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

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.

$$S = \begin{bmatrix} 5 \\ 1 \\ 9 \\ 2 \\ 7 \\ 4 \end{bmatrix}, \quad \begin{bmatrix} 3 \\ 3 \\ 8 \\ 5 \\ 11 \end{bmatrix}$$

$$\text{rank}(S, 2) = 3$$

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

$$\text{Time} : \quad \text{Time to Sort} + O(1)$$

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

**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:

\[ O(K \log n) + \text{Time to heapify} \]

\[ \leq n \text{ for } k \leq \frac{n}{\log n} \]

\[ K \approx \frac{n}{2} \implies \Omega(n \log n) \]

Idea + partition:

We find a pivot/splitter, say \( Y \) and divide into two subsets:

\( S_{< Y}, S_{> Y}, S_{\leq Y} \): all elements, \(< Y \)

\[ S \]

\[ S_{< Y}, S_{> Y} \]
While we partition, we also compute the rank of \( X \) in \( S \).

If \( \text{rank}(S,Y) = k \) then done.

Partition + rank \( \text{O}(n) \) time

Select \((S,k)\)

Find some pivot \( Y \). Partition.

If \( \text{rank}(S,Y) = k \) report \( Y \)

else if \( \text{rank}(S,Y) > k \) then

Select \((S_<,k)\)

else Select \((S_>,k-\text{rank}(S,Y))\)

What is the Running Time?

If the pivot is "bad" then we can incur \( O(n + n-1 + n-2 \ldots) \), \( O(n^2) \) cost.

The "best" pivot is the middle element \( \Rightarrow O(n + \frac{n}{2} + \frac{n}{4}, \ldots) \) \( = O(n) \)
How do we choose the pivot

- Choose the first element
- Choose an element \([1..n]\) uniformly at random \(\checkmark\)
i.e. all elements are equally likely to be chosen
(use a random no. generator)

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

\[
\begin{align*}
&\text{max}_{n=1}^{n-1} n-2\ n-3\ \max_{n=2}^{n-1}\ \{n-i, i\} \\
&1 \leq X \leq n \\
&\mathbb{E}[X] \leq \mathbb{P}(X=i) \cdot i \\
&\text{with } \frac{1}{n} \leq i \leq \frac{2}{n} \leq \frac{n}{4} \\
&\text{and } \frac{n}{4} \leq i \leq n
\end{align*}
\]
If the subproblem sizes are \((\frac{3}{4})^i n\), 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 in a random variable: \(Y\)

\[E[Y] = 2\]

The expected cost at level \(i\) is remain

\[= O(n_i)\] where \(n_i\): subproblem size at level \(i\)

\[= (\frac{3}{4})^i n\]
Expected cost of the algorithm

\[ E[T] \quad T \text{ is a r.v.} \]

\[ E[T] = E[T_1 + T_2 + T_3 + \cdots + T_{\log_2 n}] \]

\[ = E[T_1] + E[T_2] + \cdots \]

\[ = O(n) + O(3^{\frac{3}{4}n}) + \cdots \]

\[ = O(n) \]

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

\[ T(n, k) = T(n', k') + O(n) \]