Chap 9 – Medians and Order Statistics

9.1 Minimum and maximum

9.2 Selection in expected linear time

9.3 Selection in worst-case linear time

9.1 Minimum and maximum

- Minimum (or maximum)

  THEOREM Finding the minimum of n elements needs exactly $n - 1$ comparisons.

  Upper bound (sufficient)

  MINIMUM(A, n)
  $min = A[1]$
  for $i = 2$ **to** n
  if $A[i] < min$ **then** $min = A[i]$
  return min

  Clearly, this algorithm takes $n - 1$ comparisons and, hence, establishes an upper bound for the minimum problem.

9.1 Minimum and maximum

- Minimum (or maximum)

  THEOREM (Cont'd)

  Lower bound (necessary)

  For each comparison, one wins and the other loses.

  Only the minimum never loses.

  Thus, at least $n - 1$ comparisons are needed.

  Comment

  Therefore, algorithm MINIMUM is an optimal algorithm for the minimum problem.

9.1 Minimum and maximum

- Simultaneous minimum and maximum

  The algorithm below takes $2n - 3$ comparisons.

  MINMAX(A, n)

  Find min of n elements

  Find max of the remaining $n - 1$ elements

  The next algorithm takes $2n - 2$ comparisons in the worst case

  MINMAX(A, n)

  min = max = $A[1]$

  for i = 2 **to** n // worst case input:$1, 2, \ldots, n$

  if $A[i]$ < min **then** min = $A[i]$

  else if $A[i]$ > max **then** max = $A[i]$

9.1 Minimum and maximum

- Simultaneous minimum and maximum

  THEOREM Finding the minimum and maximum of n elements needs exactly $\lceil 3n/2 \rceil - 2$ comparisons in the worst case.

  Upper bound

  MINMAX(A, n)

  If n is odd, set both min and max to $A[1]$

  If n is even

  compare $A[1]$ and $A[2]$ to set min and max

  For each pair of the elements left

  compare them

  compare the smaller to min, and set min accordingly

  compare the larger to max, and set max accordingly

9.1 Minimum and maximum

- Simultaneous minimum and maximum

  THEOREM (Cont'd)

  # of comparisons done by MINMAX:

  n is even: $1 + 3 \cdot (n-2)/2 = 3n/2 - 2 = \lceil 3n/2 \rceil - 2$

  n is odd: $3 \cdot (n-1)/2 = 3n/2 - 3/2 = \lceil 3n/2 \rceil - 2$

  Thus, algorithm MINMAX establishes an upper bound.

  Oracular lower bound (Ex. 9.1-2)

  # of elements that are not yet compared

  ever lost and won

  State: (a, b, c, d)

  never lost ..... never won

9.1 Minimum and maximum

- Simultaneous minimum and maximum

  THEOREM (Cont'd)

  Every minmax algorithm must do the transition

  $(n, 0, 0, 0) \rightarrow (0, 1, 1, n-2)$

  Element never won is the min (or max).

  Element never lost is the max (or, min).

  Looking for: The minimum # of steps needed in the worst case

  Oracle arguments (or, adversary arguments)

  - Anyone who never lost will win.
  - Anyone who never won will lost.
  - For others, the result is arbitrary, as long as consistent.

9.1 Minimum and maximum

- Simultaneous minimum and maximum

  THEOREM (Cont'd)

  Consider these transitions:

  (a, b, c, d)

  $\rightarrow^{ab}$ $(a-1, b, c+1, d)$, b won; $(a-1, b, c, d+1)$ otherwise

  $\rightarrow^{ac}$ $(a-1, b+1, c, d)$, c lost; $(a-1, b, c, d+1)$, "

  $\rightarrow^{bc}$ (a, b, c, d), b won, c lost; $(a, b-1, c-1, d+2)$, "

  $\rightarrow^{bd}$ (a, b, c, d), b won; $(a, b-1, c, d+1)$, "

  $\rightarrow^{cd}$ (a, b, c, d), c lost; $(a, b, c-1, d+1)$, "

  Note that the left-column transitions occur in the worst case ($\because$ d doesn't change), and the right-column in the best case.

9.1 Minimum and maximum

- Simultaneous minimum and maximum

  THEOREM (Cont'd)

  Interested in the worst case

  The oracle tells us to ignore the right-column.

  Interested in the lower bound

  The last 3 transitions on the left-column and the transition $(a, b, c, d) \rightarrow^{2d} (a, b, c, d)$ needn't be considered ($\because$ state doesn't change).

  Thus, we need only consider the six transitions listed on the next page.

9.1 Minimum and maximum

- Simultaneous minimum and maximum

THEOREM (Cont'd)

$$(a,b,c,d) \to^{2a} (a-2,b+1,c+1,d)$$
$$\to^{2b} (a,b-1,c,d+1)$$
$$\to^{2c} (a,b,c-1,d+1)$$
$$\to^{ab} (a-1,b,c+1,d), b \text{ won}$$
$$\to^{ac} (a-1,b+1,c,d), c \text{ lost}$$
$$\to^{ad} (a-1,b+1,c,d) \text{ or } (a-1,b,c+1,d)$$

For even n, the minimum number of steps is

$$(n,0,0,0) \xrightarrow[n/2 \text{ times}]{}^{2a} (0,n/2,n/2,0) \xrightarrow[n-2 \text{ times}]{}^{2b\ 2c} (0,1,1,n-2)$$

9.1 Minimum and maximum

- Simultaneous minimum and maximum

THEOREM (Cont'd)

$\because$ Using ab, ac, ad to move elements from a to b/c is too slow

Total: $n/2 + (n-2) = 3n/2 - 2 = \lceil 3n/2 \rceil - 2$

For odd n, the minimum number of steps is

$$(n,0,0,0) \xrightarrow[(n-1)/2 \text{ times}]{} ^{2a} \left(1, \frac{n-1}{2}, \frac{n-1}{2}, 0\right)$$

$$\xrightarrow[n-3 \text{ times}]{} ^{2c}_{2b} (1,1,1,n-3) \xrightarrow[2 \text{ times}]{} (0,1,1,n-1)$$

$\because$ The last two steps: first, use $ab, ac,$ or ad; then $2b$ or $2c$

Total: $(n-1)/2 + (n-3) + 2 = 3(n-1)/2 = \lceil 3n/2 \rceil - 2$

9.2 Selection in expected linear time

- The selection problem
  - Find the ith smallest element in a set of n elements
  - Minimum $\Rightarrow$ find the 1^{st} smallest

    Maximum $\Rightarrow$ find the n^{th} smallest

    Median n is odd $\Rightarrow$ find the $\lceil n/2 \rceil^{th}$ smallest

    n is even $\Rightarrow$ find the $\lceil n/2 \rceil^{th}$ smallest (lower median)

    or, the $1 + \lceil n/2 \rceil^{th}$ smallest (upper median)
  - A sorting-based algorithm

    SELECT(A, n, i)

    Sort the array $A[1..n]$ into non-decreasing order

    return $A[i]$

9.2 Selection in expected linear time

- Expected linear time algorithm

RANDOMIZED-SELECT(A, p, r, i)

if $p == r$ **return** $A[p]$ // may be omitted; i must be 1

$q =$ RANDOMIZED-PARTITION(A, p, r)

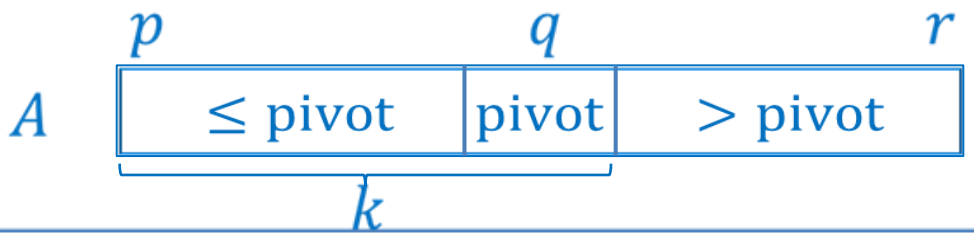
$k = q - p + 1$

if $i == k$ **return** $A[q]$

else if $i < k$

return RANDOMIZED-SELECT$(A, p, q-1, i)$

else return RANDOMIZED-SELECT$(A, q+1, r, i-k)$



9.2 Selection in expected linear time

- Worst-case running time (extremely unlucky case)

  Let $T(n) =$ the worst-case running time of RANDOMIZED-SELECT on n elements, then

$$T(n) = \max_{1 \le k \le n} \{T(k-1), T(n-k)\} + \Theta(n)$$
$$= \max_{0 \le k \le n-1} T(k) + \Theta(n)$$
$$= T(n-1) + \Theta(n)$$
$$\Rightarrow T(n) = \Theta(n^2)$$

  Note that this is independent of i – for any i, the worst case behaves like this:

$$i(n) \to i(n-1) \to \cdots \to i(i) \to i-1(i-1) \to \cdots \to 1(1)$$

9.2 Selection in expected linear time

- Expected running time

  Define random variables

  $T(n) =$ the running time of RANDOMIZED-SELECT on n elements

  $X_k = \mathrm{I}\{\text{subarray } A[p..q] \text{ has exactly } k \text{ elements}\}$

  Then,

  $$\mathrm{E}[X_k] = \Pr\{\text{subarray } A[p..q] \text{ has exactly } k \text{ elements}\}$$
  $$= \Pr\{\text{the pivot is the } k\text{th smallest}\} = 1/n$$

  Lower bound

  $T(n) \geq \Theta(n)$ $\because$ due to partition

  $\Rightarrow \mathrm{E}[T(n)] \geq \mathrm{E}[\Theta(n)] = \Theta(n)$

  $\Rightarrow \mathrm{E}[T(n)] = \Omega(n)$

9.2 Selection in expected linear time

- Expected running time

  Upper bound (independent of i)

  $\because$ may be $i = k$ or the ith smallest is in the smaller part

$$T(n) \leq \sum_{k=1}^{n} X_k \cdot \left(T(\max(k-1, n-k)) + O(n)\right)$$

$$= \sum_{k=1}^{n} X_k \cdot T(\max(k-1, n-k)) + O(n)$$

  Taking expected values give

$$\mathrm{E}[T(n)] \leq \mathrm{E}\left[\sum_{k=1}^{n} X_k \cdot T(\max(k-1, n-k)) + O(n)\right]$$

9.2 Selection in expected linear time

- Expected running time

$$\mathrm{E}[T(n)] \le \mathrm{E}\left[\sum_{k=1}^{n} X_k \cdot T(\max(k-1, n-k)) + O(n)\right]$$

$$= \sum_{k=1}^{n} \mathrm{E}[X_k \cdot T(\max(k-1, n-k))] + O(n)$$

$$= \sum_{k=1}^{n} \mathrm{E}[X_k] \cdot \mathrm{E}[T(\max(k-1, n-k))] + O(n)$$

$\because X_k$ and $T(\max(k-1, n-k))$ are independent

$$= \sum_{k=1}^{n} \frac{1}{n} \cdot \mathrm{E}[T(\max(k-1, n-k))] + O(n)$$

9.2 Selection in expected linear time

- Expected running time

$$\mathrm{E}[T(n)] \le \frac{1}{n}\sum_{k=1}^{n} \mathrm{E}[T(\max(k-1, n-k))] + O(n)$$

$$\le \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} \mathrm{E}[T[k]] + O(n)$$

$=$, if n is even; $<$, if n is odd

e.g. $n = 4$, the terms are $T(3), T(2), T(2), T(3)$

$n = 5$, the terms are $T(4), T(3), T(2), T(3), T(4)$

We show that $\mathrm{E}[T(n)] = O(n)$ by substitution.

Assume that $\mathrm{E}[T(n)] \le cn$

9.2 Selection in expected linear time

- Expected running time

$$\mathrm{E}[T(n)] \le \frac{2}{n}\sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + an$$

$$= \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2 \rfloor - 1} k\right) + an$$

$$= \frac{2c}{n}\left(\frac{(n-1)n}{2} - \frac{(\lfloor n/2 \rfloor - 1)\lfloor n/2 \rfloor}{2}\right) + an$$

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

$$= \frac{c}{n}\left(\frac{3n^2}{4} + \frac{n}{2} - 2\right) + an$$

9.2 Selection in expected linear time

- Expected running time

$$\mathrm{E}[T(n)] = c\left(\frac{3n}{4} + \frac{1}{2} - \frac{2}{n}\right) + an \le \frac{3cn}{4} + \frac{c}{2} + an$$
$$= cn - \left(\frac{cn}{4} - \frac{c}{2} - an\right)$$
$$\le cn$$

as long as

$$\frac{cn}{4} - \frac{c}{2} - an \ge 0 \Rightarrow n\left(\frac{c}{4} - a\right) \ge \frac{c}{2}$$

By picking $c > 4a$, we have

$$n\left(\frac{c}{4} - a\right) \ge \frac{c}{2} \quad \text{as long as} \quad n \ge \frac{2c}{c - 4a}$$

9.3 Selection in worst-case linear time

- Worst-case linear time algorithm

SELECT(A, n, i)

1 Divide $A[1..n]$ into $s = \lceil n/5 \rceil$ groups, each has 5 elements, except for the last group, which may have < 5 elements.

2 Find the median of each group by insertion sort

3 $x =$ SELECT$(A, s, \lceil s/2 \rceil)$ // median of medians

4 Partition A using x as the pivot, let $k = \text{rank}(x)$

5 **if** $i = k$ **then** return x

if $i < k$ **then** SELECT$(A, k-1, i)$

else SELECT$(A, n-k, i-k)$

Clearly, step 1 takes $\Theta(n)$ time. So are steps 2 and 4.

9.3 Selection in worst-case linear time

- Analysis of SELECT

  Let $T(n)$ = worst-case running time of SELECT on n elements

  Then

  $T(n) = T(\lceil n/5 \rceil) + T(\max\{k-1, n-k\}) + \Theta(n)$

  Clearly, $T(n) = \Omega(n)$

  For upper bound, how large can $\max\{k-1, n-k\}$ be?

  Key point

  Since the pivot x is chosen carefully, k ought to be closer to $n/2$ than to the two end points.

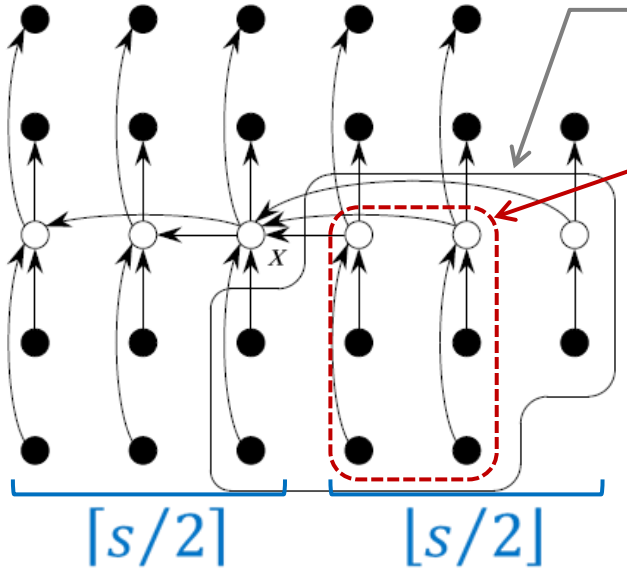
  Assuming that the n elements are distinct, w shall see:

  $\max\{k-1, n-k\} \leq 7n/10 + 6, \text{if } n \geq 140$

9.3 Selection in worst-case linear time

- Analysis of SELECT

  Assuming that the n elements are distinct



$$\#\text{ of elements} > x$$
$$\geq 3(\lfloor s/2 \rfloor - 1)$$
$$\geq 3(\lceil s/2 \rceil - 2) \quad \text{(see below)}$$
$$\geq 3\left(\left\lceil \frac{1}{2}\left\lceil \frac{n}{5} \right\rceil \right\rceil - 2\right)$$
$$\geq 3n/10 - 6$$
$$\therefore (\#\text{ of elements} < x) \leq 7n/10 + 6$$

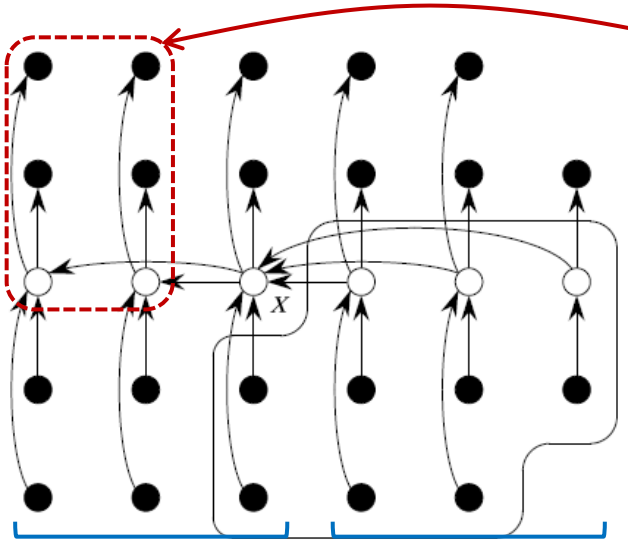
Note: $3(\lfloor s/2 \rfloor - 1) = 3(\lceil s/2 \rceil - 2)$, if s is odd

$> 3(\lceil s/2 \rceil - 2)$, if s is even

9.3 Selection in worst-case linear time

- Analysis of SELECT

Symmetrically,



# of elements $< x$

$\geq 3(\lceil s/2 \rceil - 1)$

$\geq 3(\lceil s/2 \rceil - 2)$

$\geq 3n/10 - 6$

$\therefore$ (# of elements $> x$) $\leq 7n/10 + 6$

$|< x|$ $|> x|$

Thus, we have

$$T(n) = T(\lceil n/5 \rceil) + T(\max\{k-1, n-k\}) + \Theta(n)$$
$$\leq T(\lceil n/5 \rceil) + T(7n/10 + 6) + \Theta(n)$$

9.3 Selection in worst-case linear time

- Analysis of SELECT

  We show that $T(n) = O(n)$ by substitution.

  Assume that $T(n) \leq cn$

  Then,

$$
\begin{aligned}
T(n) &\leq c\lceil n/5 \rceil + c(7n/10 + 6) + an \\
&\leq c(n/5 + 1) + c(7n/10 + 6) + an \\
&= 9cn/10 + 7c + an \\
&= cn - (cn/10 - 7c - an) \\
&\leq cn
\end{aligned}
$$

  as long as

$$cn/10 - 7c - an \geq 0 \Rightarrow cn - 70c - 10an \geq 0$$

9.3 Selection in worst-case linear time

- Analysis of SELECT

  Wanted: $cn - 70c - 10an \geq 0 \Rightarrow c(n-70) \geq 10an$

  If $n > 70$, then

  $c(n-70) \geq 10an \Rightarrow c \geq 10an/(n-70)$

  Assume arbitrarily that $n \geq 140$, then $2 \geq n/(n-70)$

  Thus, $20a \geq 10an/(n-70)$

  So, choosing $c \geq 20a$ guarantees $T(n) \leq cn \quad \forall n \geq 140$

  Comment

  Any integer > 70 will do.

  For example, assume that $n \geq 71$, then $71 \geq n/(n-70)$

  So, choose $c \geq 710a$

9.3 Selection in worst-case linear time

- Analysis of SELECT

  Alternative analysis (See p.19)

  Wanted: $cn - 70c - 10an \geq 0 \Rightarrow (c - 10a)n \geq 70c$

  By picking $c > 10a$, we have

  $(c - 10a)n \geq 70c$ as long as $n \geq \frac{70c}{c-10a}$

- An easier upper-bound analysis of SELECT

  $T(n) \leq T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n)$

  $\leq T(0.21n) + T(0.75n) + O(n), \quad \forall n \geq 120$

  since

  $\lceil n/5 \rceil \leq n/5 + 1 \leq 0.21n$, if $n \geq 100$

  $7n/10 + 6 \leq 0.75n$, if $n \geq 120$

9.3 Selection in worst-case linear time

- An easier upper-bound analysis of SELECT

  Or,

  $$T(n) \le T(\lceil n/5 \rceil) + T(7n/10 + 6) + O(n)$$
  $$\le T(0.201n) + T(0.701n) + O(n), \quad \forall n \ge 6000$$

  since

  $\lceil n/5 \rceil \le n/5 + 1 \le 0.201n$, if $n \ge 1000$

  $7n/10 + 6 \le 0.701n$, if $n \ge 6000$

  It follows from the following lemma that $T(n) = O(n)$

  LEMMA (prune-and-search)

  $T(n) = T(\alpha n) + T(\beta n) + \Theta(n)$, $\alpha + \beta < 1$

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

9.3 Selection in worst-case linear time

- An easier analysis of SELECT

  Proof of the lemma

  Clearly, $T(n) = \Omega(n)$.

  For upper bound, assume that $T(n) \leq cn$

  Then,

$$\begin{aligned} T(n) &= T(\alpha n) + T(\beta n) + \Theta(n) \\ &\leq (\alpha + \beta)cn + dn \\ &= cn - [(1 - \alpha - \beta)cn - dn] \\ &\leq cn \end{aligned}$$

  as long as

  $(1 - \alpha - \beta)cn - dn \geq 0 \Rightarrow c \geq d/(1 - \alpha - \beta)$, $\forall n \geq 0$