1 Selection

The objective of the selection problem is to find the $k^{th}$ smallest element in an array.

- **Input:** An array $A$ of $n$ distinct comparable elements, and an integer $k$.
- **Output:** An element $a$ from $A$ that is greater than exactly $k - 1$ other elements of $A$.

For example:

- $k = 1$: smallest
- $k = n$: largest
- $k = \lfloor n/2 \rfloor$: median

The output $a$ is called the $k^{th}$ order statistic for $A$.

2 Selection via sorting

One trivial way to solve the selection problem is to sort the entire array:

```
SeleccionViaSorting(A, k)
    Heapsort(A)
    return A[k]
```

This takes $\Theta(n \log n)$ time. Can we do better?

3 QuickSelect

We can adapt QuickSort for the selection problem.

**Key idea:** Only recur on the left side or the right, not both.
4 QuickSelect Analysis

Worst case: \( T(n) = T(n-1) + \Theta(n) = \Theta(n^2) \) (just like QuickSort)

Worst case expected:

\[
E(n) = \Theta(n) + \frac{1}{n} \sum_{q=1}^{n} \max \{ E(q), E(n-q-1) \}
\]

\[
= \Theta(n) + \frac{1}{n} \sum_{q=1}^{\lfloor n/2 \rfloor - 1} E(n-q-1) + \frac{1}{n} \sum_{q=\lfloor n/2 \rfloor}^{n} E(q)
\]

\[
= \Theta(n) + \frac{2}{n} \sum_{q=\lfloor n/2 \rfloor}^{n} E(q)
\]

Show by substitution that \( E(n) = \Theta(n) \). (CLRS 218–219.)

5 Deterministic Linear Time Selection

If we want to \textbf{guarantee} linear time, we can work harder to choose a good pivot, using the “median-of-medians”.

Algorithm:

- \textbf{Divide} the array into groups of 5 elements. (The last group may have less than 5.)
- \textbf{Find the median} of each group. (Sort, then return the third element.)
- Recursively \textbf{find the median} of these \( \lceil n/5 \rceil \) medians.
- \textbf{Partition} the original array using the median-of-medians as the pivot.
- \textbf{Continue} as in QuickSelect. (Stop, or recur on the left or right side as appropriate.)
6 Analysis: Part 1

Key question: How many elements are less than the pivot?

Answer: We know for sure that these elements are less than the pivot:
- The medians of roughly half of the groups.
- Within each of the those groups, two other elements.

So the total number of elements less than the pivot is:

\[
3 \left( \left\lfloor \frac{1}{2} \left\lceil \frac{n}{5} \right\rceil \right\rfloor - 2 \right) \geq \frac{3n}{10} - 6
\]

The number of groups is \( \left\lceil \frac{n}{5} \right\rceil \). The “-2” comes from the fact that there are two groups for which we might have less than three elements less than the pivot:

1. The group containing the pivot itself.
2. The last group, which might not have 5 elements.

7 Analysis: Part 2

Key question: How many elements are greater than the pivot?

Answer: We know for sure that these elements are greater than the pivot:
- The medians of roughly half of the groups.
- Within each of the those groups, two other elements.

So the total number of elements greater than the pivot is:

\[
3 \left( \left\lfloor \frac{1}{2} \left\lceil \frac{n}{5} \right\rceil \right\rfloor - 2 \right) \geq \frac{3n}{10} - 6
\]

8 Analysis: Part 3

The worst case run time is:

\[
T(n) = T(n/5) + T(7n/10 + 6) + \Theta(n)
\]

Show \( T(n) \leq cn \) by substitution:

\[
T(n) \leq \frac{cn}{5} + \frac{7cn}{10} + 6c + an \\
\leq \frac{9cn}{10} + 6c + an \\
\leq cn + \left( -\frac{cn}{10} + 6c + an \right) \\
\leq cn \quad [c \geq 20a, n \geq 120]
\]
Therefore, \( T(n) = O(n) \).

9 Analysis: Part 4

For the last step on the previous slide, we need:

\[-cn/10 + 6c + an \leq 0\]

Solve for \( c \), and choose a sufficiently large \( n \):

\[
c \geq 10a \frac{n}{n-60} \quad [n > 60]
\geq 20a \quad [n > 120]
\]

10 What’s so special about 5?

The first step of the algorithm — “Divide into groups of 5” — comes out of nowhere. Why the mysterious value 5?

**Intuition:** We can show that for this recurrence:

\[ T(n) = T(\alpha n) + T(\beta n) + \Theta(n), \]

if we have

\[ \alpha + \beta < 1, \]

then the solution is

\[ T(n) = \Theta(n). \]

In this case, we have (roughly) \( \alpha + \beta = 1/5 + 7/10 = 9/10 < 1. \)

(We need to perform a **constant fraction** less than \( n \) in recursive calls.)