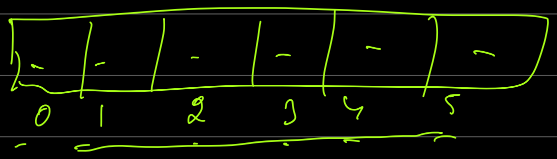
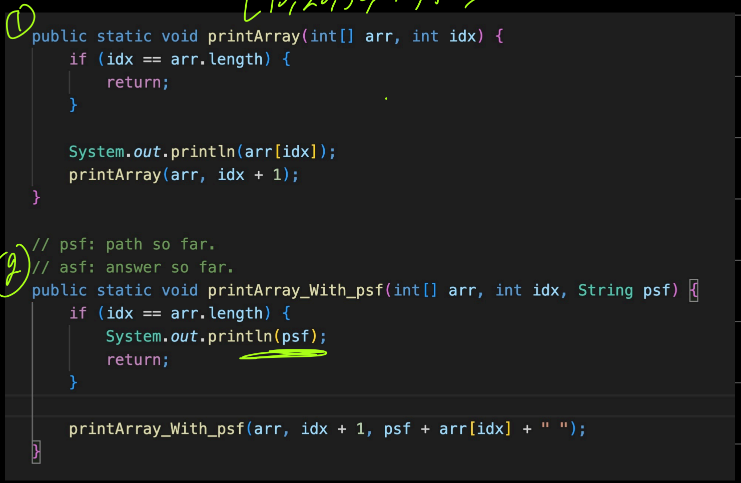
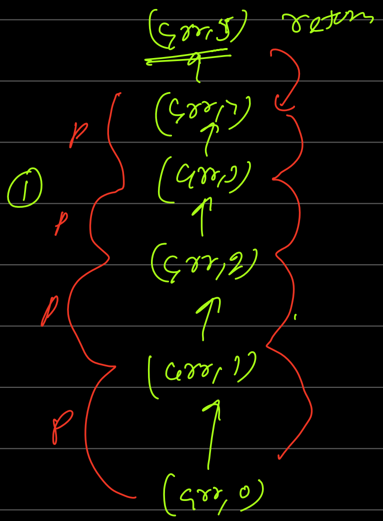


0	arr(0)	—
1	arr[1]	arr(1)
2	arr (2)	arr (2)
3	arr(3)	arr(3)
4	arr(4)	arr(4)
5	arr[5]	arr[5]

[10, 20, 30, 40, 50]



```
① public static void printArray(int[] arr, int idx) {
      if (idx == arr.length) {
          return;
      }

      System.out.println(arr[idx]);
      printArray(arr, idx + 1);
  }

  // psf: path so far.
② // asf: answer so far.
  public static void printArray_With_psf(int[] arr, int idx, String psf) {
      if (idx == arr.length) {
          System.out.println(psf);
          return;
      }

      printArray_With_psf(arr, idx + 1, psf + arr[idx] + " ");
  }
```



②

$$\begin{tikzcd}
(arr, 4, 10\_20\_30\_40) \arrow[r] & (arr, 5, 10\_20\_30\_40\_50) \\
(arr, 3, 10\_20\_30) \arrow[u] & \\
(arr, 2, 10\_20) \arrow[u] & \\
(arr, 1, 10) \arrow[u] & \\
(arr, 0, "\ ") \arrow[u] &
\end{tikzcd}$$

get maximum

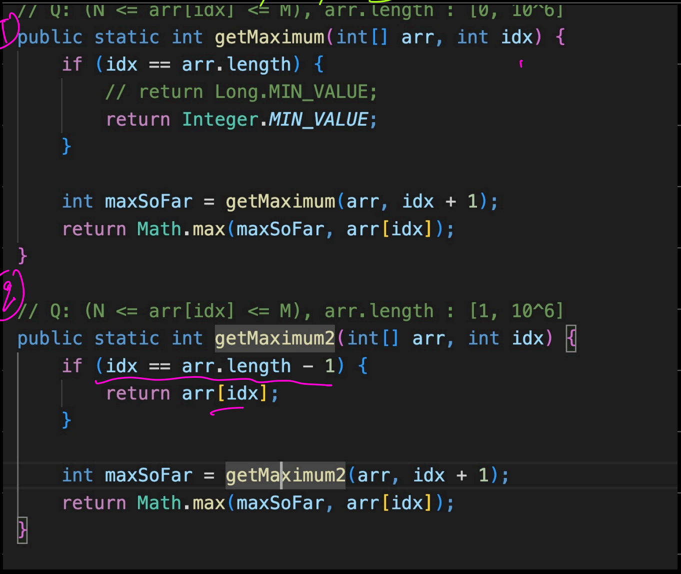
0	1	2	3	4	5	6	7
10	20	30	50	5	70	80	2

80

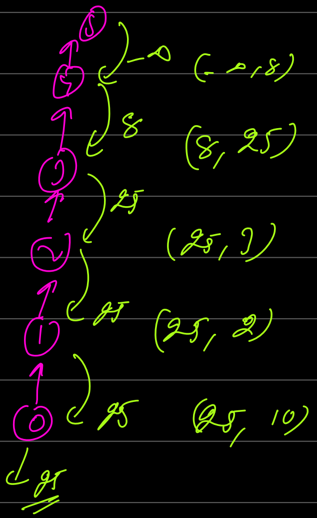
int = -2^{32}, $(2^{32}-1)$ [Integer.MIN_VALUE, Integer.MAX_VALUE]

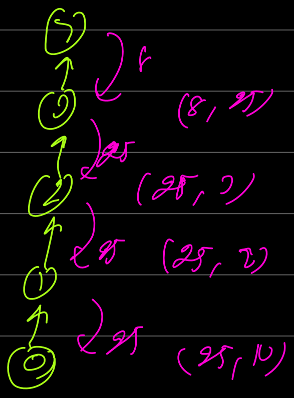
long = -2^{64}, $(2^{63}-1)$ [Long.MIN_VALUE, Long.MAX_VALUE]

0	1	2	3	4
10	2	3	25	8



```java
// Q: (N <= arr[idx] <= M), arr.length : [0, 10^6]
①public static int getMaximum(int[] arr, int idx) {
    if (idx == arr.length) {
        // return Long.MIN_VALUE;
        return Integer.MIN_VALUE;
    }

    int maxSoFar = getMaximum(arr, idx + 1);
    return Math.max(maxSoFar, arr[idx]);
}

②// Q: (N <= arr[idx] <= M), arr.length : [1, 10^6]
public static int getMaximum2(int[] arr, int idx) {
    if (idx == arr.length - 1) {
        return arr[idx];
    }

    int maxSoFar = getMaximum2(arr, idx + 1);
    return Math.max(maxSoFar, arr[idx]);
}
```



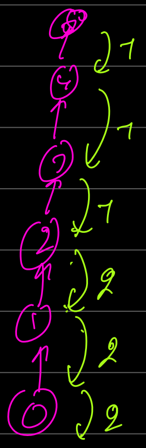


lgn

find index

[8, 9, 10, 20, 30] , 10

0 1 2 3 4



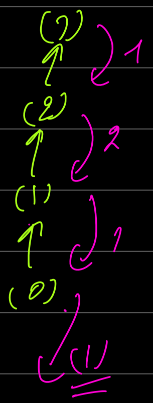
First index

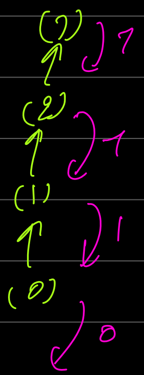
20

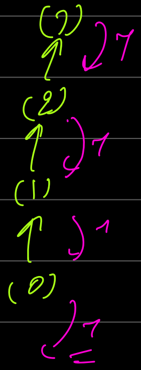
0 1 2

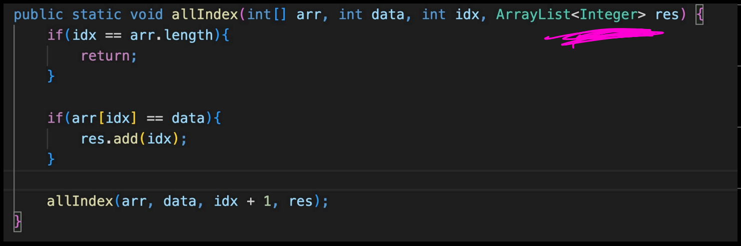
[10, 20, 20] , [20, 20, 10] , [10, 30, 40]

0 1 2 0 1 2

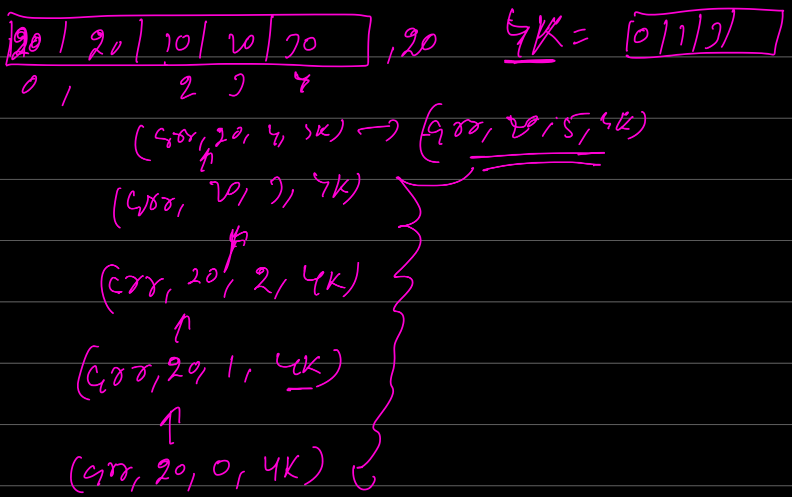


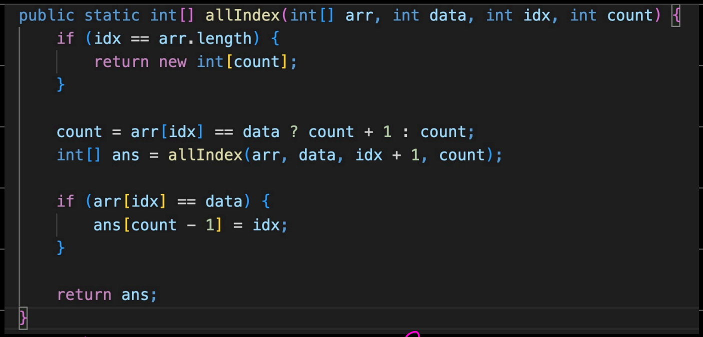




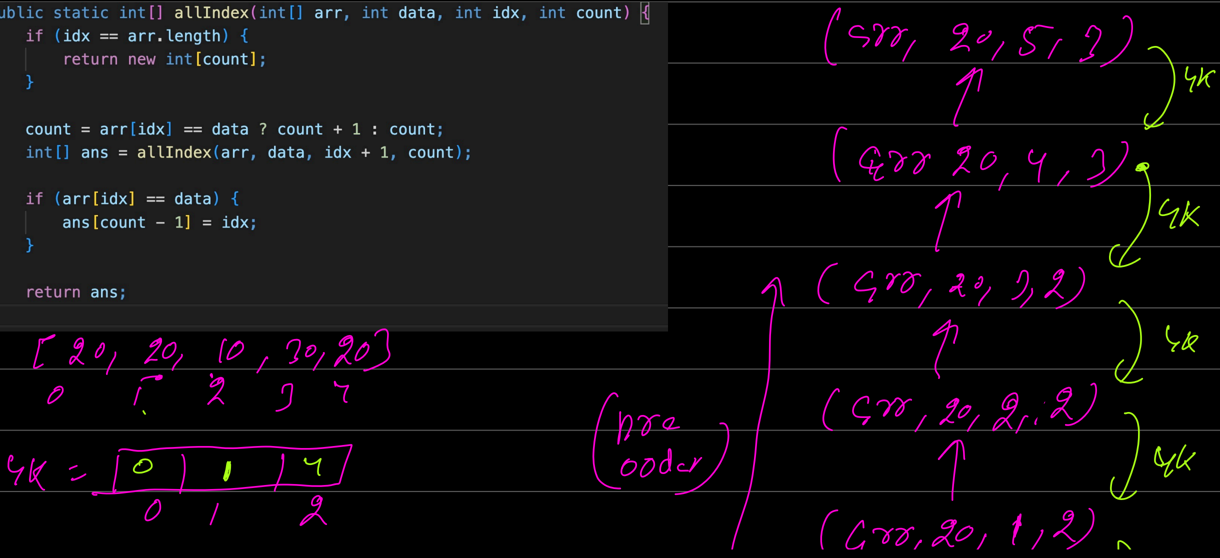


```java
public static void allIndex(int[] arr, int data, int idx, ArrayList<Integer> res) {
    if(idx == arr.length){
        return;
    }

    if(arr[idx] == data){
        res.add(idx);
    }

    allIndex(arr, data, idx + 1, res);
}
```





```java
public static int[] allIndex(int[] arr, int data, int idx, int count) {
    if (idx == arr.length) {
        return new int[count];
    }

    count = arr[idx] == data ? count + 1 : count;
    int[] ans = allIndex(arr, data, idx + 1, count);

    if (arr[idx] == data) {
        ans[count - 1] = idx;
    }

    return ans;
}
```



(400, 20, 0 1) [illegible]

inverse

[3, 1, 0, 2] → [2 | 1 | 3 | 0]

```
 0  1  2  3        0  1  2  3
```

(*) Subsequence of String

abc

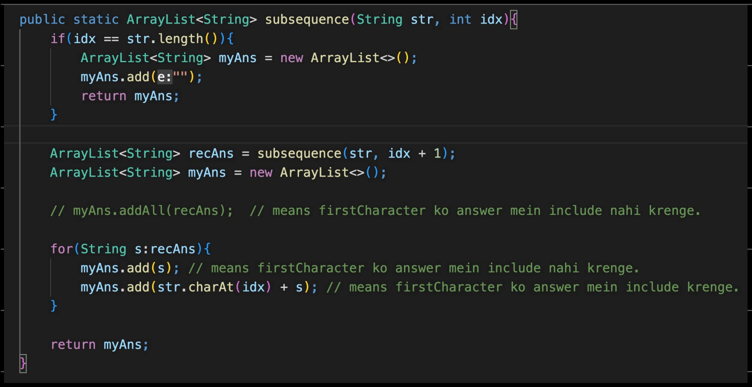
```
    -
a   a
b   b
    ab
c   c
    ac
    bc
    abc
```

abcd

-	d
a	ad
b	bd
ab	abd
c	cd
ac	acd
bc	bcd
abc	abcd.

abc

[-, b, c, bc]

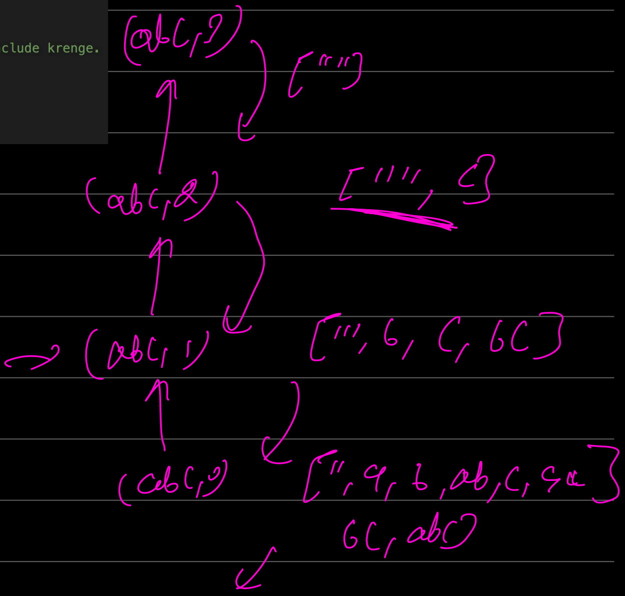


```
public static ArrayList<String> subsequence(String str, int idx){
    if(idx == str.length()){
        ArrayList<String> myAns = new ArrayList<>();
        myAns.add(e:"");
        return myAns;
    }

    ArrayList<String> recAns = subsequence(str, idx + 1);
    ArrayList<String> myAns = new ArrayList<>();

    // myAns.addAll(recAns);  // means firstCharacter ko answer mein include nahi krenge.

    for(String s:recAns){
        myAns.add(s); // means firstCharacter ko answer mein include nahi krenge.
        myAns.add(str.charAt(idx) + s); // means firstCharacter ko answer mein include krenge.
    }

    return myAns;
}
```

['', b, c, bc]

↑

5



(abc, 3)
[""]
(abc, 2)
['', c]
(abc, 1)
['', b, c, bc]
(abc, 0)
['', a, b, ab, c, ac, bc, abc]