Selection

Input: - sequence S of n numbers
- integer k with $1 \le k \le n$

Output: k-th smallest element in S.

$k=1 \Rightarrow$ minimum (can be done in $O(n)$)

$k=n \Rightarrow$ maximum (can be done in $O(n)$)

$k=n/2 \Rightarrow$ median ??

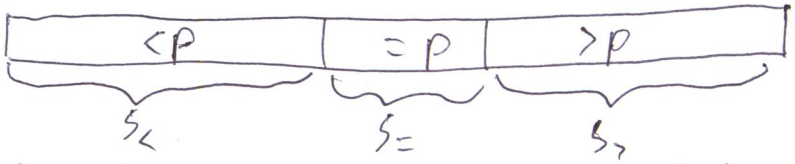
Algorithm:

```
Sort S
return S[k]
```

This runs in $\Theta(n \log n)$ time.
Can we do it in $O(n)$?

Algorithm Select(S, k):

```
if |S|=1 then return S[1]
else
    choose an element p∈S (pivot)
    divide S into S<, S=, and S>
```




```
    if k ≤ |S<| then return Select(S<, k)
    else if k > |S<|+|S=| then return Select(S>, k-|S<|-|S=|)
    else return p
```

The running time depends on the pivot.

Worst case: $k=1$, in each recursive call: $p=\max$

$\Rightarrow \Theta(n^2)$ running time

Good case: in each recursive call,

$|S_<| \leq \alpha n$ and $|S_>| \leq \alpha n$ for some constant $0<\alpha<1$

Then the running time $T(n)$ satisfies

$$T(n) = n + T(\alpha n)$$

$$\Rightarrow T(n) = (1+\alpha+\alpha^2+\ldots)\cdot n$$

$$\leq \frac{1}{1-\alpha}\cdot n$$

$$= O(n)$$

$$\left(\sum_{i=0}^{\infty} r^i = \frac{1}{1-r} \text{ provided that } |r|<1. \right)$$

How do we get the good case?

① Pick p at random. This results in the good case with high probability, giving expected $O(n)$ time. (Implement this one)

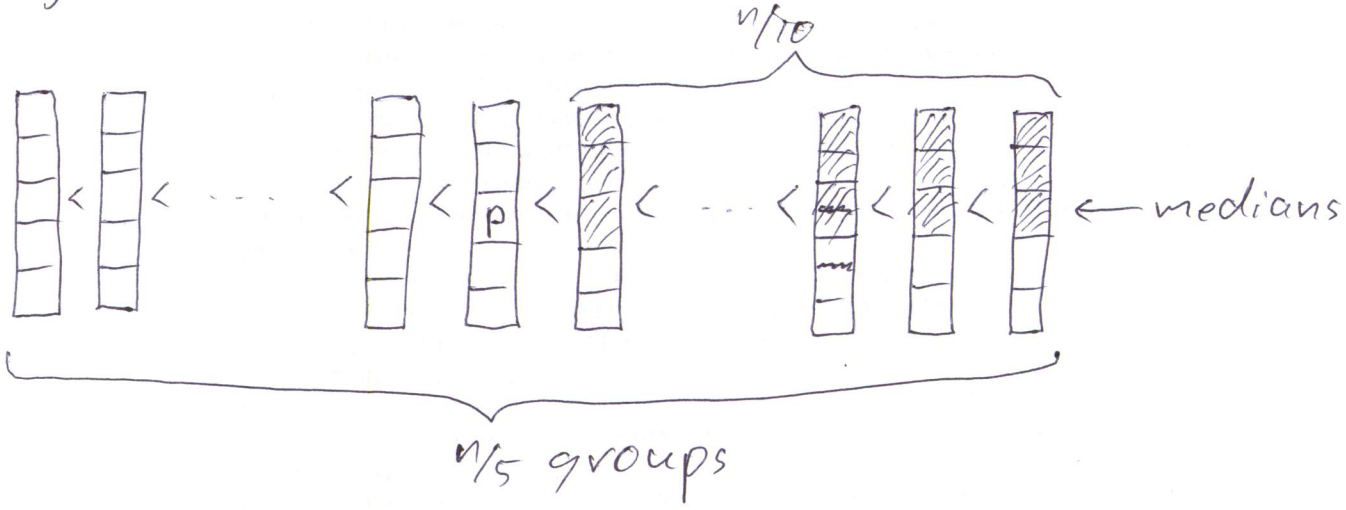
② If we really want $O(n)$ worst-case time, use median-of-medians (1973).

Algorithm:

1. Divide the input sequence into $n/5$ groups of 5 elements each.
2. Compute the median of each group (m_i).
3. Recursively compute the median of $m_1, m_2, \ldots, m_{n/5}$
4. Use this median as p in Select.

Why is p a good pivot?

Suppose all the groups of 5 are sorted top-to-bottom, and the groups are sorted left-to-right by their medians:



How many elements are bigger than p?

- All medians to the right of p ($4n/10$)
- All elements above those medians ($2n/10$)

$\Rightarrow |S_>| \geq 3n/10$

$\Rightarrow |S_<| \leq 7n/10$. (Same for $S_<$)

This makes the total runtime:

$$T(n) = T(n/5) + T(\tfrac{7n}{10}) + n$$

Solving this by unfolding or recursion tree gets very messy. We will use induction. §

Claim: $T(n)$ ~~is $O(n)$~~ $\leq c \cdot n$ for some constant $c > 0$.

Proof: By choosing c sufficiently large, the claim holds for "small" values of n.

Suppose that n is "large" and $T(m) \leq c \cdot m$ for all $1 \leq m < n$. Then

$$\begin{aligned} T(n) &= T(n/5) + T(\tfrac{7n}{10}) + n \\ &\leq c \cdot n/5 + c \cdot \tfrac{7n}{10} + n \\ &= \tfrac{9}{10} \cdot c \cdot n + n \\ &\leq c \cdot n \qquad \text{for } c \geq 10 \end{aligned}$$

$\square$

$$\begin{aligned} \tfrac{9}{10} \cdot c \cdot n + n &\leq c \cdot n \\ n &\leq \tfrac{1}{10} \cdot c \cdot n \\ 1 &\leq \tfrac{1}{10} \cdot c \\ 10 &\leq c \end{aligned}$$

Thus we can find the k-th smallest element in a sequence in $O(n)$ time.