

# Class-3 Agenda

- Binary Search:
  - [Find the first and last positions of an element in a sorted array](#)
  - [Floor square root](#)
  - Allocate the minimum number of pages

## Hashing

- Characteristics of a problem where hashing has to be used:
  - Search, Insert, Delete =  $O(1)$  avg.
  - Any subset of these operations
  - No other data structure with this much low time complexity
- Hashing in java/C++:
  - Java
    - HashMap
    - HashSet
  - C++
    - Unordered\_map
    - Unordered\_set
- Self Balancing BSTs
  - Java
    - TreeMap
    - TreeSet
  - C++
    - Set
    - map
- Hashing theory:
  - ID from 0 to  $n-1$ , use an array to create the hashmap
  - But what if  $n$  is too large?
  - Take input 'key' and convert it into an integer value
  - hash-function(key)  $\rightarrow$  integer
  - General hash-function:
    - HashFunction(key) = key % prime\_no;
    - Example:  
70,17,12,7,15...  
HF(key) = key%7  
Collision 7 & 70

Problems (Arrays + Hashing):

Subarray with 0 sum

- Brute
- Hashing

Subarrays with equal 1s and 0s

- Hashing

## Find first and last positions of an element in a sorted array

Given a sorted array with possibly duplicate elements, the task is to find indexes of first and last occurrences of an element x in the given array.

```
Input : arr[] = {1, 3, 5, 5, 5, 5, 67, 123, 125}  
        x = 5
```

```
Output : First Occurrence = 2  
          Last Occurrence = 5
```

```
Input : arr[] = {1, 3, 5, 5, 5, 5, 7, 123, 125 }  
        x = 7
```

```
Output : First Occurrence = 6  
          Last Occurrence = 6
```

## Floor square root without using sqrt() function

Given a number **N**, the task is to find the floor square root of the number N without using the built-in square root function. **Floor square root** of a number is the greatest whole number which is less than or equal to its square root.

**Input:**  $N = 25$

**Output:** 5

**Explanation:**

Square root of 25 = 5. Therefore 5 is the greatest whole number less than equal to Square root of 25.

**Input:**  $N = 30$

**Output:** 5

**Explanation:**

Square root of 30 = 5.47

Therefore 5 is the greatest whole number less than equal to Square root of 30 (5.47)

### Allocate minimum number of pages

You are given **N** number of books. Every *i*th book has **A<sub>i</sub>** number of pages.

You have to allocate contiguous books to **M** number of students. There can be many ways or permutations to do so. In each permutation, one of the **M** students will be allocated the maximum number of pages. Out of all these permutations, the task is to find that particular permutation in which the maximum number of pages allocated to a student is **minimum** of those in all the other permutations and print this minimum value.

Each book will be allocated to exactly one student. Each student has to be allocated at least one book.

#### **Input:**

N = 4

A[] = {12,34,67,90}

M = 2

#### **Output:**

113

#### **Explanation:**

Allocation can be done in following ways:

{12} and {34, 67, 90} Maximum Pages = 191

{12, 34} and {67, 90} Maximum Pages = 157

{12, 34, 67} and {90} Maximum Pages = 113

Therefore, the minimum of these cases is 113, which is selected as the output.

**Input:** [10, 20, 10, 30], k = 2

**Output:** 40

## Subarray With Zero Sum

Given an array of positive and negative numbers. Find if there is a **subarray** (of size at-least one) with **0 sum**.

### Example 1:

**Input:**

5

4 2 -3 1 6

**Output:**

Yes

**Explanation:**

2, -3, 1 is the subarray  
with sum 0.

### Example 2:

**Input:**

5

4 2 0 1 6

**Output:**

Yes

**Explanation:**

0 is one of the element  
in the array so there exist a  
subarray with sum 0.

## Subarrays With Equal 1s and 0s

Given an array containing 0s and 1s. Find the number of subarrays having equal number of 0s and 1s.

### Example 1:

**Input:**

`n = 7`

`A[] = {1,0,0,1,0,1,1}`

**Output:** 8

**Explanation:** The index range for the 8 sub-arrays are: (0, 1), (2, 3), (0, 3), (3, 4), (4, 5), (2, 5), (0, 5), (1, 6)

### Example 2:

**Input:**

`n = 5`

`A[] = {1,1,1,1,0}`

**Output:** 1

**Explanation:** The index range for the subarray is (3,4).