CSL 630 Lecture 7 Aug 14

Given a set S of n elements and an integer $1 \le k \le n$, find the k^{th} rank element is S.

An element x has rank k, if there are exactly $k-1$ elements smaller than x in S.

Assume w.l.o.g. that all elements are unique.

$$S = \begin{bmatrix}5\\1\end{bmatrix}, \begin{bmatrix}9\\2\end{bmatrix}, \begin{bmatrix}1\\3\end{bmatrix}, \begin{bmatrix}3\\4\end{bmatrix}, \begin{bmatrix}8\\5\end{bmatrix}, \begin{bmatrix}11\\6\end{bmatrix}$$

rank $(S, 2)$ 3 rank 1 : 1

Simplest way to find rank k element $\text{Rank}(S, k)$ is to sort and read off the k^{th} element

Time : Time to Sort + $O(1)$

$\text{Sort}(n) + O(1)$

Question : Do we have to Sort

Remark on uniqueness:

We can make every element unique by appending the index of the element.

Other methods: Put elements in a heap and delete-min $k-1$ times

Time: $O(k \log n)$ + Time to heapify

: $O(n + k \log n)$

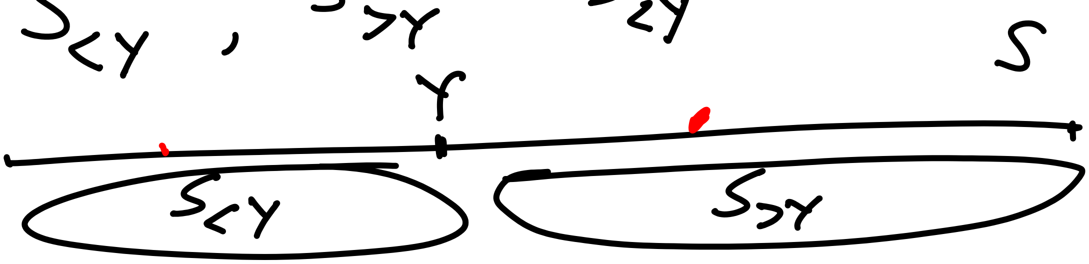
$\leq n$ for $k \leq \frac{n}{\log n}$

$k \sim \frac{n}{2}$: $\Omega(n \log n)$

Idea of partition:

We find a pivot/splitter, say Y and divide into two subsets $S_{<Y}$, $S_{>Y}$

$S_{<Y}$: all elements $< Y$



While we partition, we also compute the rank of Y in S

If rank(S, Y) = K then done

Partition + rank $O(n)$ time

Select (S, K)

Find some pivot Y. Partition.

If rank(S, Y) = K report Y

else if rank(S, Y) > K then

Select ($S_<$, K)

else Select ($S_>$, K − rank(S, Y))

What is the Running Time?

If the pivot is "bad" then we can incur $O(n + n-1 + n-2 \cdots)$

$= O(n^2)$ cost

The "best" pivot is the middle element $\Rightarrow O(n + \frac{n}{2} + \frac{n}{4} + \cdots)$

$= O(n)$

How do we choose the pivot

- Choose the first element
- choose an element [1.. n] uniformly at random ✓

i.e. all elements are equally likely to be chosen
(use a random no. generator)

S

After the recursive call, what is the average size of the "larger subset"

rank 1	2	3	i		n
$n-1$	$n-2$	$n-3$	$\max\{n-i, i\}$	$n-2$	$n-1$

Size of the subproblem : X : random variable

$$1 \leq X \leq n$$

$$E[X] = \sum_i \Pr(X=i) \cdot i$$

$$= \frac{1}{n} \sum_i i = \frac{2}{n} \sum_{\frac{n}{2} < i \leq n} i$$

$$= \frac{3n}{4}$$

If the subproblem sizes are $\left(\frac{3}{4}\right)^i n$, then running time is $O(n)$

One option will be to repeatedly choose pivots until we find a "good" pivot and then call recursively

A "good" pivot y will have rank $\left[\frac{n}{4}, \frac{3n}{4}\right]$

(larger subproblem $\leq \frac{3}{4}n$)

How many times do we need to try before we succeed?

Prob of success $= \frac{1}{2}$

Find the expected no. of trials (independent)

trials is a random variable : Y

$$E[Y] = 2$$

The expected cost at level i of recursion

$= O(n_i)$ where n_i : subproblem size at level i

$$= \left(\frac{3}{4}\right)^i n$$

Expected cost of the algorithm

$E[T]$ $\qquad$ T is a r.v.

$$E[T] = E[T_1 + T_2 + T_3 + \cdots T_{\log_{4/3} n}]$$

$$= E[T_1] + E[T_2] + \cdot$$

$$O(n) + O(\tfrac{3}{4}n) + \cdots$$

$$= O(n)$$

For the original algorithm write the recurrence for the expected running time and solve it

$$T(n, k) = T(n', k') + O(n)$$