AGENDA:

1. Hashing basics
2. Practice problems:
    a. Subarray with 0 sum.
        i. Brute-force
        ii. Hashing
    b. Subarrays with equal number of 1s and 0s.
        i. Hashing
    c. Longest substring with distinct characters.
        i. 2-pointer + Hashing
        ii. Hashing
    d. Group Anagrams together.
        i. Custom-Hashing

TODO AFTER THE SESSION:

Please cover the following articles after this session. Contains some basics revolving around hashing.

- https://www.geeksforgeeks.org/hashing-set-1-introduction/
- https://www.geeksforgeeks.org/index-mapping-or-trivial-hashing-with-negatives-allowed/
- https://www.geeksforgeeks.org/hashing-set-2-separate-chaining/
- https://www.geeksforgeeks.org/hashing-set-3-open-addressing/
- https://www.geeksforgeeks.org/double-hashing/

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) → integer
    - General hash-function:
        - HashFunction(key) = key % prime_no;
        - Example:
          70,17,12,7,15…
          HF(key) = key%7
          Collision 7 & 70

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).
```

Longest substring with distinct characters

Given a string **s**, find the length of the longest substring with all distinct characters.

Examples:

Input: s = "geeksforgeeks"
Output: 7
Explanation: "eksforg" is the longest substring with all distinct characters.

Input: s = "aaa"
Output: 1
Explanation: "a" is the longest substring with all distinct characters.

Input: s = "abcdefabcbb"
Output: 6
Explanation: The longest substring with all distinct characters is "abcdef", which has a length of 6.

Constraints:
$1 <= s.size() <= 3*10^4$
All the characters are in lowercase.

Group Anagrams together

Given an array of strings, return all groups of strings that are anagrams. The strings in each group must be arranged in the order of their appearance in the original array. Refer to the sample case for clarification.

Examples:

Input: arr[] = ["act", "god", "cat", "dog", "tac"]
Output: [["act", "cat", "tac"], ["god", "dog"]]
Explanation: There are 2 groups of anagrams "god", "dog" make group 1. "act", "cat", "tac" make group 2.

Input: arr[] = ["no", "on", "is"]
Output: [["is"], ["no", "on"]]
Explanation: There are 2 groups of anagrams "is" makes group 1. "no", "on" make group 2.

Input: arr[] = ["listen", "silent", "enlist", "abc", "cab", "bac", "rat", "tar", "art"]
Output: [["abc", "cab", "bac"], ["listen", "silent", "enlist"], ["rat", "tar", "art"]]
Explanation:
Group 1: "abc", "bac", and "cab" are anagrams.
Group 2: "listen", "silent", and "enlist" are anagrams.
Group 3: "rat", "tar", and "art" are anagrams.