# *Medians and Order Statistics*

交通大學 資訊工程系

# *Introduction*[1]

- **Order statistic**
  - The *i*th ***order statistic*** of a set of *n* elements is the *i*th smallest element.
  - ***Minimum***: The first order statistic ($i = 1$).
  - ***Maximum***: The *n*th order statistic ($i = n$).
- ***Median***: The "halfway point" of the set.
  - *n* is odd: Occurring at $i = (n + 1)/2$.
  - *n* is even: Occurring at $i = n/2$ and $i = n/2 + 1$.
  - Regardless of the parity:
    - ***Lower median***: $i = \lfloor (n + 1)/2 \rfloor$.
    - ***Upper median***: $i = \lceil (n + 1)/2 \rceil$.

# *Introduction*$^2$

- ***Selection problem***
  - Input: A set $A$ of $n$ (distinct) numbers and a number $i$, with $1 \le i \le n$.
  - Output: The element $x \in A$ that is larger than exactly $i$ - 1 other elements of $A$.
- Solution: $O(n \lg n)$
  - Sorting
  - Index the $i$th element
  - There are faster algorithms.
- Average and worst case: $O(n)$.

# *Minimum and Maximum*[1]

- How many comparisons are necessary to determine the minimum?
  - An upper bound of ***n*** **– 1** comparisons.

```
MINIMUM(A)
1  min = A[1]
2  for i = 2 to A.length
3       if min > A[i]
4            min = A[i]
5  return min
```

  - A lower bound of ***n*** **– 1** comparisons.

# *Minimum and Maximum*$^2$

- Simultaneous minimum and maximum
  - In graphics applications
  - At most $3\lfloor n/2 \rfloor$ comparisons are sufficient to find both.
    - Process elements in pairs.
    - First compares each other.
    - Compare the smaller to the current minimum.
    - Compare the larger to the current maximum.
    - At a cost of 3 comparisons for every 2 elements.

# *Minimum and Maximum*[3]

- Setting up initial values for current minimum and maximum:
  - If *n* is odd, set both the minimum and maximum to the value of the first element.
  - If *n* is even, perform 1 comparison on the first two elements to determine the initial values.

# Selection in Expected Linear Time[1]

- **RANDOMIZED-SELECT: Selects the *i*th smallest element of the array *A*[*p* . . *r*].**

```
RANDOMIZED-SELECT(A, p, r, i)
1  if p == r
2       return A[p]
3  q = RANDOMIZED-PARTITION(A, p, r)
4  k = q − p + 1
5  if i == k              // the pivot value is the answer
6       return A[q]
7  elseif i < k
8       return RANDOMIZED-SELECT(A, p, q − 1, i)
9  else return RANDOMIZED-SELECT(A, q + 1, r, i − k)
```

# Selection in Expected Linear Time[2]

- The worst-case running time: $\Theta(n^2)$
- The average-case running time: $O(n)$
  - For each $k$ such that $1 \leq k \leq n$, the subarray $A[p \,..\, q]$ has $k$ elements (all less than or equal to the pivot) with probability $1/n$.
  - Define an indicator random variable

    $X_k = I$ {the subarray $A[p \,..\, q]$ has exactly $\boldsymbol{k}$ elements}

    Assuming that the elements are distinct, we have

    $E[X_k] = 1/n$

# Selection in Expected Linear Time[3]

- For a given call of RANDOMIZED-SELECT
  - $X_k$ has the value 1 for exactly one value of $k$, and it is 0 for all other $k$.
  - When $X_k = 1$, the two subarrays on which we might recurse have sizes $k-1$ and $n-k$. Hence we have

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

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

# Selection in Expected Linear Time[4]

- Taking expected values

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

# Selection in Expected Linear Time[5]

$$\max(k-1, n-k) = \begin{cases} k-1 & \text{if } k > \lceil n/2 \rceil , \\ n-k & \text{if } k \le \lceil n/2 \rceil . \end{cases}$$

- If $n$ is even, each term from $T(\lceil n/2 \rceil)$ up to $T(n-1)$ appears exactly twice in the summation.
- If $n$ is odd, all these terms appear twice and $T(\lfloor n/2 \rfloor)$ appears once.
- Thus, we have

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

# Selection in Expected Linear Time[6]

$$
\begin{aligned}
\mathrm{E}[T(n)] &\le \frac{2}{n}\sum_{k=\lfloor n/2\rfloor}^{n-1} ck + an \\
&= \frac{2c}{n}\left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{\lfloor n/2\rfloor-1} k\right) + an \\
&= \frac{2c}{n}\left(\frac{(n-1)n}{2} - \frac{(\lfloor n/2\rfloor - 1)\lfloor n/2\rfloor}{2}\right) + an \\
&\le \frac{2c}{n}\left(\frac{(n-1)n}{2} - \frac{(n/2-2)(n/2-1)}{2}\right) + an
\end{aligned}
$$

# Selection in Expected Linear Time[7]

$$
\begin{aligned}
&= \frac{2c}{n}\left(\frac{n^2-n}{2} - \frac{n^2/4 - 3n/2 + 2}{2}\right) + an \\
&= \frac{c}{n}\left(\frac{3n^2}{4} + \frac{n}{2} - 2\right) + an \\
&= c\left(\frac{3n}{4} + \frac{1}{2} - \frac{2}{n}\right) + an \\
&\le \frac{3cn}{4} + \frac{c}{2} + an \\
&= cn - \left(\frac{cn}{4} - \frac{c}{2} - an\right) .
\end{aligned}
$$

$E[T(n)] = O(n)$, for $n \geq \frac{c/2}{c/4-a} = \frac{2c}{c-4a}$

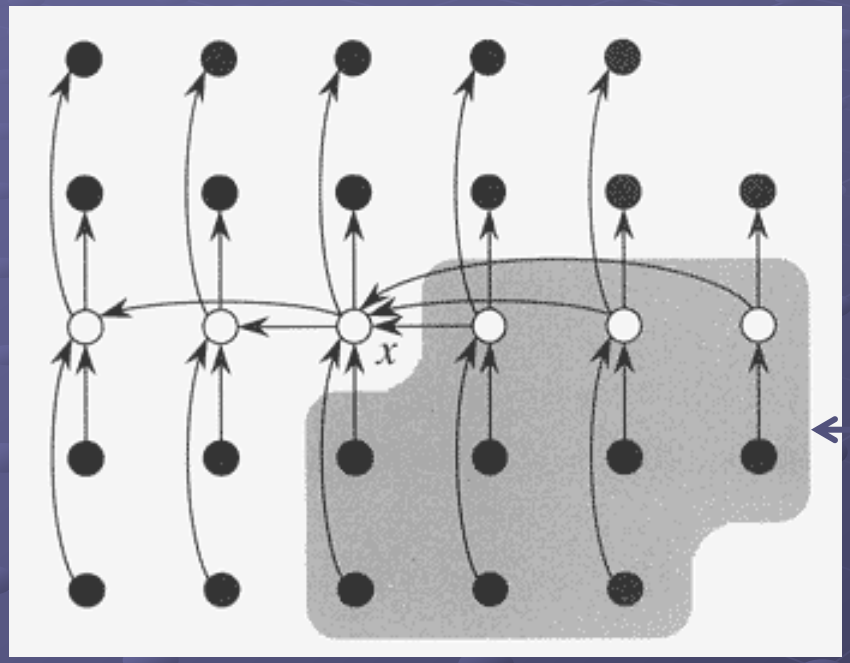
# *Selection in Worst-Case Linear Time*[1]

- **Algorithm**
  - **Divide the $n$ elements of the input array into $\lfloor n/5 \rfloor$ groups of 5 elements each and at most one group made up of the remaining $n$ mod 5 elements.**
  - **Find the median of each of the $\lceil n/5 \rceil$ groups by first insertion sorting the elements of each group and then picking its median.**
  - **Use SELECT recursively to find the median $x$ of the medians found in Step 2.**

# *Selection in Worst-Case Linear Time*[2]

- Partition the input array around the ***median-of-medians*** $x$ using a modified version of partition. Let $k$ be the number of elements on the lower side of the partition, so that $x$ is the $k$th smallest element and there are $n - k$ elements on the high side of the partition.
- If $i = k$, then return $x$. Otherwise, use SELECT recursively to find the ith smallest element on the low side if $i < k$, or ($i - k$)th smallest element on the high side if $i > k$.

# *Selection in Worst-Case Linear Time*




$$3\left(\left\lceil\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil-2\right\rceil\right)\geq\frac{3n}{10}-6$$

**At least** $\frac{3n}{10}-6$ **nodes.**

☞ **Analysis**

- **The number of elements greater than *x* is at least 3*n*/10 – 6.**

# Selection in Worst-Case Linear Time

$$T(n) \le \begin{cases} \Theta(1) & \text{if } n < 140 \\ T(\lceil \frac{n}{5} \rceil) + T(\frac{7n}{10} + 6) + O(n) & \text{if } n \ge 140 \end{cases}$$

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

**Which is at most *cn* if**

$$-cn/10 + 7c + an \le 0 \quad \Rightarrow \quad T(n) = O(n)$$