CS473 - Algorithms I

Lecture 7
Medians and Order Statistics

View in slide-show mode

Cevdet Aykanat and Mustafa Ozdal
Computer Engineering Department, Bilkent University

Medians and Order Statistics

i^{th} order statistic: i^{th} smallest element of a set of n elements

minimum: first order statistic

maximum: n^{th} order statistic

median: "halfway point" of the set

$$i = \lfloor (n+1)/2 \rfloor \text{ or } \lceil (n+1)/2 \rceil$$

Selection Problem

- *Selection problem*: Select the i^{th} smallest of n elements

- *Naïve algorithm*: Sort the input array A; then return A[i]

  $T(n) = \Theta(n\lg n)$

  using e.g. merge sort (but not quicksort)

- Can we do any better?

Selection in Expected Linear Time

- Randomized algorithm using divide and conquer

- Similar to randomized quicksort
  - *Like quicksort*: Partitions input array recursively
  - *Unlike quicksort*: Makes a single recursive call

    Reminder: Quicksort makes two recursive calls

- Expected runtime: $\Theta(n)$

  Reminder: Expected runtime of quicksort: $\Theta(n\lg n)$

Cevdet Aykanat and Mustafa Ozdal
Computer Engineering Department, Bilkent University

Selection in Expected Linear Time: Example 1

Select the 2nd smallest element:

6	10	13	5	8	3	2	11

$i = 2$

Partition the input array:

2	3	5	13	8	10	6	11

make a recursive call to select the 2nd smallest element in left subarray

Selection in Expected Linear Time: Example 2

Select the 7th smallest element:

6	10	13	5	8	3	2	11

$i = 7$

Partition the input array:

2	3	5	13	8	10	6	11

make a recursive call to select the 4th smallest element in right subarray

Selection in Expected Linear Time

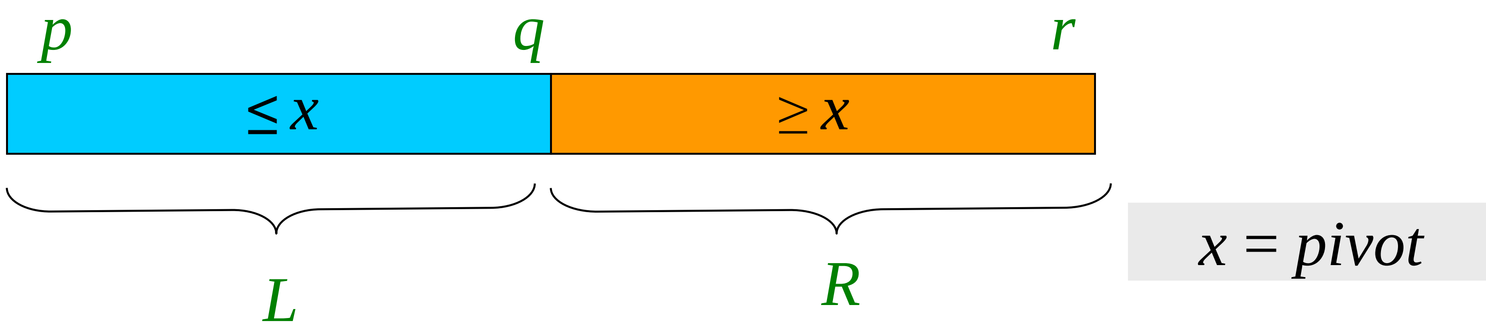
```
R-SELECT(A,p,r,i)
   if p = r then
       return A[p]
  q ← R-PARTITION(A, p, r)
  k ← q–p+1
  if i ≤ k then
       return R-SELECT(A, p, q, i)
else
       return R-SELECT(A, q+1, r, i-k)
```

$\leq x$ (k smallest elements)	$\geq x$

p $\qquad\qquad q \qquad\qquad r$

$x = pivot$

Selection in Expected Linear Time



- All elements in L $\leq$ all elements in R
- L contains $|L| = q-p+1 = k$ smallest elements of $A[p...r]$

if $i \leq |L| = k$ then

search L recursively for its i-th smallest element

else

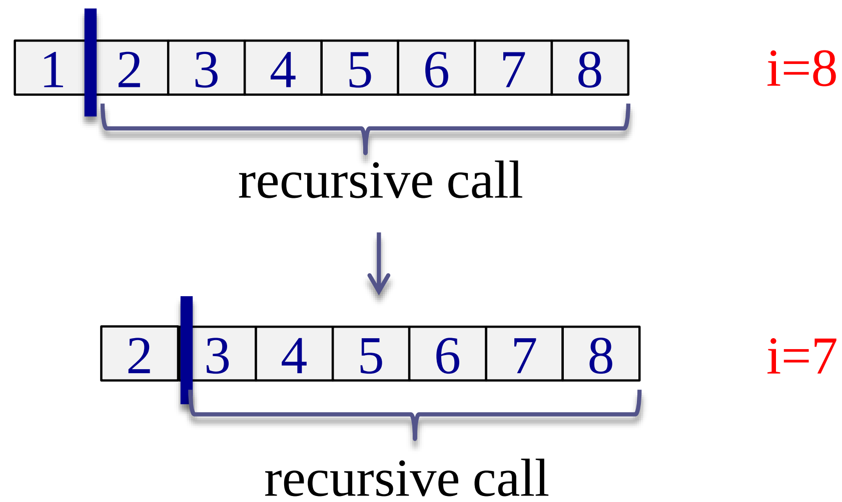
search R recursively for its $(i-k)$-th smallest element

Runtime Analysis

- ***Worst case:***

Imbalanced partitioning at every level

and the recursive call always to the larger partition



Runtime Analysis

- ***Worst case:***

  $T(n) = T(n-1) + \Theta(n)$

  ➔ $T(n) = \Theta(n^2)$

  Worse than the naïve method (based on sorting)

- ***Best case***: Balanced partitioning at every recursive level

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

  ➔ $T(n) = \Theta(n)$

- ***Avg case***: Expected runtime – need analysis

Reminder: Various Outcomes of H-PARTITION

$\mathbf{P(}\text{rank}(x) = i\mathbf{)} = 1/n$ for $1 \le i \le n$

x: pivot

|L|: size of left region

if $\text{rank}(x) = 1$ **then** $|L| = 1$

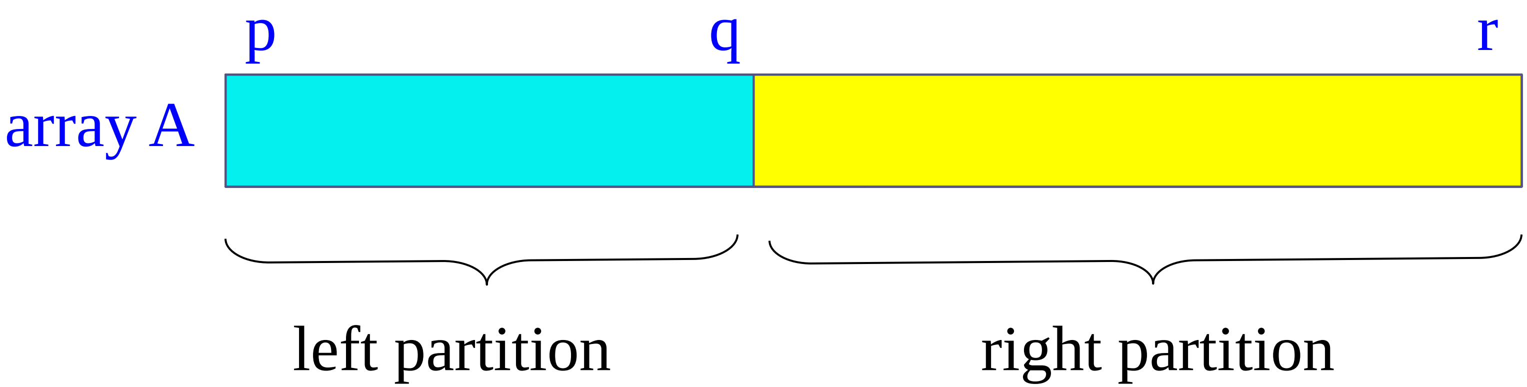
if $\text{rank}(x) > 1$ **then** $|L| = \text{rank}(x) - 1$

$\mathbf{P(}|L| = 1\mathbf{)} = \mathbf{P(}\text{rank}(x) = 1\mathbf{)} + \mathbf{P(}\text{rank}(x) = 2\mathbf{)}$ ⇒ $\mathbf{P(}|L| = 1\mathbf{)} = 2/n$

$\mathbf{P(}|L| = i\mathbf{)} = \mathbf{P(}\text{rank}(x) = i+1\mathbf{)}$ for $1 < i < n$ ⇒ $\mathbf{P(}|L| = i\mathbf{)} = 1/n$ for $1 < i < n$

Average Case Analysis of Randomized Select

- To compute the upper bound for the avg case, assume that the i^{th} element always falls into the larger partition.

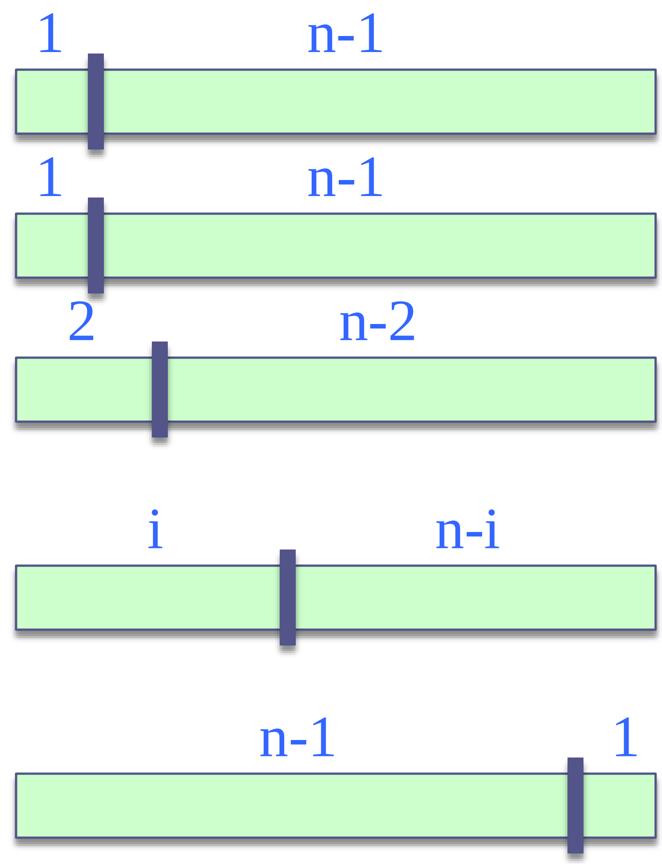


We will analyze the case where the recursive call is always made to the larger partition

➔ this will give us an upper bound for the avg case

Various Outcomes of H-PARTITION

rank(x)	prob.	T(n)
1	1/n	$\leq T(\max(1, n-1)) + \Theta(n)$
2	1/n	$\leq T(\max(1, n-1)) + \Theta(n)$
3	1/n	$\leq T(\max(2, n-2)) + \Theta(n)$
⋮	⋮	⋮
i+1	1/n	$\leq T(\max(i, n-i)) + \Theta(n)$
⋮	⋮	⋮
n	1/n	$\leq T(\max(n-1, 1)) + \Theta(n)$



Average-Case Analysis of Randomized Select

Recall: $P(|L|=i) = \begin{cases} 2/n & \text{for } i = 1 \\ 1/n & \text{for } i = 2,3,\ldots,n-1 \end{cases}$

Upper bound: Assume i-th element always falls into the larger part

$$T(n) \leq \frac{1}{n} T(\max(1, n-1)) + \frac{1}{n} \sum_{q=1}^{n-1} T(\max(q, n-q)) + O(n)$$

Note: $\frac{1}{n} T(\max(1, n-1)) = \frac{1}{n} T(n-1) = \frac{1}{n} O(n^2) = O(n)$

$$\therefore \quad T(n) \leq \frac{1}{n} \sum_{q=1}^{n-1} T(\max(q, n-q)) + O(n)$$

Average-Case Analysis of Randomized Select

$$\therefore\ T(n) \le \frac{1}{n} \sum_{q=1}^{n-1} T(\max(q, n-q)) + O(n)$$

$$\max(q, n-q) = \begin{cases} q & \text{if } q \ge \lceil n/2 \rceil \\ n-q & \text{if } q < \lceil n/2 \rceil \end{cases}$$

n is odd: $T(k)$ appears twice for $k = \lceil n/2 \rceil +1, \lceil n/2 \rceil +2, \ldots, n-1$

n is even: $T(\lceil n/2 \rceil)$ appears once $T(k)$ appears twice for

$k = \lceil n/2 \rceil +1, \lceil n/2 \rceil +2, \ldots, n-1$

Hence, in both cases: $\sum_{q=1}^{n-1} T(\max(q, n-q)) + O(n) \le 2 \sum_{q=\lceil n/2 \rceil}^{n-1} T(q) + O(n)$

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

Average-Case Analysis of Randomized Select

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

By substitution guess $T(n) = O(n)$

Inductive hypothesis: $T(k) \le ck, \ \forall\, k < n$

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

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

$$\frac{2c}{n}\left(\frac{1}{2}\, n\,(n\text{-}1) - \frac{1}{2}\left\lceil \frac{n}{2} \right\rceil \left(\frac{n}{2} - 1 \right) \right) + O(n)$$

Average-Case Analysis of Randomized Select

$$T(n) \le \frac{2c}{n}\left(\frac{1}{2}\,n(n\text{-}1) - \frac{1}{2}\left\lceil\frac{n}{2}\right\rceil\left(\frac{n}{2} - 1\right)\right) + \mathrm{O}(n)$$

$$\le c(n\text{-}1) - \frac{c}{4}n + \frac{c}{2} + \mathrm{O}(n)$$

$$= cn - \frac{c}{4}\,n - \frac{c}{2} + \mathrm{O}(n)$$

$$= cn - \left(\left(\frac{c}{4}\,n + \frac{c}{2}\right) - \mathrm{O}(n)\right)$$

$$\le cn$$

since we can choose c large enough so that **(** $\boldsymbol{cn/4+c/2}$ **) dominates O(*n*)**

Summary of Randomized Order-Statistic Selection

- Works fast: linear expected time
- Excellent algorithm in practise
- But, the worst case is very bad: $\Theta(n^2)$

Q: Is there an algorithm that runs in linear time in the worst case?

A: Yes, due to Blum, Floyd, Pratt, Rivest & Tarjan[1973]

Idea: Generate a good pivot recursively..

Selection in Worst Case Linear Time

SELECT(S, n, i) ▷ return ***i***-th element in set S with ***n*** elements

if $n \le 5$ then

SORT S and return the i-th element

DIVIDE S into $\lceil n/5 \rceil$ groups

▷ first $\lceil n/5 \rceil$ groups are of size 5, last group is of size n mod 5

FIND median set M=$\{m_1, \ldots, m_{\lceil n/5 \rceil}\}$ ▷ m_j: median of j-th group

$x \leftarrow$ SELECT(M, $\lceil n/5 \rceil$, $\lfloor (\lceil n/5 \rceil + 1)/2 \rfloor$)

PARTITION set S around the pivot x into L and R

if $i \le |L|$ then

return SELECT(L, $|L|$, i)

else

return SELECT(R, $n - |L|$, $i - |L|$)

Selection in Worst Case Linear Time - Example

Input: Array S and index i

Output: The i^{th} smallest value

S = {25 9 16 8 11 27 39 42 15 6 32 14 36 20 33 22 31 4 17 3 30 41
2 13 19 7 21 10 34 1 37 23 40 5 29 18 24 12 38 28 26 35 43}

Selection in Worst Case Linear Time - Example

Step 1: Divide the input array into groups of size 5

25	27	32	22	30	7	37	18	26
9	39	14	31	41	21	23	24	35
16	42	36	4	2	10	40	12	43
8	15	20	17	13	34	5	38	
11	6	33	3	19	1	29	28	

Selection in Worst Case Linear Time - Example

Step 2: Compute the median of each group → $\Theta(n)$

9	15	14	4	2	7	5	18	
8	6	20	3	13	1	23	12	26
11	**27**	**32**	**17**	**19**	**10**	**29**	**24**	**35**
16	42	33	31	30	34	40	28	43
25	39	36	22	41	21	37	38	

Let M be the set of the medians computed:

M = {11, 27, 32, 17, 19, 10, 29, 24, 35}

Selection in Worst Case Linear Time - Example

Step 3: Compute the median of the median group M

$x \leftarrow \text{SELECT}\left(M, |M|, \left\lfloor (|M|+1)/2 \right\rfloor\right)$ where $|M| = \lceil n/5 \rceil$

	9	15	14	4	2	7	5	18	
	8	6	20	3	13	1	23	12	26
M	11	27	32	17	19	10	29	24	35
	16	42	33	31	30	34	40	28	43
	25	39	36	22	41	21	37	38	

→ median = 24

The runtime of the recursive call: $T(|M|) = T\left(\lceil n/5 \rceil\right)$

Selection in Worst Case Linear Time - Example

Step 4: Partition the input array S around the median-of-medians x

S = {25 9 16 8 11 27 39 42 15 6 32 14 36 20 33 22 31 4 17 3 30 41
2 13 19 7 21 10 34 1 37 23 40 5 29 18 24 12 38 28 26 35 43}

Partition S around x = 24

Claim: Partitioning around x is guaranteed to be *well-balanced*.

Selection in Worst Case Linear Time - Example

Claim: Partitioning around x=24 is guaranteed to be *well-balanced*.

2	7	4	9	18	14	5	15	
13	1	3	8	12	20	23	6	26
19	10	17	11	24	32	29	27	35
30	34	31	16	28	33	40	42	43
41	21	22	25	38	36	37	39	

About half of the medians greater than x

about n/10

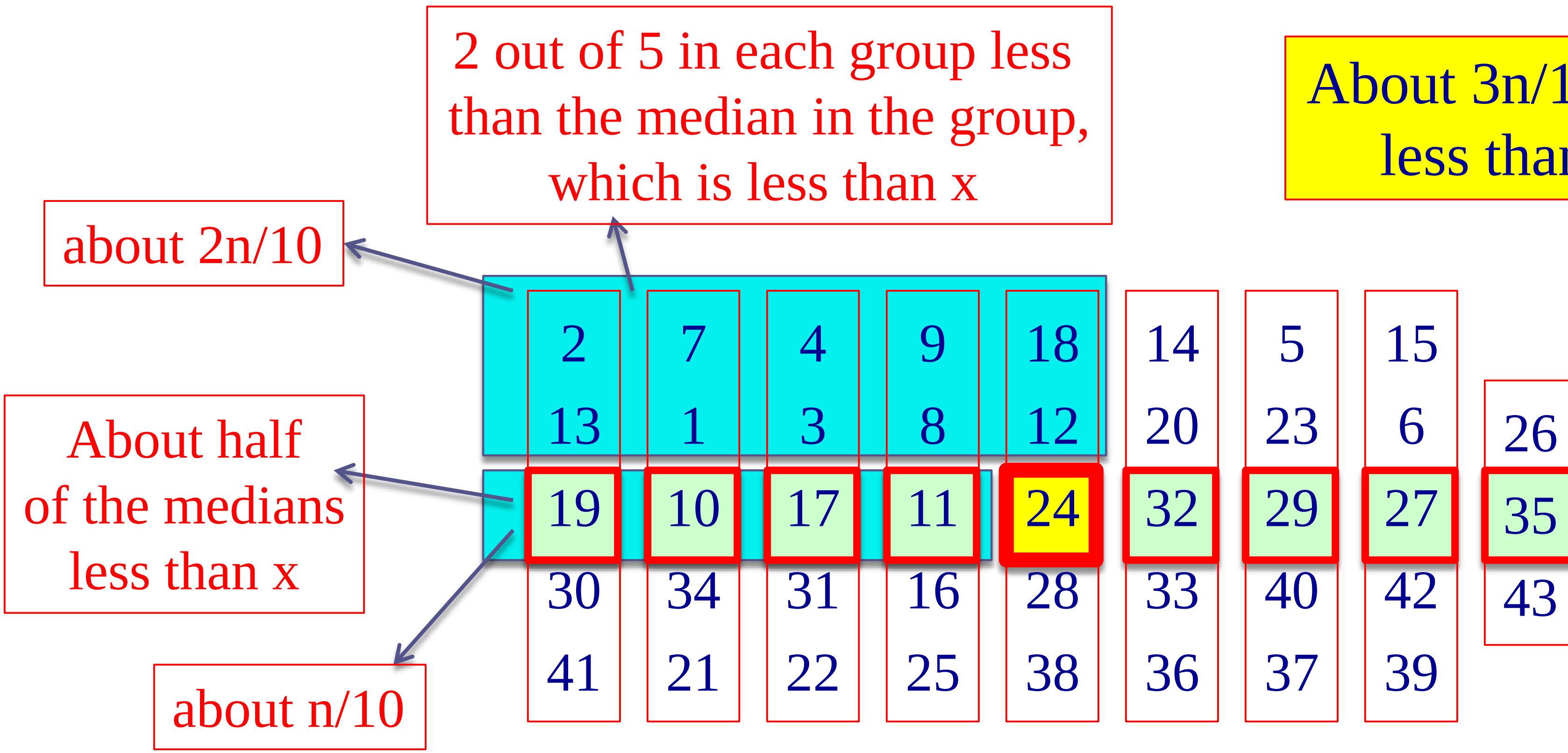
2 out of 5 in each group greater than the median in the group, which is greater than x

about 2n/10

About 3n/10 elts greater than x

Selection in Worst Case Linear Time - Example

Claim: Partitioning around x=24 is guaranteed to be *well-balanced*.



Selection in Worst Case Linear Time - Example

S = {25 9 16 8 11 27 39 42 15 6 32 14 36 20 33 22 31 4 17 3 30 41
2 13 19 7 21 10 34 1 37 23 40 5 29 18 24 12 38 28 26 35 43}

Partitioning S around x = 24 will lead to partitions of sizes ~3n/10 and ~7n/10 in the worst case.

Step 5: Make a recursive call to one of the partitions

if $i \le |L|$ then
    return SELECT(L, $|L|$, i)
else
    return SELECT(R, n–$|L|$, i–$|L|$)

Selection in Worst Case Linear Time

SELECT(S, n, i) ▷ return ***i***-th element in set S with ***n*** elements

if $n \leq 5$ then

SORT S and return the i-th element

DIVIDE S into $\lceil n/5 \rceil$ groups

▷ first $\lceil n/5 \rceil$ groups are of size 5, last group is of size n mod 5

FIND median set $M=\{m_1, \ldots, m_{\lceil n/5 \rceil}\}$ ▷ m_j: median of j-th group

$x \leftarrow$ SELECT(M, $\lceil n/5 \rceil$, $\lfloor (\lceil n/5 \rceil + 1)/2 \rfloor$)

PARTITION set S around the pivot x into L and R

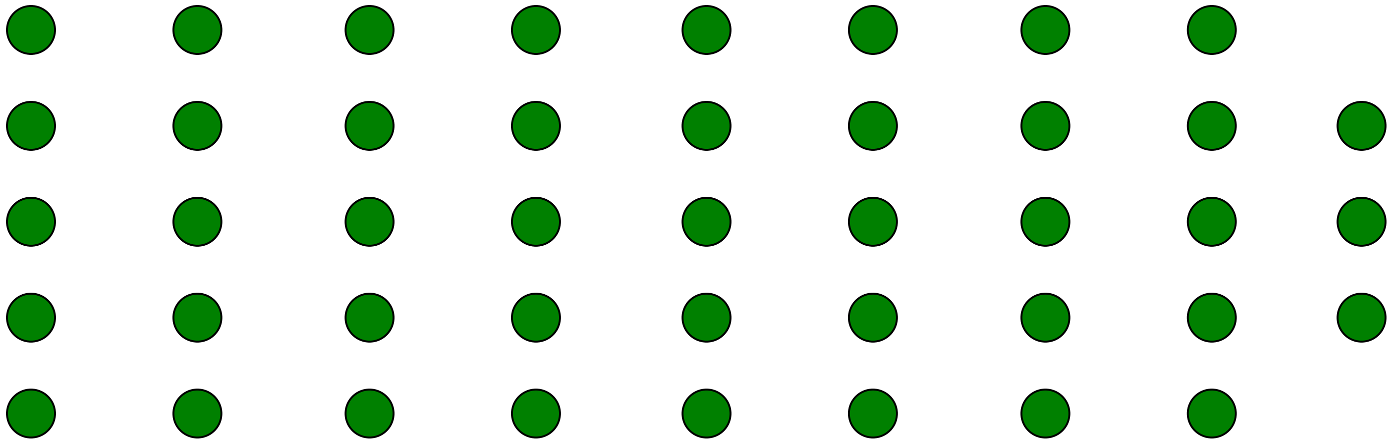
if $i \leq |L|$ then

return SELECT(L, $|L|$, i)

else

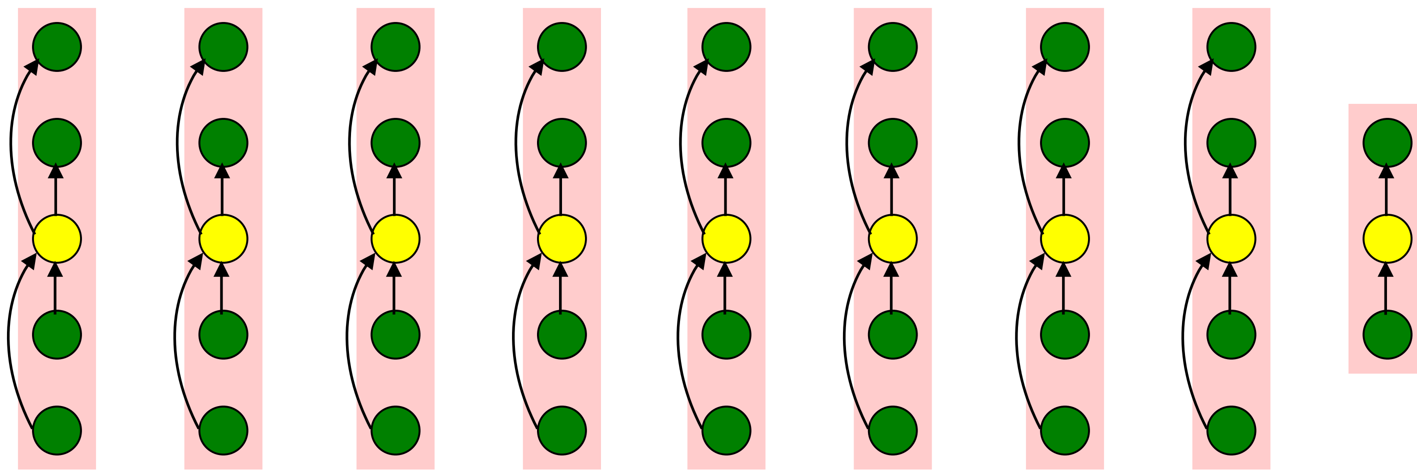
return SELECT(R, $n-|L|$, $i-|L|$)

Choosing the Pivot

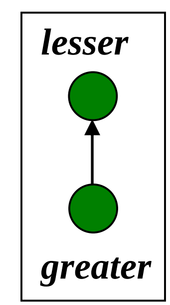


1. Divide S into groups of size 5

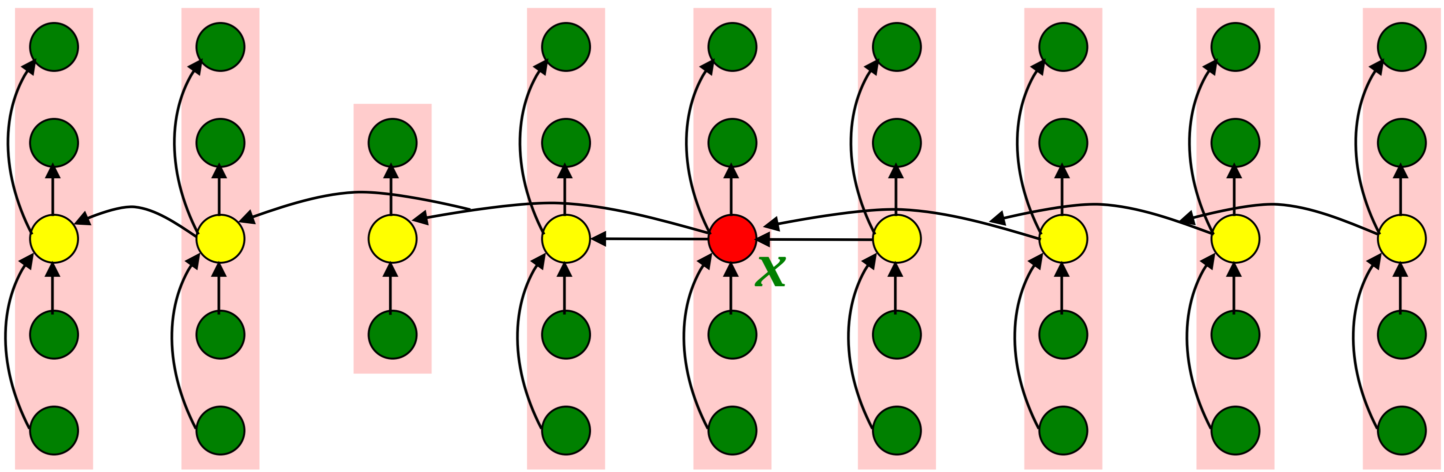
Choosing the Pivot



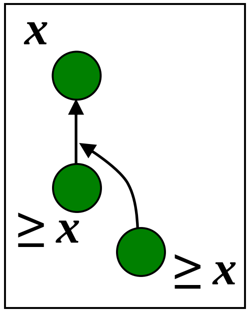
1. Divide S into groups of size 5
2. Find the median of each group



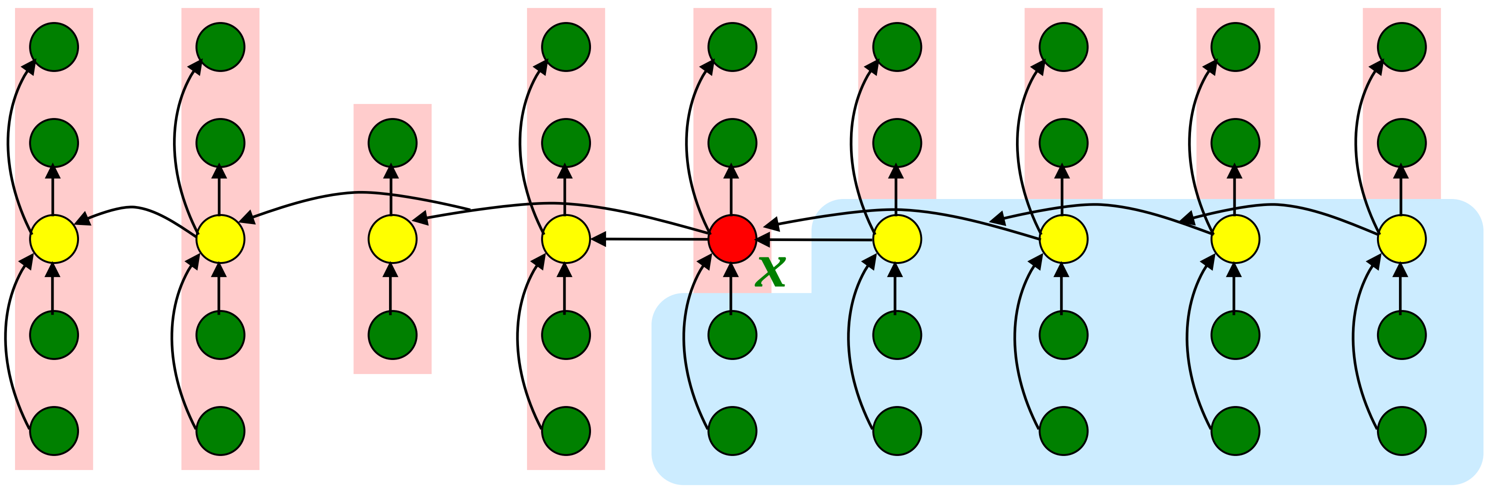
Choosing the Pivot



1. Divide S into groups of size 5
2. Find the median of each group
3. Recursively select the median x of the medians



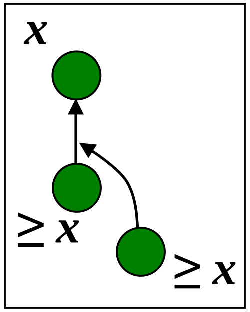
Choosing the Pivot



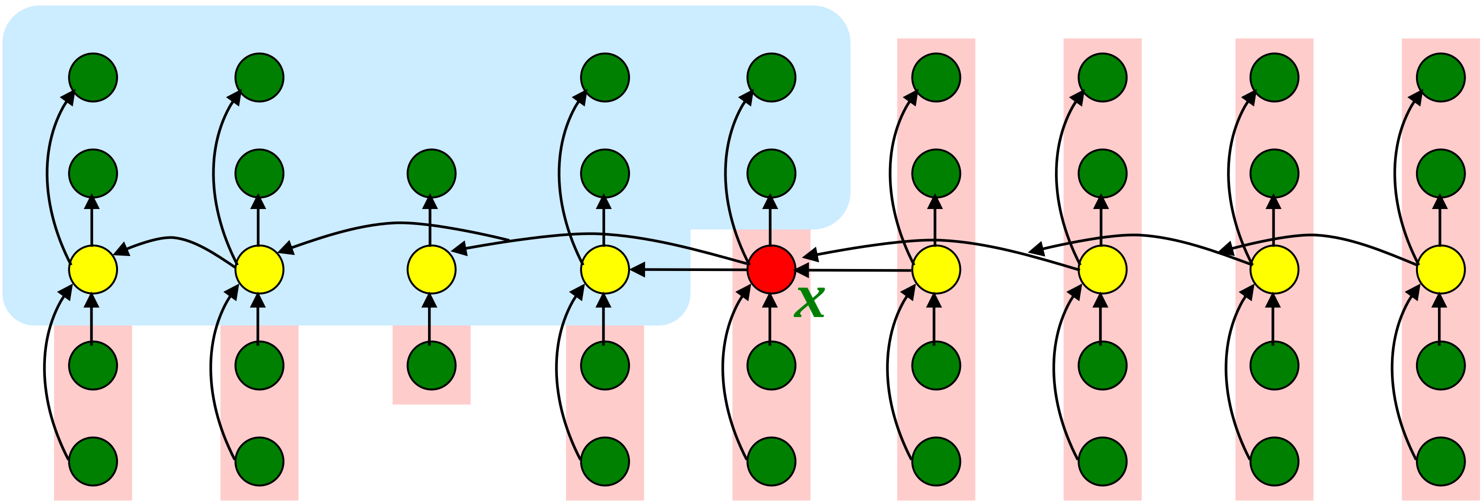
At least half of the medians $\geq x$

Thus $m = \left\lceil \lceil n/5 \rceil / 2 \right\rceil$ groups contribute 3 elements to R except possibly the last group and the group that contains x

$$|R| \geq 3\left(m - 2\right) \geq \frac{3n}{10} - 6$$



Analysis

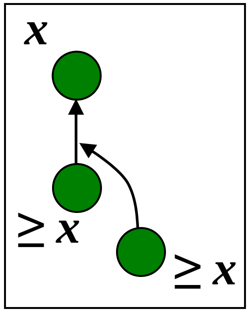


Similarly

$|L| \geq \frac{3n}{10} - 6$

Therefore, SELECT is recursively called on at most

$n - \left(\frac{3n}{10} - 6\right) = \frac{7n}{10} + 6$ elements



Selection in Worst Case Linear Time

SELECT(S, n, i) ▹ return ***i***-th element in set S with ***n*** elements

if $n \le 5$ then

SORT S and return the i-th element

$\Theta(n)$ { DIVIDE S into $\lceil n/5 \rceil$ groups

▹ first $\lfloor n/5 \rfloor$ groups are of size 5, last group is of size n mod 5

$\Theta(n)$ { FIND median set M={m_1 , …, $m_{\lceil n/5 \rceil}$} ▹ m_j: median of j-th group

$T(\lceil n/5 \rceil)$ { $x \leftarrow$ SELECT(M, $\lceil n/5 \rceil$, $\lfloor (\lceil n/5 \rceil+1)/2 \rfloor$)

$\Theta(n)$ { PARTITION set S around the pivot x into L and R

$T(\frac{7n}{10}+6)$ {

if $i \le |L|$ then

return SELECT(L, $|L|$, i)

else

return SELECT(R, n$-|L|$, $i-|L|$)

Selection in Worst Case Linear Time

Thus recurrence becomes

$$T(n) \le T\left(\left\lceil \frac{n}{5} \right\rceil\right) + T\left(\frac{7n}{10} + 6\right) + \Theta(n)$$

Guess $T(n) = O(n)$ and prove by induction

Inductive step: $T(n) \le c\lceil n/5 \rceil + c\,(7n/10+6) + \Theta(n)$

$\le cn/5 + c + 7cn/10 + 6c + \Theta(n)$

$= 9cn/10 + 7c + \Theta(n)$

$= cn - [c(n/10 - 7) - \Theta(n)] \le cn$ for large c

Work at each level of recursion is a constant factor (9/10) smaller