CS473-Algorithms I

Lecture 7

Median and Order Statistics

Cevdet Aykanat - Bilkent University
Computer Engineering Department

Order Statistics(Selection Problem)

- Select the i-th smallest of n elements

  (select the 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: Sort and index i-th element

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

$$= \Theta(nlgn)$$

using merge sort or heapsort(not quicksort)

Selection in Expected Linear Time

- Randomized algorithm
- Divide and conquer
- Similar to randomized quicksort

– Like quicksort: Partitions input array recursively

– Unlike quicksort:

    – Only works on one side of the partition

    – Quicksort works on both sides of the partition

– Expected running times:

  – SELECT: $E[n] = \Theta(n)$

  – QUICKSORT: $E[n] = \Theta(n \lg n)$

Selection in Expected Linear Time (example)

Select the $i = 7$th smallest

6	10	13	5	8	3	2	11	$i = 7$

Partition:

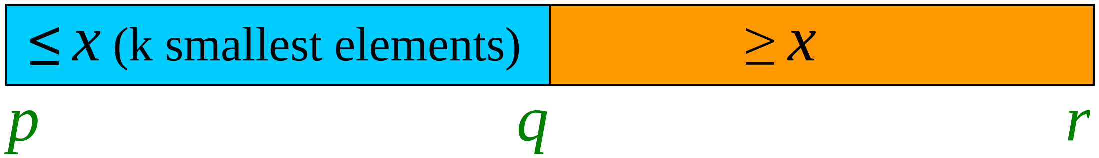
2	3	5	13	8	10	6	11

2	3	5	13	8	10	6	11

select the 7-3=4th smallest element recursively

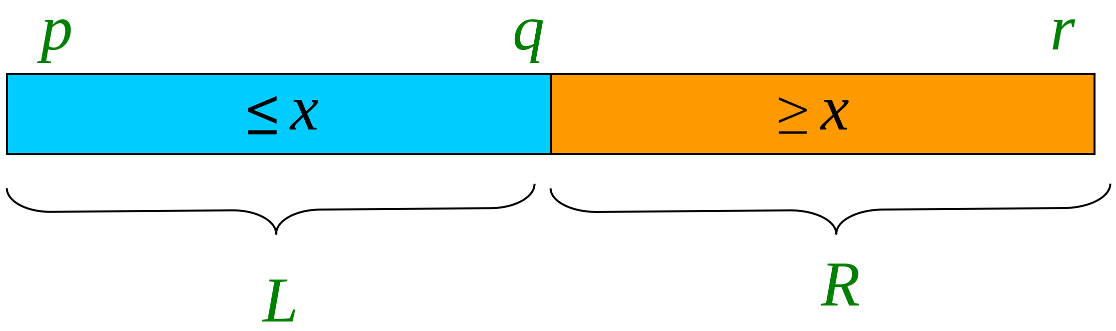
Selection in Expected Linear Time

R-SELECT(A,p,r,i)
  if $p = r$ then
    return $A[p]$
  $q \leftarrow$ R-PARTITION(A, p, r)
  $k \leftarrow q - p + 1$
  if $i \leq k$ then
    return R-SELECT(A, p, q, i)
else
    return R-SELECT(A, $q+1$, r, $i-k$)



$x = pivot$

Selection in Expected Linear Time



$x = pivot$

- 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

Selection in Expected Linear Time

- Excellent algorithm in practise
- Worst-case: $T(n) = T(n-1)+\Theta(n) \Rightarrow T(n) = \Theta(n^2)$
  - Worse than sorting
  - e.g., occurs when
    - $i = 1$ and
    - Partition returns $q = r - 1$ at each level of recursion
- Best-case: $T(n) = T(n/2)+\Theta(n) \Rightarrow T(n) = \Theta(n)$

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

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

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

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

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

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

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

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

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

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

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 **($cn/4 + c/2$) 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 $\lfloor n/5 \rfloor$ 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

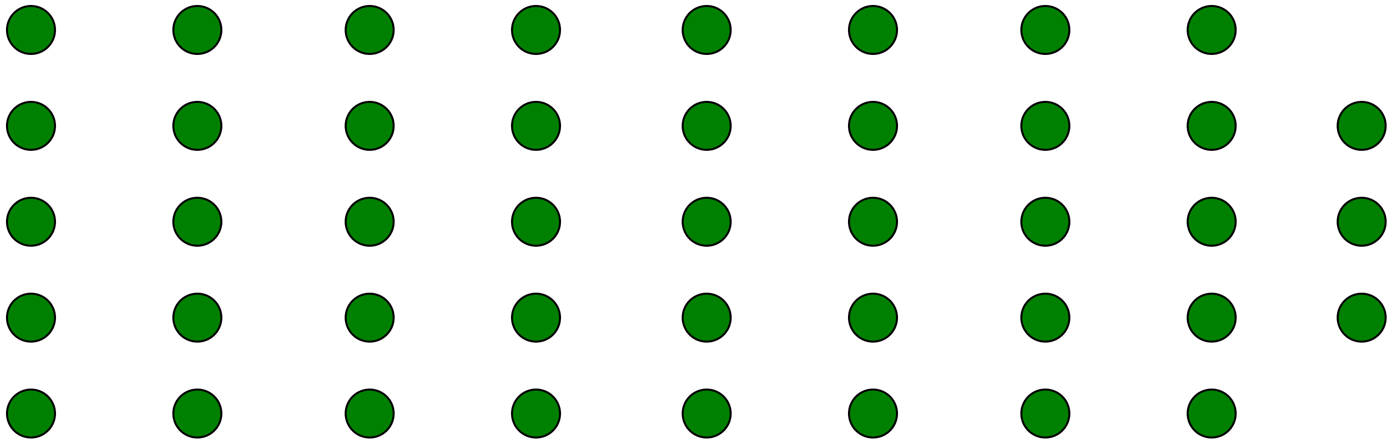
if $i \le |L|$ then

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

else

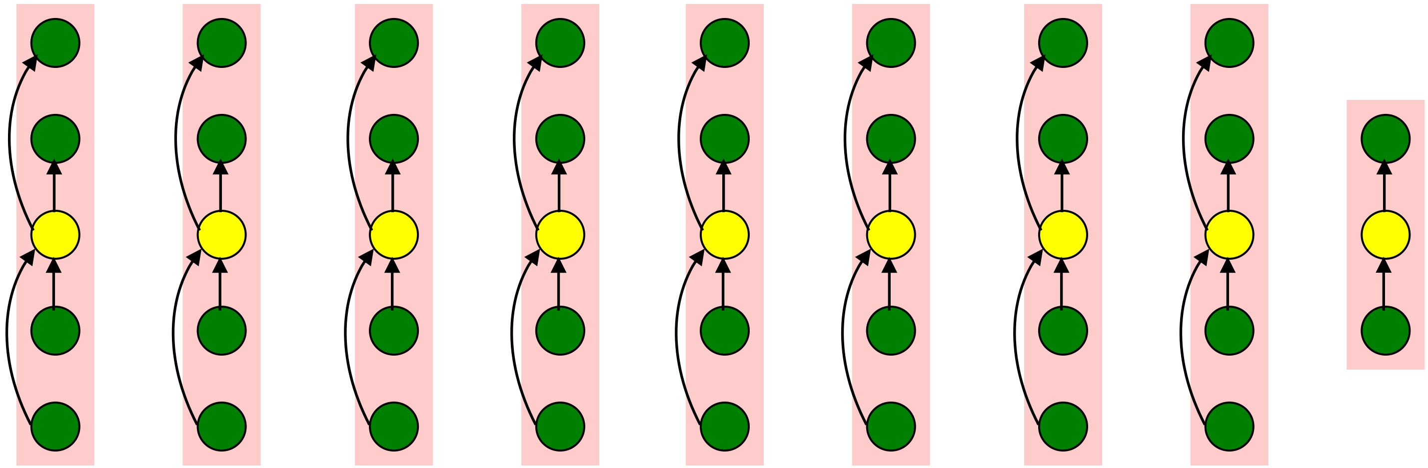
return SELECT(R, n$-|L|$, $i-|L|$)

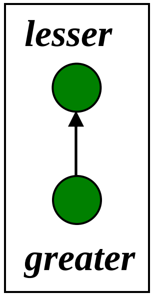
Choosing the Pivot



1. Divide S into groups of size 5

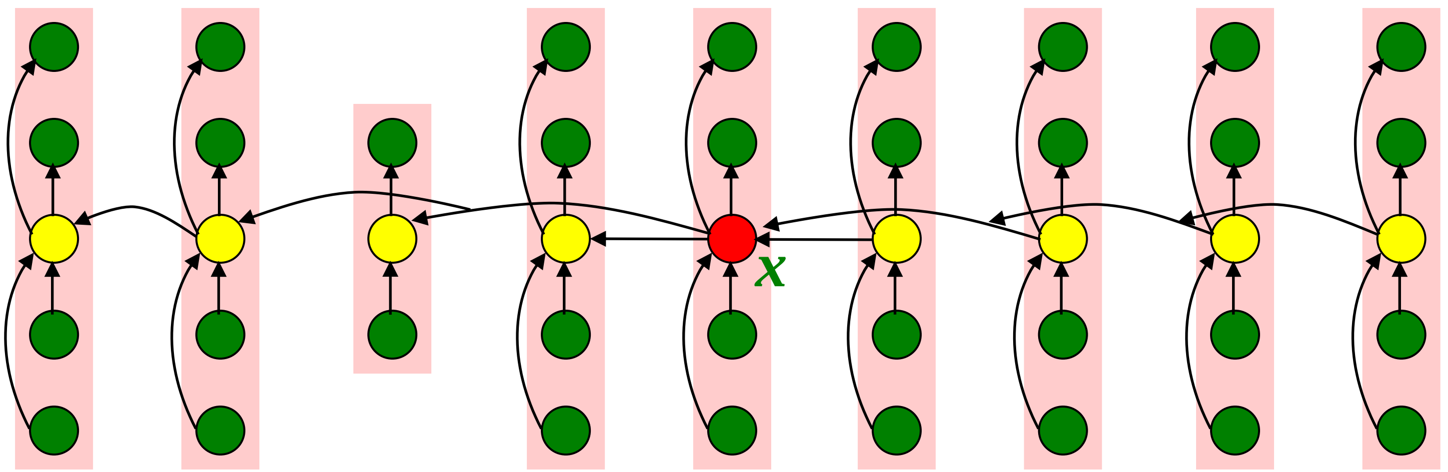
Choosing the Pivot



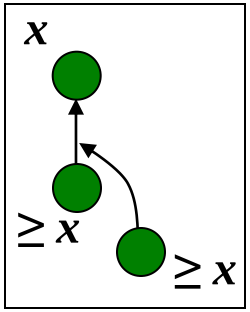


1. Divide S into groups of size 5
2. Find the median of each group

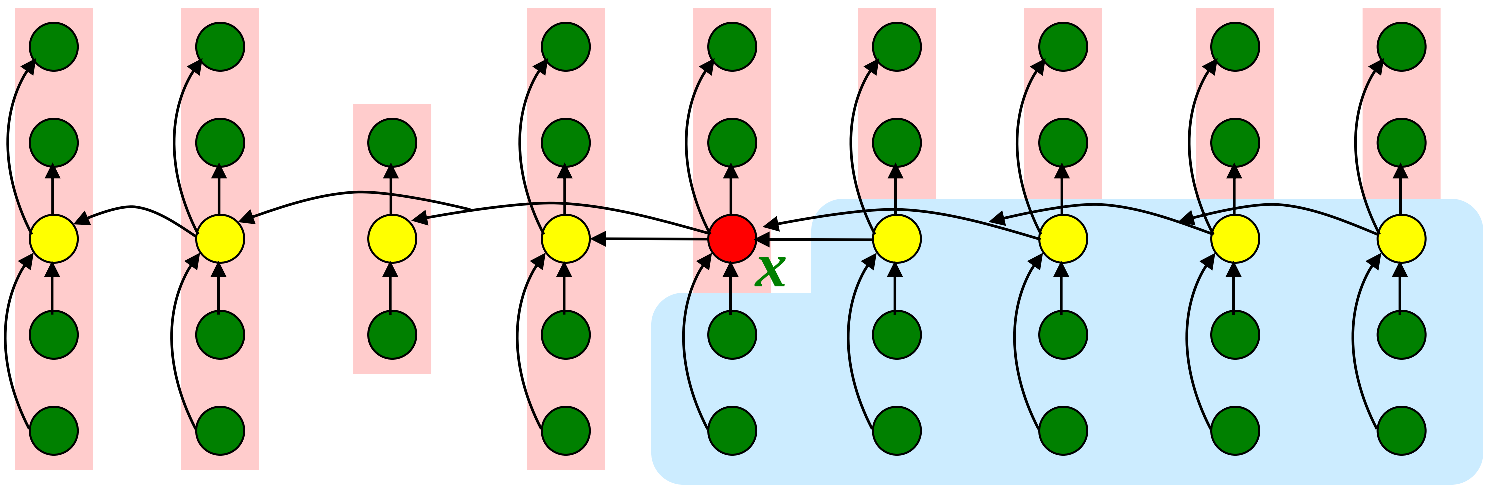
Choosing the Pivot

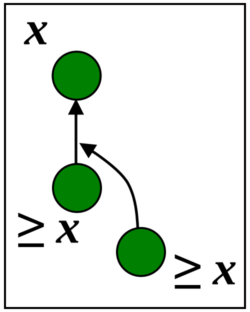


1. Divide S into groups of size 5
2. Find the median of each group
3. Recursively select the median x of the medians



Choosing the Pivot



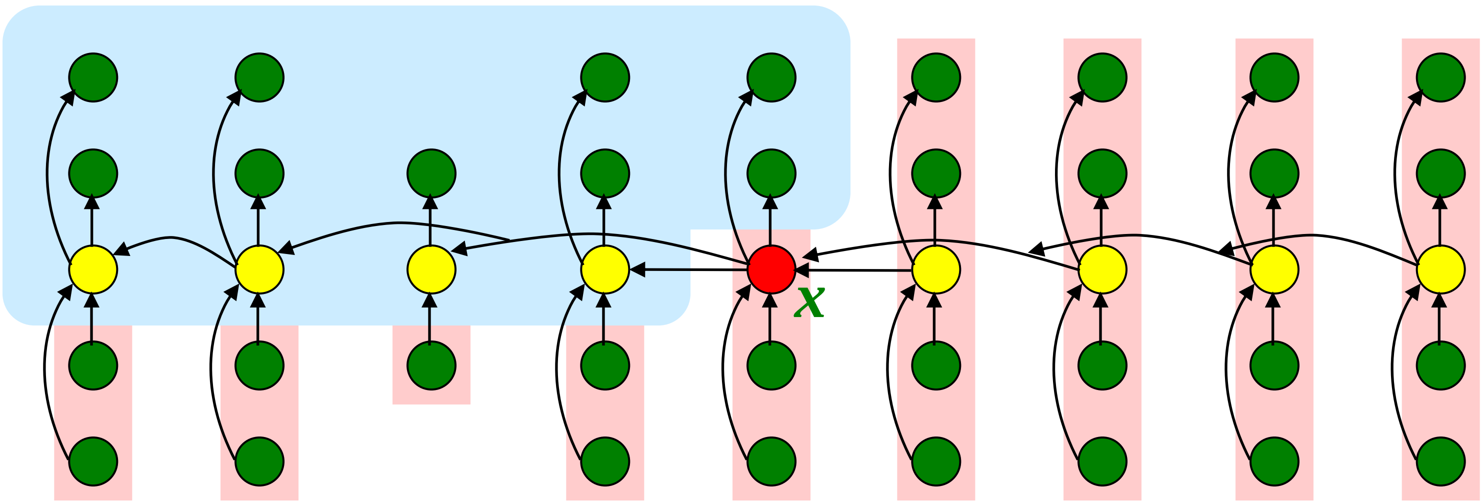


At least half of the medians $\geq x$

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

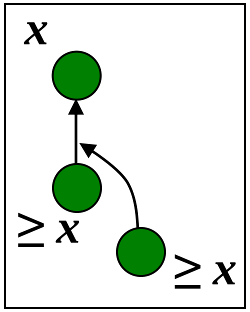


Similarly

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

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

$\Theta(1)$ { if $n \leq 5$ then
SORT S and return the i-th element

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

$\Theta(n)$ { ▹ 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 $i \leq |L|$ 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