Java DSA Notes - Arrays Part 1

1. Linear Search

```
public static int LinearSearch(int[] marks, int key) {
    for (int i = 0; i < marks.length; i++) {
        if (key == marks[i]) {
            return i;
        }
    }
    return -1;
}
```

Linear search checks each element one by one. Time complexity: O(n).

2. Find Max in Array

```
public static int Max(int[] marks) {
    int max = marks[0];
    for (int index = 0; index < marks.length; index++) {
        if (marks[index] >= max) {
            max = marks[index];
        }
    }
    return max;
}
```

Traverse array, compare and update max. Time complexity: O(n).

3. Binary Search (Sorted Arrays)

```
public static int BinarySearch(int[] marks, int key) {
    int start = 0, end = marks.length - 1;
    while (start <= end) {
        int mid = (start + end) / 2;
        if (marks[mid] == key) return mid;
        else if (marks[mid] < key) start = mid + 1;
        else end = mid - 1;
    }
    return -1;
}
```

Efficient searching in sorted array. Time complexity: O(log n).

4. Reverse an Array

```
public static void ReverseArr(int[] marks) {
    int first = 0, last = marks.length - 1;
    while (first < last) {
        int temp = marks[last];
        marks[last] = marks[first];
        marks[first] = temp;
        first++;
        last--;
    }
}
```

Swap start and end moving inward. Time complexity: O(n/2).

5. Print All Pairs

```java
public static void PairsArr(int[] marks) {
    for (int i = 0; i < marks.length; i++) {
        for (int j = i + 1; j < marks.length; j++) {
            System.out.print("(" + marks[i] + "," + marks[j] + ")");
        }
        System.out.println();
    }
}
```

Print all combinations of pairs. Total pairs = n(n-1)/2.

6. Print All Subarrays

```java
public static void SubArr(int[] marks) {
    int ts = 0;
    for (int i = 0; i < marks.length; i++) {
        for (int j = i; j < marks.length; j++) {
            for (int k = i; k <= j; k++) {
                System.out.print(marks[k] + " ");
            }
            ts++;
            System.out.println();
        }
    }
    System.out.println("Total Sub arrays : " + ts);
}
```

Subarrays are contiguous parts. Total = n(n+1)/2. Time complexity: O(n^3).