Selection

The problem: Find the k^{th} smallest key in a set of keys.

1. Selection of the minimum.

$\Theta(n)$
```
function Min
  min := 1;
  for i := 2 to n do if A[min] > A[i] then min := i endif
  endfor
```

2. Randomized Selection : Similar to Quicksort

```
function Random-Select (p, r, i)
    if p=r then return p
        else q := Random-Partition (p, r)
             k := q-p+1;
             if i ≤ k then Return
                      Random-Select (p, q, i)
                  else Return
                      Random-Select (q+1, r, i-k)
             endif
    endif
```

Average Case Analysis:

Recall with prob. $\frac{2}{n}$ the array is split 1 and $(n-1)$, and for $2 \le k \le n-1$, ~~the~~ with prob. is $\frac{1}{n}$ it is split k and $(n-k)$.

$$T(n) \le \frac{1}{n}\left(T(\max\{1, n-1\}) + \sum_{k=1}^{n-1} T(\max\{k, n-k\})\right) + \Theta(n)$$

$$\le \frac{1}{n}\left(T(n-1) + 2\sum_{k=\lceil \frac{n}{2} \rceil}^{n-1} T(k)\right) + \Theta(n)$$

$$= \frac{2}{n}\sum_{k=\lceil n/2 \rceil}^{n-1} T(k) + O(n)$$

Assume inductively that

$T(n) \le cn$ for a large enough c.

$$T(n) \le \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lceil n/2 \rceil - 1} k\right) + O(n)$$

$$\le c(n-1) - \frac{c}{2}\left(\frac{n}{2} - 1\right) + O(n)$$

$$\le \frac{3c}{4} n + O(n) \le cn \text{ if } c \text{ is large enough.}$$

Think about which c you should choose.

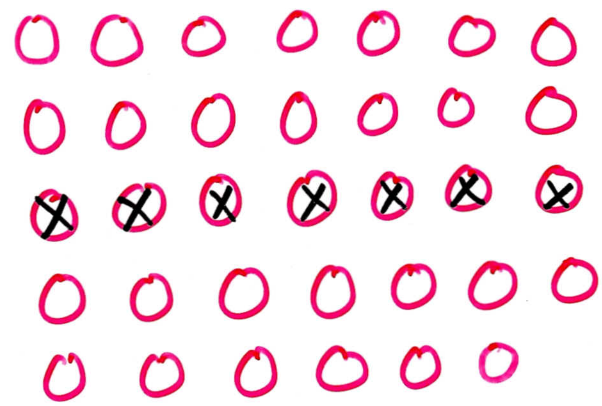
3. Deterministic Selection

Observe that the randomized one takes $\Omega(n^2)$ in the worst-case if splits are unbalanced.

It is possible to select in $O(n)$ time.

Algorithm

1. Divide the keys in groups of 5



(e.g. for $k = \lceil n/5 \rceil$, $A[j]$, $A[j+k]$, $A[j+2k]$, $A[j+3k]$, $A[j+4k]$ is a group for $1 \le j \le k-1$.)

2. Find the median of each group.
   (use some simple sorting on each group)

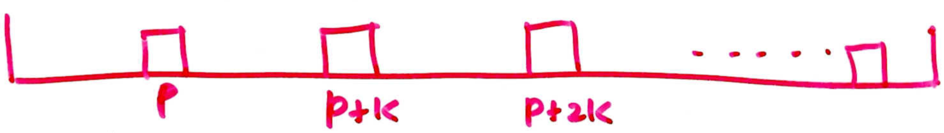
3. Recurse on the $\lceil \frac{n}{5} \rceil$ medians

4. Partition the array with the median-of-medians as pivot.

5. if $i \le q$ then find the key on the low part
   else find the $(i-q)$th key on higher part

4. Insertion Sort with Jumps

```
Procedure Insertion-Sort (p, k, n);
    last := p+k;
    while last ≤ n do
        pos := last;
            while pos > p and A[pos] > A[pos-k] do
                A[pos] ⟷ A[pos-k];
                pos := pos-k
            endwhile
        last := last + k;
    endwhile
```



We will use this sort in our Selection algorithm.

We implement the high-level algorithm for the Selection.

```
function Select (p, r, i)
    if r-p+1 ≤ 50 then Insertion-Sort (p, 1, r)
                 return A[p+i-1]
    else
        k = ((r-p+1)+4) div 5
        for j:=0 to k-1 do
            Inser-Sort (p+j, k, r)
        endfor
        medmed := Select(p+2k, p+3k-1, k div 2)
        A[p] <-> A[medmed]
        q := Partition (p, r)
        if i ≤ q-p+1 then return
                 Select (p, q, i)
            else return
                 Select (q+1, r, i-(q-p+1))
        endif
    endif
```

Small data

mod 5

Sorting 5 elements in each group

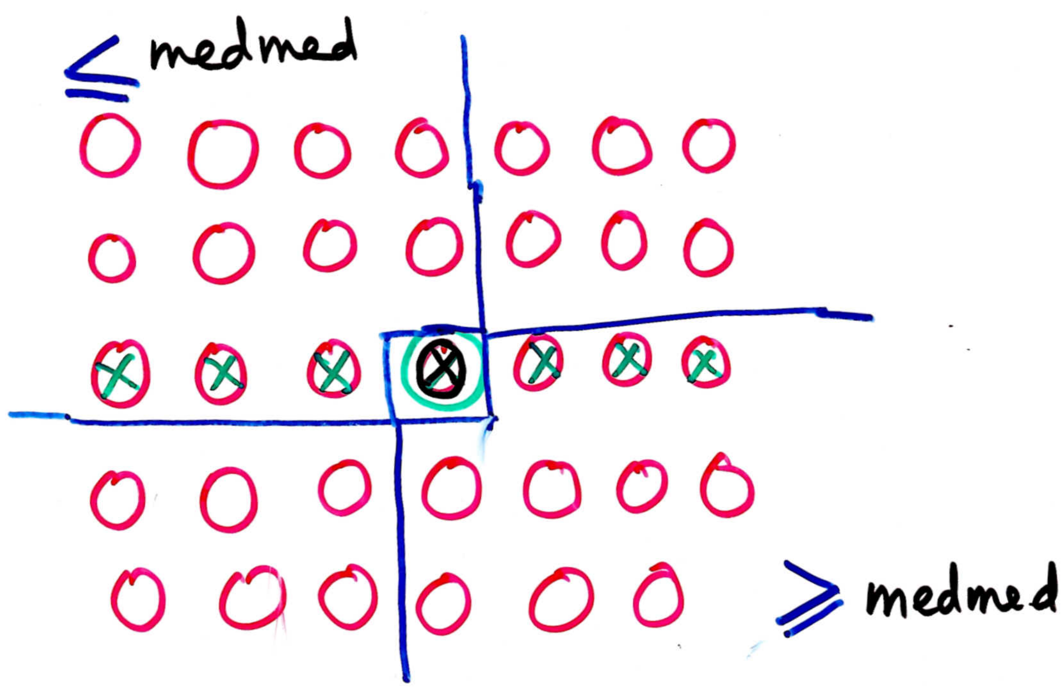
median of medians

Partition with median of medians

prune and search

Analysis. For analysis we ignore the ceiling and floor functions.

- The number of medians less than or greater than the medmed is $\frac{n}{10}$
- The number of elements less than or greater than these medians is $\frac{2n}{10}$



- We throw away (prune) at least $\frac{3n}{10}$ elements.

Recurrence relation for time Complexity.

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

Prove $T(n) = O(n)$ by assuming

$T(n) \le cn.$ ← Inductive hypothesis

$$T(n) \le c\frac{n}{5} + c\frac{7n}{10} + O(n)$$

applying induction

$$\le c \cdot \frac{9n}{10} + O(n)$$

$$\le c \cdot \frac{9}{10} n + c'n$$

$$\le cn\left(\frac{9}{10} + \frac{c'}{c}\right)$$

Choose c such that $\frac{c'}{c} + \frac{9}{10} \le 1$.

choice of c is upto us.

or, $c \geq 10c'$.