Lecture 7: March 1, 2000

Today:

- Median
- Order Statistics

Order Statistics

- Find the ith smallest of n elements.
- Find element with **rank** i; i.e., element larger than exactly $i-1$ others
- $i = 1$: Minimum
- $i = n$: Maximum
- $i = \lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$: Median.

First idea:

- Sort, then take ith element.
- Running time $= O(n \lg n) + O(n) = O(n \lg n)$
- Can we do this faster?

Order Statistics

Can find some kinds of elements in $O(n)$ time.
Example: Minimum

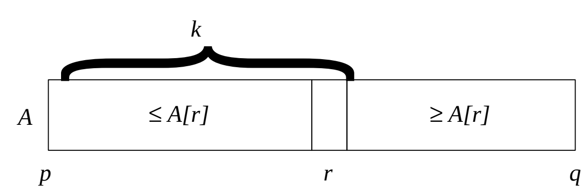
```
MINIMUM(A)
  min ← A[1]
  for i ← 2 to length[A]
       do if min > A[i]
             then min ← A[i]
  return min
```

Same for Maximum.
What about general selection problem?

Divide and conquer

To select an element with rank i:

- partition the array according to some x (as in Quicksort), assume k elements $\leq x$
- if $i > k$ recurse on the right side, otherwise recurse on the left side



Option 1: choose x at random

```
RANDOMIZED-SELECT(A, p, r, i)
if p = r
    then return A[p]
q ← RANDOMIZED-PARTITION(A, p, r)
k ← q − p + 1 ▷ What is k?
if i ≤ k
    then return
            RAND-SELECT(A, p, q, i)
    else return
            RAND-SELECT(A, q + 1, r, i − k)
```


Example

Find 4th element of

A: | 4 | 7 | 2 | 9 | 8 |

A: | 4 | 2 | 7 | 9 | 8 |

A: | 9 | 8 |

A: | 8 | 9 |

A: | 8 |

Analysis

Very lucky (split in the middle):

$$\begin{aligned} T(n) &\leq T(\frac{1}{2}n) + \Theta(n) \\ &= \Theta(n) \end{aligned}$$

Unlucky:

$$\begin{aligned} T(n) &= T(n-1) + \Theta(n) \\ &= \Theta(n^2) \end{aligned}$$

Worst case is worse than sorting!

Analysis Cont.

Recall from the Quicksort lecture:

- a split (say, of $A[p \dots r]$) is "lucky" for a_i, if the part where a_i ends up in is of size $\leq 3/4(r-p+1)$, i.e., if we reduce the array size by a factor of $3/4$
- after $\log_{4/3} n$ lucky splits, we are done
- at any time, a_i is lucky with probability at least $1/4$

Analysis Cont.

Let T_i be the number of partitions between the $i-1$th and ith lucky split.
The total time is bounded by

$$T = \sum_i T_i (3/4)^{i-1} n = n \sum_i T_i (3/4)^{i-1}$$

Therefore, the *expected* running time is at most

$$E[T] = E[n \sum_i T_i (3/4)^{i-1}] = n \sum_i (3/4)^{i-1} E[T_i]$$

(by linearity of expectation).

Analysis cont.

What is $E[T_i]$?

- sequence of independent trials with probability $p \geq 1/4$ of success
- $T_i = T_1$ is the waiting time for the next success

Turns out $E[T_1] = 1/p$.

Proof: We can write

$$E[T_1] = p \cdot 1 + (1-p) \cdot (1 + E[T_1])$$

since with probability p we are done, and with probability $1-p$ we used one trial and have to continue. The theorem follows.

Analysis - the final step

$$E[T] = n \sum_i (3/4)^{i-1} E[T_i] = n \sum_i (3/4)^{i-1} = O(n)$$

(since $(3/4)^{i-1}$ forms a geometric sequence).

Done !

(but the algorithms is randomized)

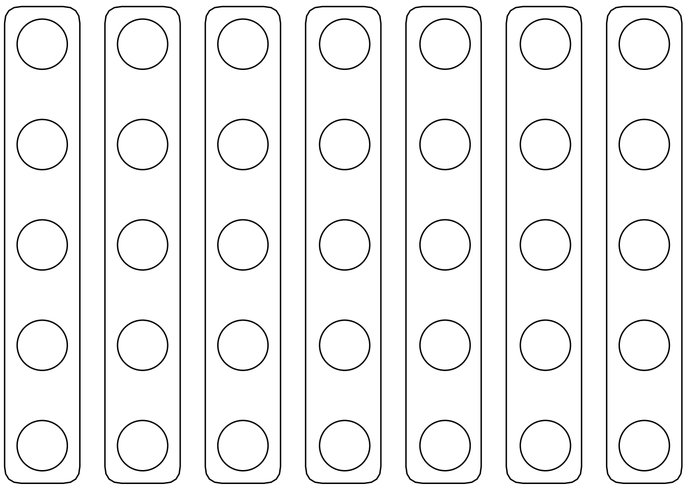
Linear-Time Selection

Running time **linear** in worst case
Of theoretical interest only
Idea: generate a good partitioning element x.

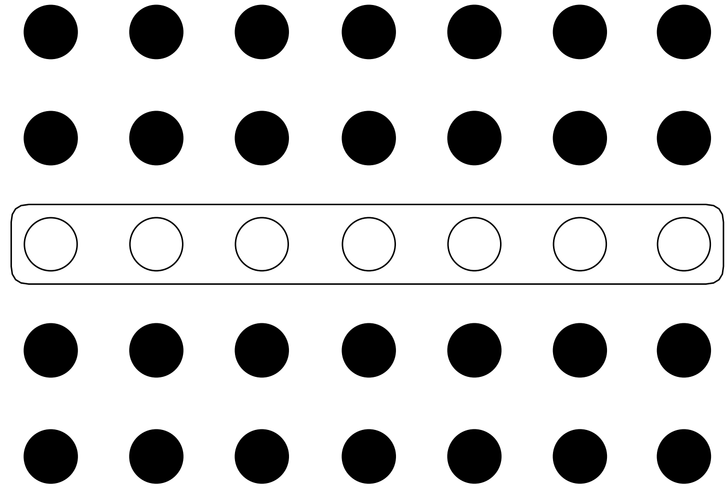
Select Algorithm

SELECT(i)

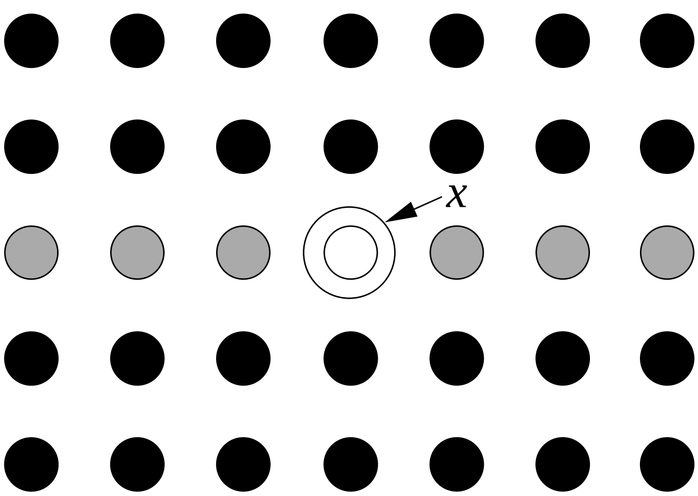
1. Divide the n elements into groups of 5.



2. Find median of each group of 5 by sorting (constant time!).

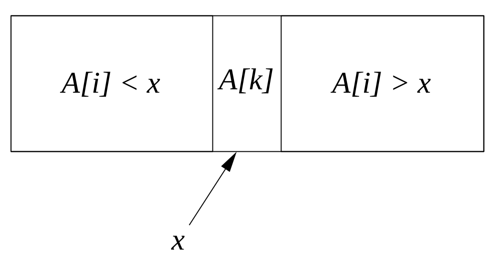


3. Use SELECT recursively to find median x of $\lfloor n/5 \rfloor$ medians from Step 2



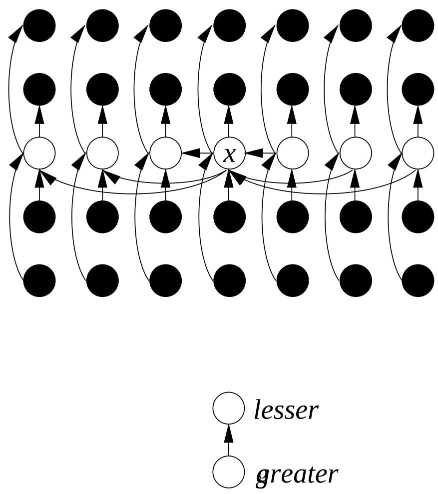
Select Algorithm

Partition elements around x. Let $k = \textbf{rank}(x)$.



if $i = k$
  then return x
if $i < k$
  then use SELECT recursively to find ith smallest in low part
  else use SELECT recursively to find $(i-k)$th smallest in high part

Fractional Ordering



Analysis

At least 1/2 of 5-element medians $\leq x$.
$\lfloor \lfloor n/5 \rfloor /2 \rfloor = \lfloor n/10 \rfloor$ medians are $\leq x$.
At least $3\lfloor n/10 \rfloor$ elements $\leq x$.
For $n \geq 50$, $3\lfloor n/10 \rfloor \geq n/4$.

Analysis

So, after partitioning around x, step 5 is called on $\leq 3n/4$ elements, so

$$T(n) \leq T(n/5) + T(3n/4) + O(n)$$

Looks like linear time, since $1/5 + 3/4 < 1$.

Formally, substitute $T(n) \leq cn$

$$\begin{aligned} T(n) &\leq cn/5 + 3cn/4 + O(n) \\ &= 19cn/20 + O(n) \\ &= cn - (cn/20 - O(n)) \\ &\leq cn \text{ if } c \text{ big enough} \end{aligned}$$

$\Rightarrow$ SELECT is deterministic, and linear time!

Intuition

- work at each level of recursion is reduced by a constant fraction
- analysis involves geometric sum
- work at root dominates $\Rightarrow$ linear time

Applications

Worst-case $\Theta(n \lg n)$ quicksort

1 Find median x and partition

2 Recursively sort two halves.

$T(n) = 2T(n/2) + \Theta(n) = \Theta(n \lg n)$

With worst case linear-time median, many problems reduce to simple divide and conquer methods!