Notes (Lecture 3)

1 The Selection Problem & Prune-and-Search

1.1 Problem Statement

You are given an array A of n **distinct** elements.

Your goal is to find the k^{th} **smallest element** efficiently.

Example:

If

$$A = [7, 2, 1, 6, 8, 5, 3, 4]$$

and $k = 3$, then the answer is **3**.

1.2 Looking at the Naive Approach

Sort the array and return the k^{th} element.

- Sorting takes $O(n \log n)$.
- But we do not need the full order, only one element.

So we want something faster.

1.3 Idea: Prune and Search

Instead of sorting everything, we:

> Guess, partition, and discard the part that cannot contain the answer.

This is known as the **Prune-and-Search paradigm**. Each step removes many unnecessary elements.

2 General Select Algorithm (Prune & Search)

2.1 Step 1: Pick a Guess g

Choose an element from the array as a **pivot**.

- Deterministic: Median of Medians.
- Randomized: Pick uniformly at random.

2.2 Step 2: Partition

Split the array into:

$$L = \{x \mid x < g\}, \quad R = \{x \mid x > g\}$$

Everything smaller goes left and everything larger goes right.
This takes:

$$O(n)$$

2.3 Step 3: Decide

Let $|L|$ be the size of L.

- If $|L| = k - 1$, then g is the answer.
- If $|L| \geq k$, search in L for the k^{th} smallest.
- If $|L| < k - 1$, search in R for the $(k - |L| - 1)^{th}$ smallest.

Each round throws away part of the array.

2.4 Pseudocode

```
Select(A, k):
    choose pivot g from A
    partition A into L and R

    if |L| == k-1:
        return g
    else if |L| >= k:
        return Select(L, k)
    else:
        return Select(R, k - |L| - 1)
```

3 Dry Run Example

Given:

$$A = [7, 2, 1, 6, 8, 5, 3, 4], \quad k = 3$$

Step 1

Pick $g = 5$.

Step 2

$$L = [2, 1, 3, 4], \quad R = [7, 6, 8]$$

$$|L| = 4$$

Step 3

Since $|L| \geq k$, we recurse:

$$Select([2, 1, 3, 4], 3)$$

Next Round

Pick $g = 2$.

$$L = [1], \quad R = [3, 4]$$
$$|L| = 1$$

Since $|L| < k - 1$, compute:

$$k' = 3 - 1 - 1 = 1$$

Recurse:

$$Select([3, 4], 1)$$

Next Round

Pick $g = 3$.

$$L = []$$
$$|L| = 0 = k - 1$$

Thus the answer is:

$$\boxed{3}$$

Each step shrinks the problem size.

4 Deterministic vs Randomized Selection

Feature	Deterministic	Randomized
Pivot Choice	Median of Medians	Random
Worst Case	$O(n)$	$O(n^2)$
Expected	$O(n)$	$\Theta(n)$

4.1 Deterministic (Median of Medians)

Median of Medians guarantees a good pivot so that a constant fraction is discarded each step.

The standard recurrence is:

$$T(n) \leq T(n/5) + T(7n/10) + cn$$

which solves to:

$$T(n) = O(n).$$

5 Why Randomize?

Even though the worst case is $O(n^2)$, it is extremely unlikely. The probability of picking a bad pivot at every step is roughly $(1/n)^n$, which is effectively zero for large n. Hence, although the worst case exists, the expected running time remains linear in practice.