Order statistics

Select the ith smallest element of n elements (element with *rank* i).

- $i = 1$: *minimum*
- $i = n$: *maximum*
- $i = \lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$: *median*

Naive algorithm

NAIVE-SELECT(A, i)

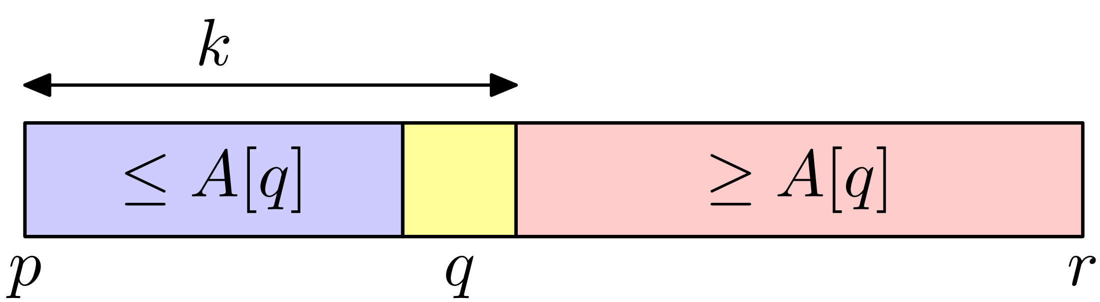
```
1.    MERGE-SORT(A)        ▷ Don't use quicksort—why?
2.    return A[i]
```

Worst-case running time $= \Theta(n \lg n)$.

Randomized divide-and-conquer algorithm

$\textsc{Rand-Select}(A, p, r, i)$

if $p = r$ **then return** $A[p]$
$q \leftarrow \textsc{Rand-Partition}(A, p, r)$
$k \leftarrow q - p + 1$
if $i = k$ **then return** $A[q]$
if $i < k$
    then return $\textsc{Rand-Select}(A, p, q-1, i)$
    else return $\textsc{Rand-Select}(A, q+1, r, i-k)$



Example

- Select the $i = 7$th smallest:

11	10	13	5	8	3	2	6
							pivot

$i = 7$

- Partition:

5	3	2	6	8	10	13	11

$k = 4$

$$5 \quad 3 \quad 2 \quad 6 \quad \underbrace{8 \quad 10 \quad 13 \quad 11}_{\text{Select the } 7-4=3\text{rd smallest recursively}}$$

Analysis of Rand-Select

- **Lucky:** $T(n) = T(9n/10) + \Theta(n) = \Theta(n)$

  by case 3 of the master method.

- **Unlucky:** $T(n) = T(n-1) + \Theta(n) = \Theta(n^2)$

  which is worse than $\textsc{Naive-Select}$!

- **Summary:**
  - Linear expected time.
  - Excellent algorithm in practice.
  - But worst case is *very* bad.
    Imagine spending $\Theta(n^2)$ time to find the minimum element :-(

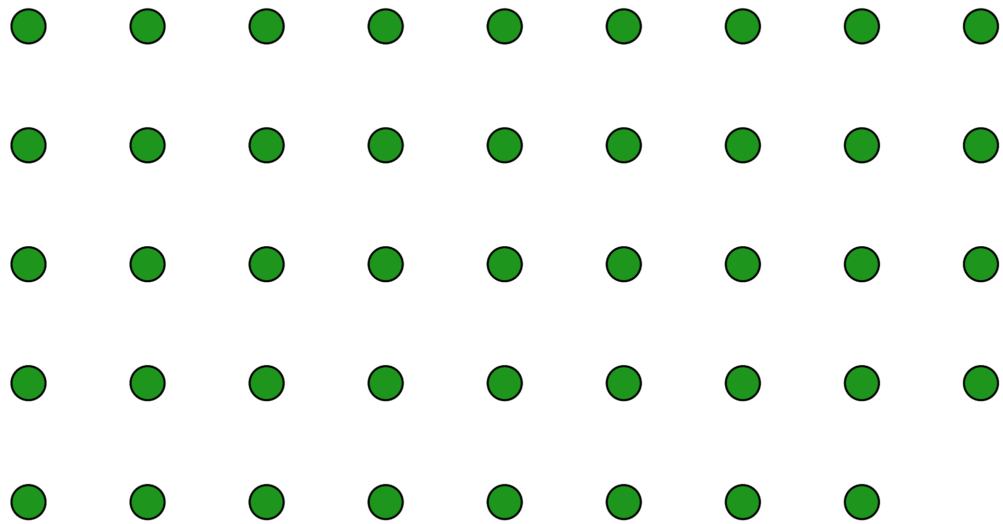
Selection in worst-case linear time

- Algorithm due to Blum, Floyd, Pratt, Rivest, and Tarjan [1973].
- Idea: generate a good pivot recursively.

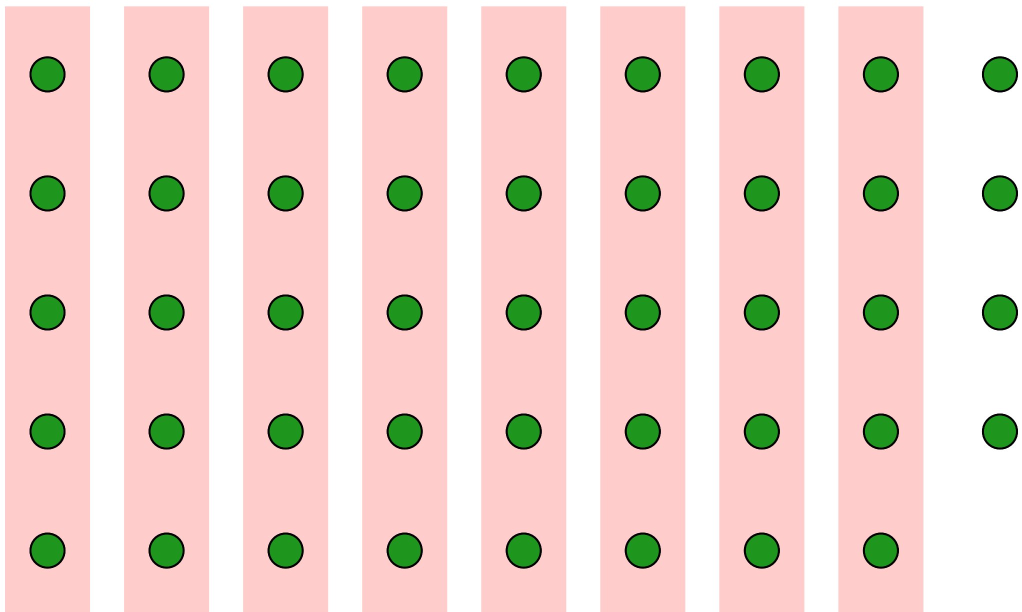
$\textsc{Select}(i, n)$

1. Divide the n elements into groups of 5.
2. Find the median of each 5-element group.
3. Recursively $\textsc{Select}$ the median x of the $\lfloor \frac{n}{5} \rfloor$ group medians to be the pivot.
4. Partition around the pivot x.
5. Let $k = rank(x)$.
6. If $i = k$, then return x.
7. If $i < k$, then recursively $\textsc{Select}$ the ith smallest element in first part.
8. If $i > k$, then recursively $\textsc{Select}$ the $(i - k)$th smallest element in last part.

Choosing the pivot

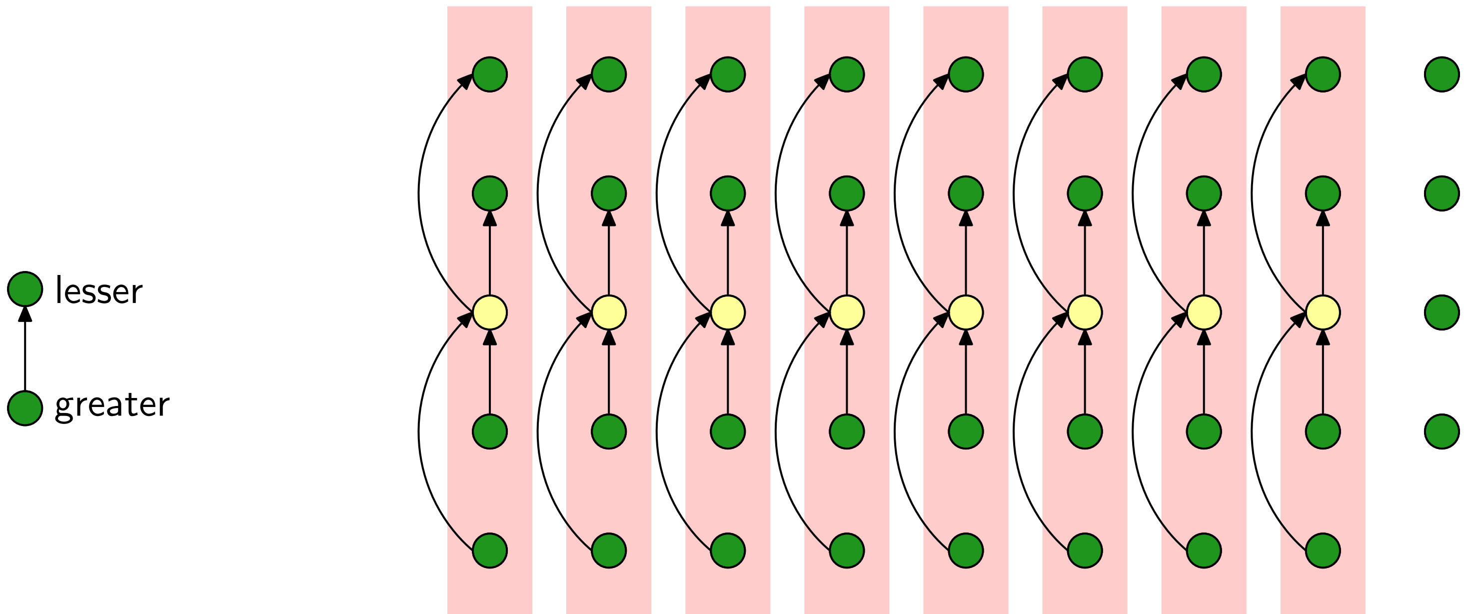


Choosing the pivot



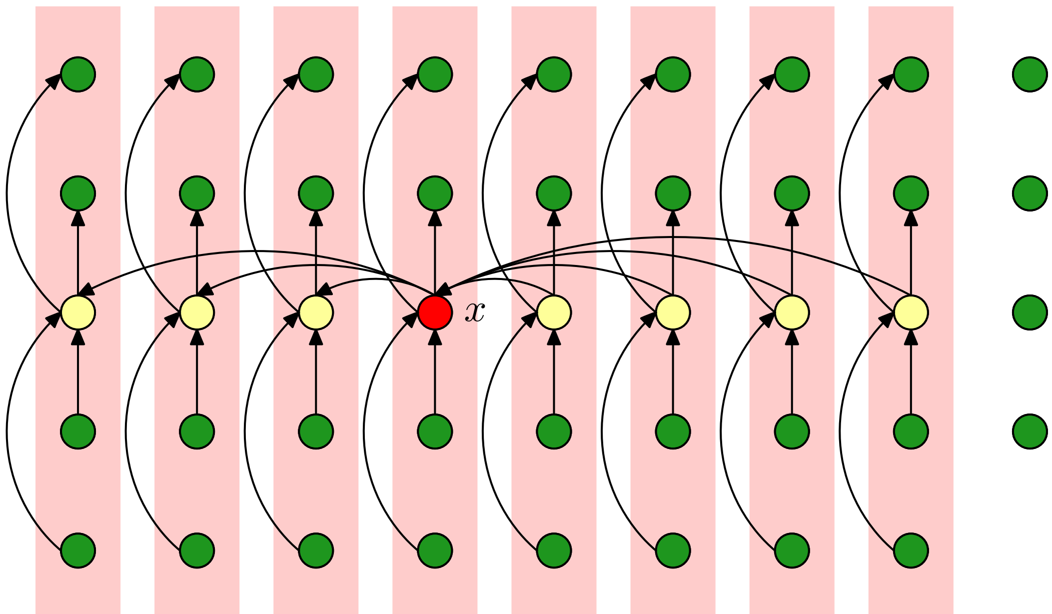
- Divide the n elements into groups of 5.

Choosing the pivot



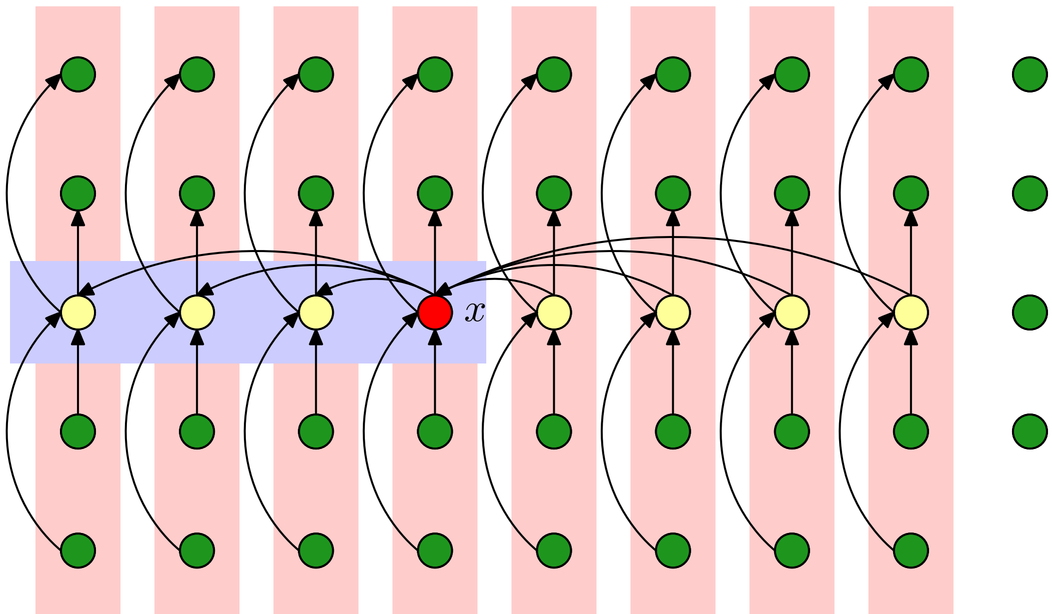
- Divide the n elements into groups of 5.
- Find the median of each 5-element group.

Choosing the pivot



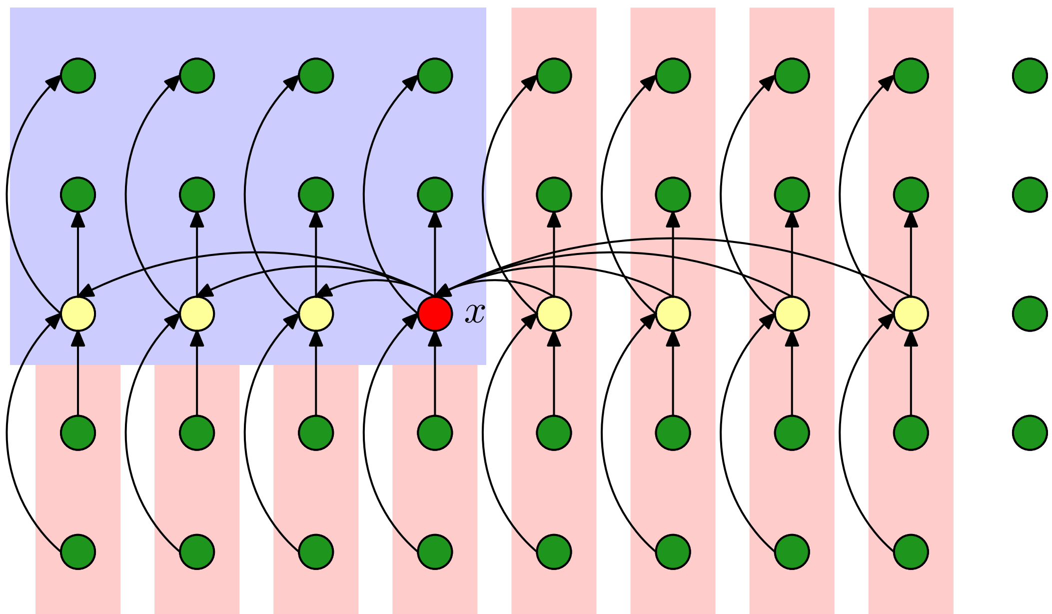
- Recursively $\textsc{Select}$ the median x of the $\lfloor \frac{n}{5} \rfloor$ group medians to be the pivot.

Analysis



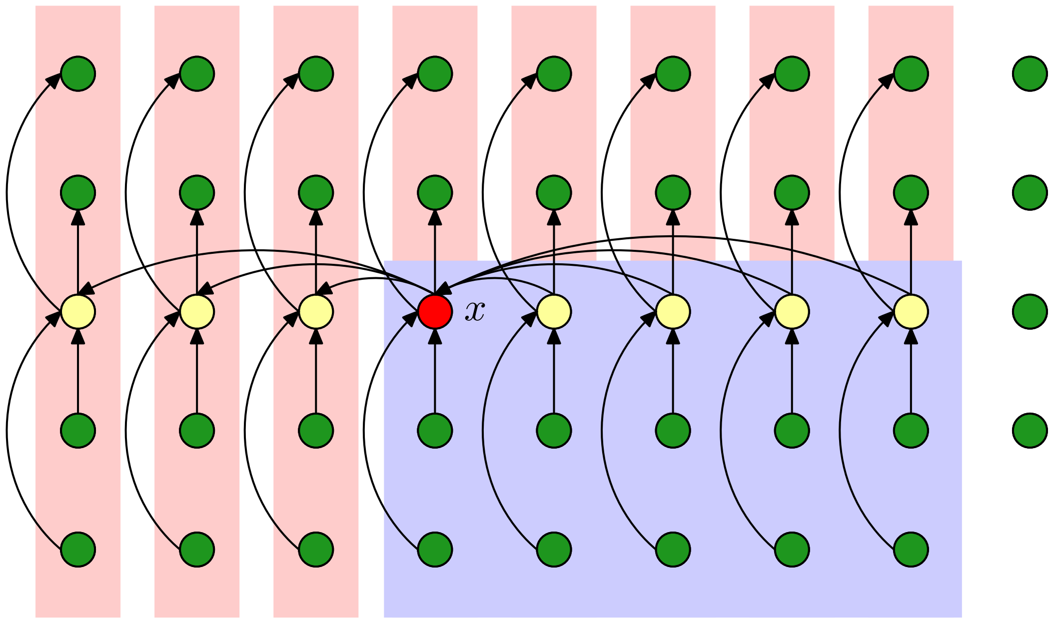
- At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor \frac{n}{5} \rfloor / 2 \rfloor = \lfloor \frac{n}{10} \rfloor$ group medians.

Analysis (assume all elements are distinct)



- At least half the group medians are $\leq x$, which is at least $\lfloor \lfloor \frac{n}{5} \rfloor / 2 \rfloor = \lfloor \frac{n}{10} \rfloor$ group medians.
- Therefore, at least $3 \lfloor \frac{n}{10} \rfloor$ elements are $\leq x$.

Analysis (assume all elements are distinct)



- Similarly, at least $3\lfloor \frac{n}{10} \rfloor$ elements are $\geq x$.

Minor simplification

- For $n \geq 50$, we have $3\lfloor \frac{n}{10} \rfloor \geq n/4$.
- Therefore, for $n \geq 50$ the recursive call to $\textsc{Select}$ in step 7–8 is executed recursively on $\leq 3n/4$ elements.
- Thus, the recurrence for running time can assume that step 7–8 takes time $T(3n/4)$ in the worst case.
- For $n < 50$, we know that the worst-case time is $T(n) = \Theta(1)$.

Developing the recurrence

$\textsc{Select}(i, n)$	$T(n)$
1. Divide the n elements into groups of 5.	
2. Find the median of each 5-element group.	$\Theta(n)$
3. Recursively $\textsc{Select}$ the median x of the $\lfloor \frac{n}{5} \rfloor$ group medians to be the pivot.	$T(n/5)$
4. Partition around the pivot x.	$\Theta(n)$
5. Let $k = rank(x)$.	
6. If $i = k$, then return x.	
7. If $i < k$, then recursively $\textsc{Select}$ the ith smallest element in first part.	$T(3n/4)$
8. If $i > k$, then recursively $\textsc{Select}$ the $(i - k)$th smallest element in last part.	$T(3n/4)$

Solving the recurrence

$$T(n) = T(\frac{1}{5}n) + T(\frac{3}{4}n) + \Theta(n)$$

Show $T(n) = O(n)$ with substitution: $T(n) \leq cn$:

$$\begin{aligned} T(n) &\leq \frac{1}{5}cn + \frac{3}{4}cn + \Theta(n) \\ &= \frac{19}{20}cn + \Theta(n) \\ &= cn - (\frac{cn}{20} - \Theta(n)) \leq cn \end{aligned}$$

if c is chosen large enough to handle both the $\Theta(n)$ and the initial conditions.

Conclusions

- Since the work at each level of recursion is a constant fraction ($\frac{19}{20}$) smaller, the work per level is a geometric series dominated by the linear work at the root.
- In practice, this algorithm runs slowly because the constant in front of n is large.
- The randomized algorithm is far more practical.
- Exercise: why not divide into groups of 3?