Session 1

Topic: Array & Matrix

Characteristics of Array

- An array is a collection of items of the same data type
- Memory Locations need to be contiguous

400	401	402	403	404	405	406
a	b	c	d	e	f	g

Characteristics of Array

- Elements can be randomly accessed
- Searching in an array is O(N) time complexity
- Insertion in an array at the end is O(1) time complexity
- Insertion in an array in between any index is O(N). How?
- Deletion in an array via searching or via an index is O(N)

Fixed Size Array vs Dynamic Size Array

- Java

```
int [] arr = new int[100];

int [] arr = new int[n];

int arr[] = {1,2,3,4}
```

Arrays always created on heap segment of memory

- C / C++

```
int arr[100];  //stack segment

int arr[n];  // stack segment

int * arr = new int[n]; // heap segment

int arr[] = {1, 2, 3, 4}; // stack segment
```

Dynamic Sized Array

- C++ : vector
- Java : ArrayList
- Python : list

how a dynamic sized array works - discussed in previous class

Recommended to use dynamic sized arrays in language of your choice -> flexibility, **don't reinvent the wheel (for inserting, deleting, sorting, reversing)**

Resources if not familiar with Dynamic Sized Arrays

- C++ : vector -> https://www.javatpoint.com/cpp-vector
- Java : ArrayList -> https://www.javatpoint.com/java-arraylist
- Python : list -> https://www.w3schools.com/python/python_lists.asp

Multi-Dimensional Array (Java)

Array Declaration

```
int[][] arr_twoD = new int[5][10];
arr_twoD[0][0] = 1;

System.out.println("arr[0][0] = " + arr_twoD[0][0]);

int[][][] arr_threeD = new int[5][10][8];
arr_threeD[0][0][0] = 1;

System.out.println("arr[0][0][0] = " + arr_threeD[0][0][0]);

int[][] arr = { { 1, 2 }, { 3, 4 } };
```

Argument passing

```java
public static void main(String[] args) {
    int[][] arr = { { 1, 2 }, { 3, 4 }, { 7, 8 } };
    acceptIt(arr); // pass it to the method
}

public static void acceptIt(int[][] a) {
    System.out.println("Elements are :");

    for (int i = 0; i < a.length; i++) {
        for (int j = 0; j < a[i].length; j++) {
            System.out.print(a[i][j] + "\t");
        }
        System.out.println("");
    }
}
```

Multi-Dimensional Array (C++)

Array Declaration

```
int arr_twoD[2][3];
arr_twoD[0][0] = 1;

cout<<"arr[0][0] = "<<arr_twoD[0][0];

int arr_threeD[5][10][8];
arr_threeD[0][0][0] = 1;

cout<<"arr[0][0][0] = "<<arr_threeD[0][0][0];

vector<vector<int>> arr = {{0,1}, {2,3}, {4,5}};
cout<<"arr[0][0] = "<<arr[0][0];
```

Argument passing

```
vector<vector<int>> arr = { { 1, 2 }, { 3, 4 }, { 7, 8 } };
acceptIt(arr); // pass it to the method

void acceptIt(vector<vector<int>> &a) {
    cout<<"Elements are :"<<endl;

    for (int i = 0; i < a.size(); i++) {
        for (int j = 0; j < a[i].size(); j++) {
            cout<<a[i][j]<<" ";
        }
        cout<<endl;
    }
}
```

Common Interview Concepts related to Array through Problems

Problem 1: Array Rotation

Array

1	2	3	4	5	6	7

Left Rotate the Array by 2

3	4	5	6	7	1	2

Right Rotate the Array by 3

5	6	7	1	2	3	4

Array Rotation Implementation Discussion

Sliding Window Technique

Problem 2: Given an array of size n, lets calculate the maximum sum of k consecutive elements in the array.

Where all can the sliding window technique be easily applied?

- The operation to be computed is dependent upon the clear cut addition & removal kind of scenario.

- There are scenarios where it is not so easy to apply sliding window. Eg: Max in array

Google's Question on Sliding Window with a twist

Problem 3: Given an array of size n, lets calculate the maximum ***product*** of k consecutive elements in the array.

Problem 4: Prefix Sum of an Array

Given an array arr[] of size N, the task is to generate the *prefix sum array* of the given array.

Prefix Sum Array: The prefix sum array of any array, arr[] is defined as an array of same size say, prefixSum[] such that the value at any index **i** in prefixSum[] is sum of all elements from indexes **0 to i** in arr[].

Problem 5

Problem Statement

- Given an array A of N integers, find how many pairs i, j (1 <= i < j <= N) exist such that A[i] is even and A[j] is odd.

- **Overall constraints:** A[i] <= 10^5
- **Constraint 1:** N <= 100
- **Constraint 2:** N <= 10^5

Examples

Input

4

1 2 1 3

Output

2

Explanation: The pairs are the indices (2,3) and (2,4).

Input

5

5 4 1 2 3

Output

3

Explanation: The pairs are indices (2,3), (2,5) and (4,5).

Solution

[Constraint 1]

Since N is only 100, we can check all pairs of integers. We can run a loop for i=1 to N. If N is even, we run a nested loop from j=i+1 to N. Now we add 1 to the answer for each odd A[j].

Solution

[Constraint 2]

Notice that in our initial solution, we are wasting time by running a loop from i+1 to N each time. Instead, we can precompute it once and decrement as the outer loop moves forward. To do this we can maintain a variable to store even numbers so far. Everytime an odd number is encountered, we add the even numbers so far to the answer.

Complexity Analysis

Time Complexity:
Constraint 1**:** $O(N^2)$
Explanation: We're running 2 nested loops. In the worst case, the first half of the numbers in the array are odd, so we do N/2 + N/2-1 + N/2-2 … 1. This gives us a complexity of $O(N^2)$.

Constraint 2: O(N)
Explanation: We're running only a single loop. Updating our variable is done in O(1).

Space Complexity: O(1) for both.

Code

Code for constraint 1

Code for constraint 2

Problem 6

Problem Statement

- **Given an array of positive integers, determine the largest value of perimeter of a non-degenerate triangle whose side lengths are any three values from the given array. Each value from the array can be used at most once. If no non-degenerate triangles exist, return -1.**

Examples

- arr = [1 1 1 3 3]

  Ans = 7, form a triangle [1 3 3]

- arr = [1 2 3]

  Ans = -1

- arr = [1 2 3 1 1 5]

  Ans = 3

Solution

- The solution follows greedy algorithm.
- Sort the array in descending order
- Keep comparing in groups of three whether arr[i+1] + arr[i+2] > arr[i]
- If true, then sum of the above three is the ans
- Else, keep moving one iterator ahead.
- If no group of three exist that satisfies the above condition, return -1

Complexity Analysis

Time Complexity: O(NlogN) -> order of the best sorting algorithm

Code

https://ideone.com/cveoiS

Problem 7 - Google Intw Question

Problem Statement

- Question: Given a word, return the first unique character.

- The question looks quite simple at beginning, but the trick is that it is very open ended. What exactly is meant by unique? What is the word length you are dealing with? What sort of letters will be there in the word? small a to z, A to Z , a mix of both, will numbers be also there, can the word also contain special characters? All this things need to be cleared before jumping straight to the solution. (Very Important to clarify in Interview Setting)
- There are even more cases and questions than above that needs to figured out. Refer to examples in the later slide.

- Defn of unique: A character which occurs only once. If there are multiple such characters that occur only once, the one with the least index among them is the first unique character that needs to returned.
- for starting, keep the word of a length that fits into memory like of the length 10^5
- for starting, keep the word to only consist of small a-z

Examples

- "abccdef" , Ans: 'a'
- "", Ans: Interviewee has to take care to handle this situation, or ask the interviewer if the word length can be 0
- "dcasghgsacd", Ans: 'h'
- "abccba", Ans: Now, what to do in this case where no there is no such character that occurs only once. The interviewee has to clarify this with the interviewer. The interviewer can ask him to return -1 in such cases or assure that you can ignore such cases and continue coding.

Solution

- Consider the defn of unique as given in the last slide and a word length that fits in the memory. Also, the word consists of only small a to z characters.
- Just store a simple hashmap of 'a' to 'z' characters in the form of a array or data structure of choice.
- Basically, count['a'], count['b'], ….. and so on after running through the word
- Run through the word again, whichever character's count is exactly 1, return that index.
- Alternatively, you can store the count as well as the least index in the hashmap which means that you may not have to run through the word again giving you an even cleaner solution.
- Exceptions like empty string or no unique character needs to be handled.

Complexity Analysis

Time: O(N) -> we are basically running through the length of the word

Space: O(constant) -> the extra space that is required is a constant that is a 26 length hashmap or array

Code

https://ideone.com/AiOgNY

Few other problems to try ...

1. Merging Two Sorted Arrays

2. Range Sum Queries using Prefix Sum (Assignment)

Interview Problems & Patterns on Matrix

Problem 1: How to print a matrix in snake pattern (the first row should be printed from left to right, second row from right to left, and so on)

```
for (int i = 0; i < n; i++) {
    if (i % 2 == 0) {
        // even row
        for (int j = 0; j < m; j++) {
            cout << arr[i][j] << " ";
        }
    } else {
        // odd row
        for (int j = m - 1; j >= 0; j--) {
            cout << arr[i][j] << " ";
        }
    }
    cout << endl;
}
```

Problem 2: Matrix boundary traversal problem (first row left to right, last column top to bottom, last row right to left and first column bottom to top)

```
for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {
        if (i == 0 || j == 0 || i == n - 1 || j == m - 1) {
            cout << arr[i][j] << " ";
        } else {
            cout << "  ";
        }
    }
    cout << endl;
}
```

Problem 3: Find transpose of matrix

```
for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {
        transpose[j][i] = arr[i][j];
    }
}
```

In-Place Transpose

```
for (int i = 0; i < n; i++) {
    for (int j = i + 1; j < m; j++) {
        swap(arr[i][j], arr[j][i]);
    }
}
```

Problem 4: Search in a row-wise & column wise sorted matrix

```
// set indexes for top right element
int i = 0, j = m - 1;

while (i < n && j >= 0) {
    if (mat[i][j] == x) {
        cout << "Found at " << i << ", " << j;
        return;
    }
    if (mat[i][j] > x) {
        j--;
    } else {
        // Check if mat[i][j] < x
        i++;
    }
}
```

Problem 5: Rotate matrix anti-clockwise by 90 degree

```
for (int i = 0; i < n / 2; i++) {
    // Consider elements in group
    // of 4 in current square
    for (int j = i; j < n - i - 1; j++) {
        // Store current cell in
        // temp variable
        int temp = mat[i][j];

        // Move values from right to top
        mat[i][j] = mat[j][n - 1 - i];

        // Move values from bottom to right
        mat[j][n - 1 - i] = mat[n - 1 - i][n - 1 - j];

        // Move values from left to bottom
        mat[n - 1 - i][n - 1 - j] = mat[n - 1 - j][i];

        // Assign temp to left
        mat[n - 1 - j][i] = temp;
    }
}
```

Assignment Problem HW: Spiral Traversal of a matrix

Input:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Output: 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

Concept: Prefix Sum of a Matrix (1 Assignment problem based on this concept)

Input:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Output = ?

Array & Matrix Assignment released today

8 problems **mostly** on the concepts taught around the class

Do try it out.