Selection

- General Selection Problem:
  - select the i-th smallest element from a set of n distinct numbers
  - that element is larger than exactly i - 1 other elements

- The selection problem can be solved in $O(n \log n)$ time
  - Sort the numbers using an $O(n \log n)$-time algorithm, such as merge sort
  - Then return the i-th element in the sorted array

Medians and Order Statistics

**Def:** The i-th order statistic of a set of n elements is the i-th smallest element.

- The minimum of a set of elements:
  - The first order statistic $i = 1$
- The maximum of a set of elements:
  - The n-th order statistic $i = n$
- The median is the “halfway point” of the set
  - $i = \lceil (n+1)/2 \rceil = n/2$ (lower median) and $\lfloor (n+1)/2 \rfloor = n/2 + 1$ (upper median), when $n$ is even

Finding Minimum or Maximum

**Alg:**

```plaintext
MINIMUM(A, n)
min ← A[1]
for i ← 2 to n
    do if min > A[i]
        then min ← A[i]
return min
```

- How many comparisons are needed?
  - $n - 1$: each element, except the minimum, must be compared to a smaller element at least once
  - The same number of comparisons are needed to find the maximum
  - The algorithm is optimal with respect to the number of comparisons performed
Simultaneous Min, Max

- Find min and max independently
  - Use \( n - 1 \) comparisons for each \( \Rightarrow \) total of \( 2n - 2 \)
- At most \( 3n/2 \) comparisons are needed
  - Process elements in pairs
  - Maintain the minimum and maximum of elements seen so far
  - Don’t compare each element to the minimum and maximum separately
  - Compare the elements of a pair to each other
  - Compare the larger element to the maximum so far, and compare the smaller element to the minimum so far
  - This leads to only 3 comparisons for every 2 elements

Analysis of Simultaneous Min, Max

- Setting up initial values:
  - \( n \) is odd: set both \( \text{min} \) and \( \text{max} \) to the first element
  - \( n \) is even: compare the first two elements, assign the smallest one to \( \text{min} \) and the largest one to \( \text{max} \)
- Total number of comparisons:
  - \( n \) is odd: we do \( 3(n-1)/2 \) comparisons
  - \( n \) is even: we do 1 initial comparison + \( 3(n-2)/2 \) more comparisons = \( 3n/2 - 2 \) comparisons

Example: Simultaneous Min, Max

- \( n = 5 \) (odd), array \( A = \{2, 7, 1, 3, 4\} \)
  1. Set \( \text{min} = \text{max} = 2 \)
  2. Compare elements in pairs:
     - \( 1 < 7 \) \( \Rightarrow \) compare 1 with \( \text{min} \) and 7 with \( \text{max} \)
       \( \Rightarrow \) \( \text{min} = 1, \text{max} = 7 \)
     - \( 3 < 4 \) \( \Rightarrow \) compare 3 with \( \text{min} \) and 4 with \( \text{max} \)
       \( \Rightarrow \) \( \text{min} = 1, \text{max} = 7 \)
     \( \Rightarrow \) 3 comparisons
  3. We performed: \( 3(n-1)/2 = 6 \) comparisons

Example: Simultaneous Min, Max

- \( n = 6 \) (even), array \( A = \{2, 5, 3, 7, 1, 4\} \)
  1. Compare 2 with 5: 2 < 5 \( \Rightarrow \) 1 comparison
  2. Set \( \text{min} = 2, \text{max} = 5 \)
  3. Compare elements in pairs:
     - \( 3 < 7 \) \( \Rightarrow \) compare 3 with \( \text{min} \) and 7 with \( \text{max} \)
       \( \Rightarrow \) \( \text{min} = 2, \text{max} = 7 \)
     - \( 1 < 4 \) \( \Rightarrow \) compare 1 with \( \text{min} \) and 4 with \( \text{max} \)
       \( \Rightarrow \) \( \text{min} = 1, \text{max} = 7 \)
     \( \Rightarrow \) 3 comparisons
  4. We performed: \( 3n/2 - 2 = 7 \) comparisons
General Selection Problem

- Select the $i$-th order statistic ($i$-th smallest element) from a set of $n$ distinct numbers.

\[ A[p, q, r] \]

- **Idea:**
  - Partition the input array similarly with the approach used for QuickSort (use RANDOMIZED-PARTITION).
  - Recurs on one side of the partition to look for the $i$-th element depending on where $i$ is with respect to the pivot.

- Selection of the $i$-th smallest element of the array $A$ can be done in $\Theta(n)$ time.

Randomized Select

**Alg:** RANDOMIZED-SELECT($A, p, r, i$)

1. If $p = r$ then return $A[p]$.
2. $q \leftarrow$ RANDOMIZED-PARTITION($A, p, r$).
3. $k \leftarrow q - p + 1$.
4. If $i = k$ then return $A[q]$.
5. Else if $i < k$ then return RANDOMIZED-SELECT($A, p, q-1, i$).
6. Else return RANDOMIZED-SELECT($A, q+1, r, i-k$).

Try: $A =$ \{1, 4, 2, 6, 8, 5\}

Analysis of Running Time

- **Worst case** running time: $\Theta(n^2)$
  - If we always partition around the largest/smallest remaining element.
  - Partition takes $\Theta(n)$ time.
  - $T(n) = O(1)$ (choose the pivot) + $\Theta(n)$ (partition) + $T(n-1)$
    \[ = 1 + n + T(n-1) = \Theta(n^2) \]

A Better Selection Algorithm

- Can perform Selection in $O(n)$ Worst Case

  - **Idea:** guarantee a good split on partitioning.
    - Running time is influenced by how “balanced” are the resulting partitions.
  - Use a modified version of PARTITION
    - Takes as input the element around which to partition.
Selection in $O(n)$ Worst Case

A: $x_1, x_2, x_3, \ldots, x_k$

1. Divide the $n$ elements into groups of 5 $\Rightarrow \lceil n/5 \rceil$ groups
2. Find the median of each of the $\lceil n/5 \rceil$ groups
   - Use insertion sort, then pick the median
3. Use SELECT recursively to find the median $x$ of the $\lceil n/5 \rceil$ medians
4. Partition the input array around $x$, using the modified version of PARTITION
   - There are $k-1$ elements on the low side of the partition and $n-k$ on the high side
5. If $i = k$ then return $x$. Otherwise, use SELECT recursively:
   - Find the $i$-th smallest element on the low side if $i < k$
   - Find the $(i-k)$-th smallest element on the high side if $i > k$

Example

Find the –11th smallest element in array:
$A = \{12, 34, 0, 3, 22, 4, 17, 32, 3, 28, 43, 82, 25, 27, 34, 2, 19, 12, 5, 18, 20, 33, 16, 33, 21, 30, 3, 47\}$

1. Divide the array into groups of 5 elements

Example (cont.)

2. Sort the groups and find their medians

| 0 | 3 | 12 | 34 | 22 | 4 | 17 | 25 | 32 | 3  | 20 | 82 | 33 | 19 | 16 | 12 | 21 | 28 | 34 | 30 | 33 | 47 |
|---|---|----|----|----|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 3 | 3 | 17 | 34 | 28 | 2 | 20 | 5  | 12 | 3  | 20 | 19 | 16 | 30 | 33 | 22 | 19 | 18 | 33 | 33 | 30 |

3. Find the median of the medians

12, 12, 17, 21, 34, 30

Example (cont.)

4. Partition the array around the median of medians (17)

First partition:
$\{12, 0, 3, 4, 3, 2, 12, 5, 16, 3\}$

Pivot:
17 (position of the pivot is $q = 11$)

Second partition:
$\{34, 22, 32, 28, 43, 82, 25, 27, 34, 19, 18, 20, 33, 33, 21, 30, 47\}$

To find the 6-th smallest element we would have to recurse our search in the first partition.