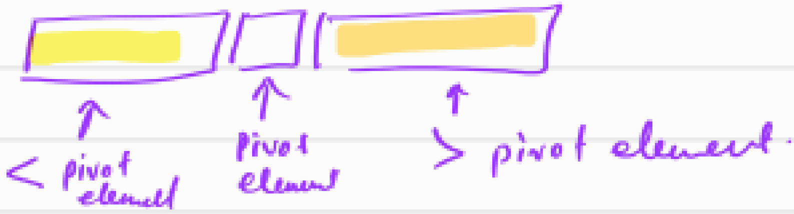
$r \leftarrow$ Partition $(A[1,\dots,n], P]$

if $k = r$
return $A[r]$

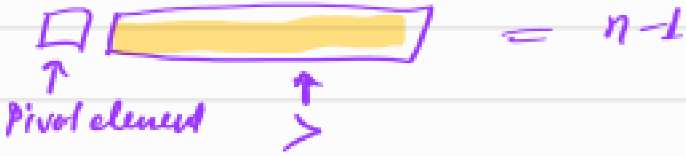
if $k < r$,
return Select-V1 $(A[1,\dots,r-1], k)$

else if $k > r$
return Select-V1 $(A[r+1,\dots,n], k-r)$.

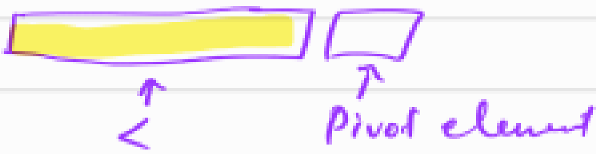




Unlucky:- Pivot element is the minimum in the array.



Similar things will happen if your pivot element is the maximum in the array.



How does the recursion looks like.

$$T(n) \leq T(n-1) + \Theta(n)$$

$$\leq O(n^2).$$

Another Solution:- Randomization.

Choose a pivot element uniformly at random $A[1, \ldots, n]$

Can show that in expectation this will run in $O(n)$ time.

Worst case linear time algorithm:

Suppose you have a recurrence relation as follows.

$$T(n) \leq T(\lambda n) + \theta(n)$$

when $\lambda < 1$.

$$\leq T(\lambda \cdot \lambda n) + \theta(\lambda n) + \theta(n)$$

$$\leq T(\lambda^3 \cdot n) + \theta(\lambda^2 n) + \theta(\lambda n) + \theta(n)$$

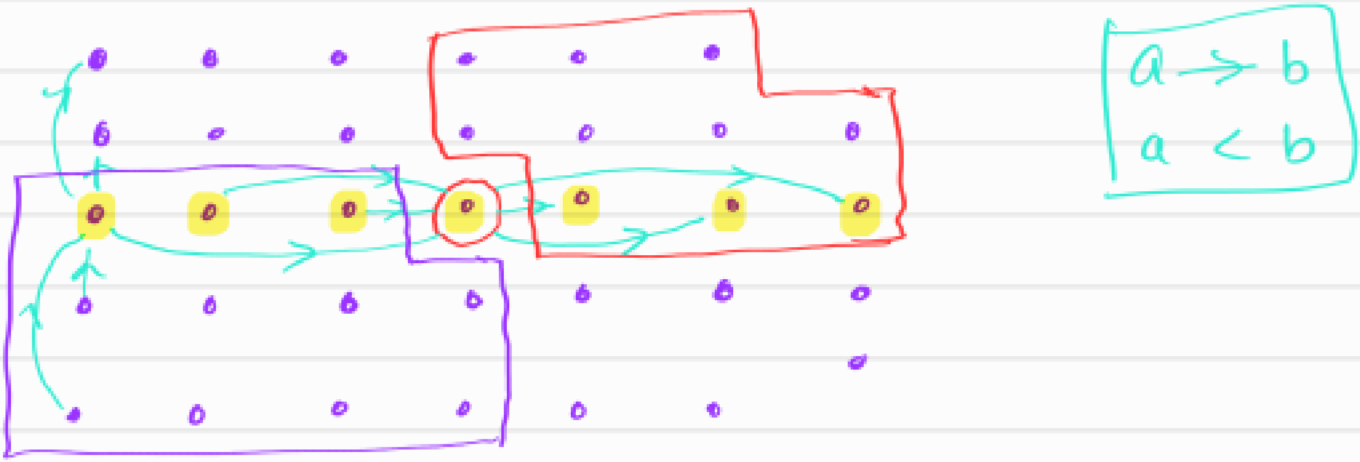
$$\leq \theta(n)\left(1 + \lambda + \lambda^2 + \lambda^3 + \ldots + \lambda^{\log n}\right)$$

↑ Geometric Series.

$$\leq \theta(n) \cdot \frac{1(1-\lambda^{\log n+1})}{1-\lambda}$$

$$\leq \theta(n) \cdot \frac{1}{1-\lambda} \leftarrow \text{Some constant.}$$

Aim is to find a pivot that guarantees that the size of the subproblem is at most a fraction of intial size. (strictly < 1).



Step 1 :- make $\lceil \frac{n}{5} \rceil$ groups of size exactly 5 except the last one.

Step 2 :- find median in each group.
you will take constant time for each group.
So overall $\Theta(n)$ time.

Step 3 :- Find median in the set of yellow elements. Make a recursive call on this set of $\lceil \frac{n}{5} \rceil$ elements.

elements smaller than median

$$\geq 3\left(\left\lfloor \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rfloor - 1\right) \geq \frac{3n}{10} - 3$$

elements larger than median

$$\geq 3\left(\left\lfloor \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rfloor - 2\right) \geq \frac{3n}{10} - 6$$

Step 4 :- Choosing the median found in Step 3, make a call to the partition subroutine.

Step 5 :- if the rank of the median found in Step 3 is r.

if $k = r$, then return $A[r]$

if $k < r$, then make a recursive call to $A[1, \dots, r-1]$ to find the k-th smallest element.

if $k > r$, then make a recursive call to $A[r+1, \dots, n]$ to find the $(k-r)$th smallest element.

Runtime recursion :-

$$T(n) \le \underline{T\left(\frac{n}{5}\right)} + \underline{T\left(\frac{7n}{10}\right)} + \Theta(n)$$

$$\le \Theta(n) \qquad \text{(Solve!)}$$

Recall :- $T(n) \le 2T\left(\frac{n}{2}\right) + \Theta(n)$

This solves to $\rightarrow$ $n \log n$.

$\frac{n}{5} = \frac{2n}{10}$, $\frac{7n}{10}$: size of two subproblems $\le \frac{2n}{10} + \frac{7n}{10} = \frac{9n}{10}$

Quick Sort: (1) find $\frac{n}{2}$-th smallest element using the above algorithm in $O(n)$ time.

(2) recurse on to two subproblems

Runtime: $T(n) \leq 2 \cdot T\left(\frac{n}{2}\right) + \Theta(n)$