① Primes in Range - 1

Find all prime numbers between A and B.

Eg - A = 1, B = 9

ans[] → [2, 3, 5, 7]

idea-1. For every number from A and B, check whether it is prime or not.

T.C → $O(N\sqrt{N})$, S.C → $O(1)$

idea-2. Using sieve.

```
boolean prime[] = new boolean[B+1];
∀i, prime[i] = true;
prime[1] = false;
for( i = 2; i*i <= B; i++){
    if( prime[i] == true){
        for( j = i*i; j <= B; j += i){
            prime[j] = false;
        }
    }
}
```

```
List<Integer> ans;
for( i = A ; i <= B ; i++){
    if ( prime[i] == true)
    {
        ans.insert(i);
    }
}
return ans;
```

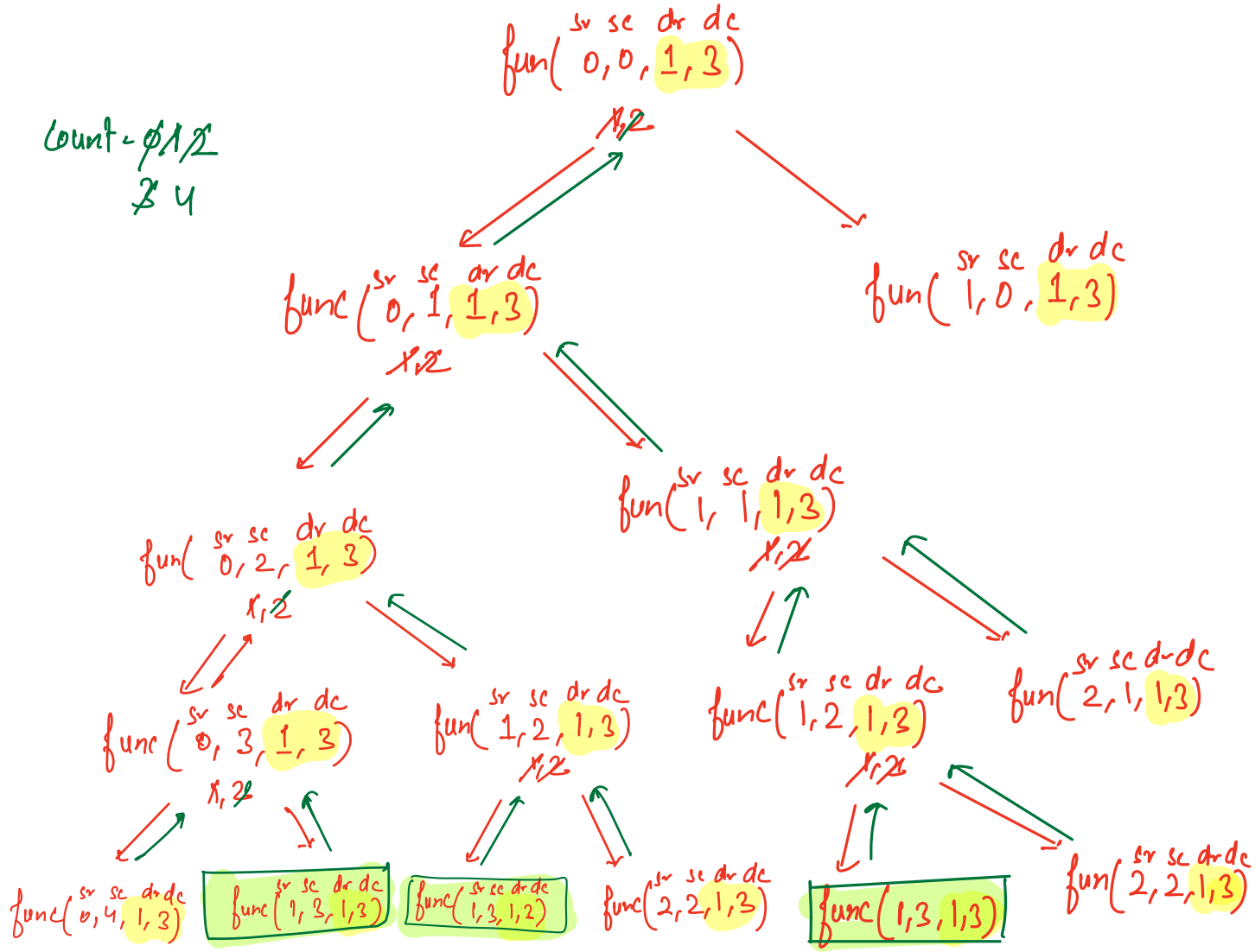
T.C → O(B log (log B))
S.C → O(B)

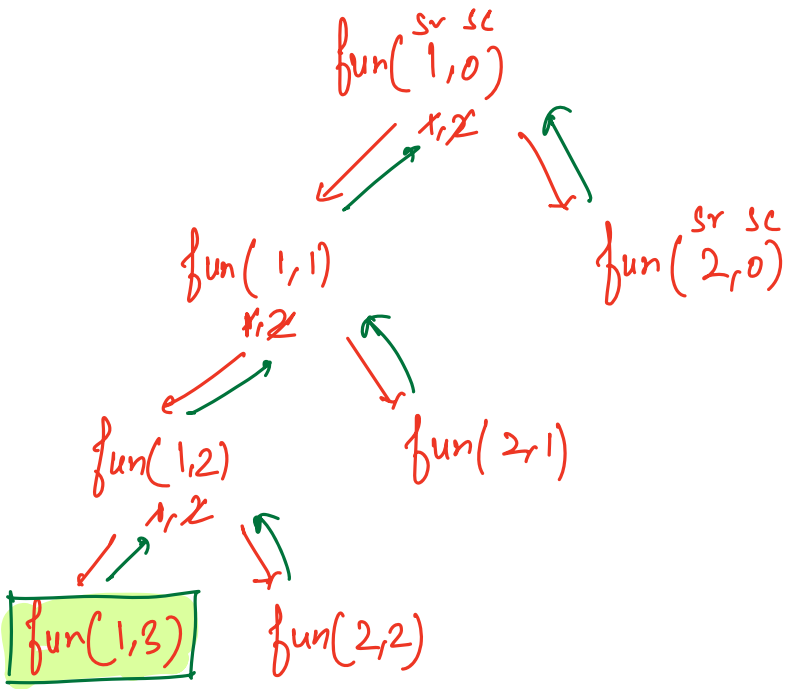
2. Output of the following recursive code →

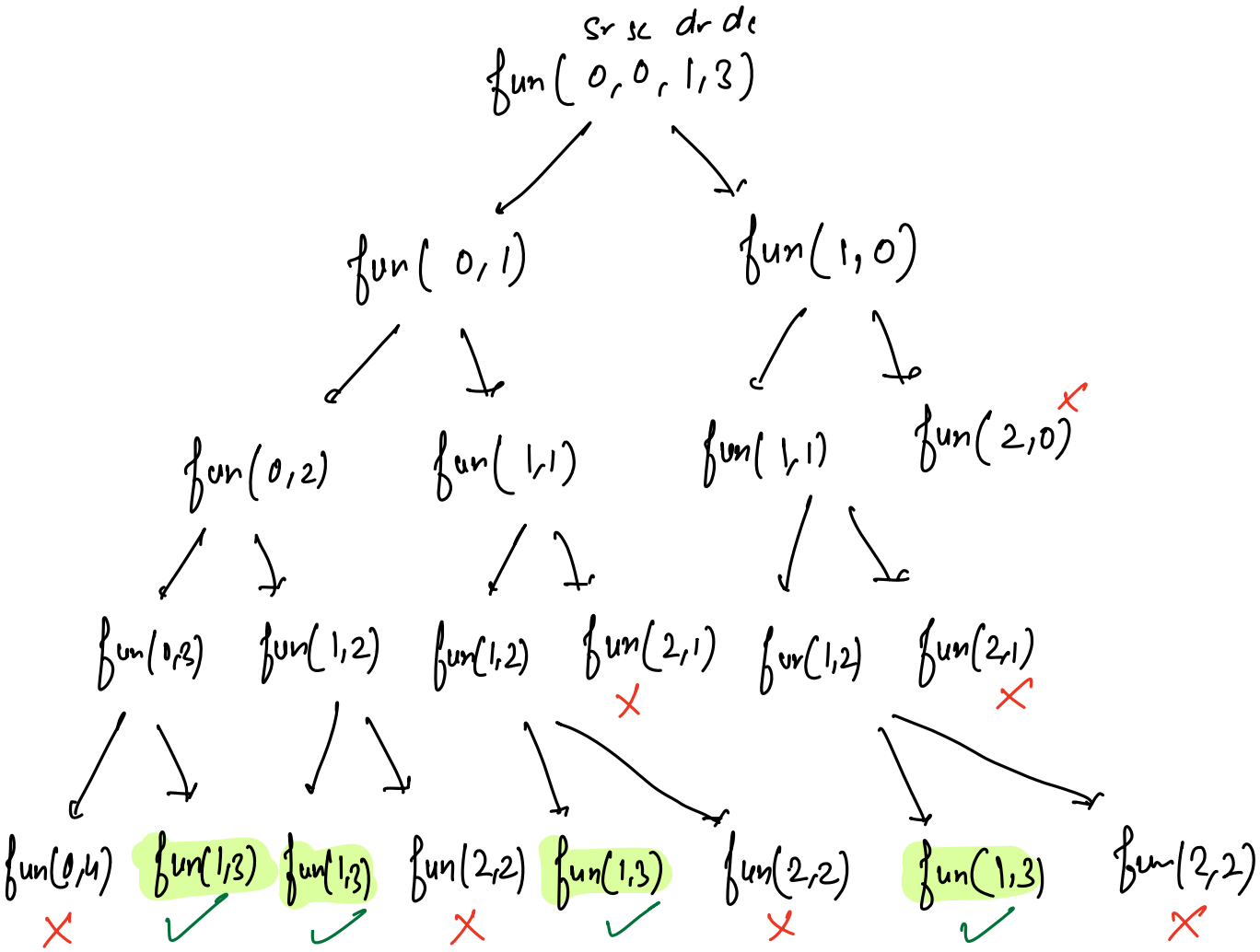
```
void main (){
    func( 0,0, 1, 3);
    print( count)
}

int count = 0;
void func( int sr, int sc, int dr, dc){
    if (sr > dr || sc > dc){
        return ;
    }

    if (sr == dr && sc == dc){
        count++;
        return;
    }

①   func( sr, sc+1, dr, dc);
②   func( sr+1, sc, dr, dc);
}
```

(0) (0) (1) (3)

count = 0 1 2
3 4


fun(sr sc dr dc
0, 0, 1, 3
fun(1, 0, 1, 3
func(0, 1, 1, 3
fun(1, 1, 1, 3
fun(0, 2, 1, 3
func(0, 3, 1, 3
fun(1, 2, 1, 3
func(1, 2, 1, 3
fun(2, 1, 1, 3
func(0, 4, 1, 3
func(1, 3, 1, 3
func(1, 3, 1, 2
fun(2, 2, 1, 3
func(1, 3, 1, 3
fun(2, 2, 1, 3


fun(sr sc
1, 0
fun(1, 1
fun(2, 0
fun(1, 2
fun(2, 1
fun(1, 3
fun(2, 2


sr sc dr dc
fun(0, 0, 1, 3)
fun(0, 1)
fun(1, 0)
fun(0,2)
fun(1,1)
fun(1,1)
fun(2,0)
fun(0,3)
fun(1,2)
fun(1,2)
fun(2,1)
fun(1,2)
fun(2,1)
fun(0,4)
fun(1,3)
fun(1,3)
fun(2,2)
fun(1,3)
fun(2,2)
fun(1,3)
fun(2,2)

3. **Maximum Unsorted Subarray** →

Given arr[N] of non-negative integers. Find the shortest subarray such that if we sort that subarray, whole array get sorted.

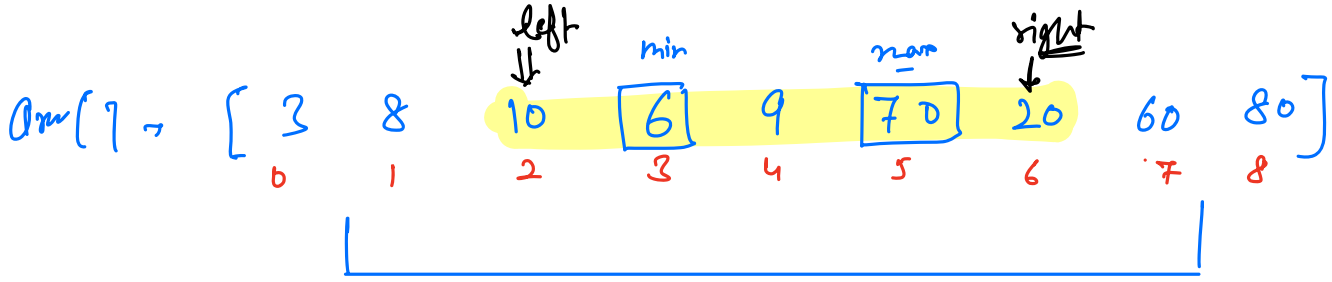
If A is already sorted, return -1.

arr[] → [3 8 10 6 9 30 20 60 80]
0 1 2 3 4 5 6 7 8

[3 6 8 9 10 20 30 60 80]

idea.1 → Copy all the elements of array into another array. Sort another array & compare every element of original and copied array.

T.C → O(NlogN)
S.C → O(N)



① Find left and right problamatic points.

```
left = -1 , right = -1
for( i = 1 ; i < N; i++){
    if ( arr[i] < arr[i-1]){
        left = i-1 , break
    }
}

for( i = N-2; i ≥ 0 ; i--){
    if (arr[i+1] < arr[i]){
        right = i+1 , break
    }
}

if ( left == -1 && right == -1) {
    int[] ans = new int[1] ; ans[0] = -1 , return ans;
}
```

// array is already sorted.

- Find min & max element in problamatic region.

```
min → ∞ , max → -∞
for( i = left; i ≤ right ; i++){
    min = Min( min, arr[i]);
    max = Max( max, arr[i]);
}
```

→ Find correct indices for min & max

```
int[] res = new int[2];
for( i = 0; i < N; i++){
    if (arr[i] > min){
        res[0] = i;
        break;
    }
}

for( i = N-1; i >= 0; i--){
    if (arr[i] < max){
        res[1] = i;
        break;
    }
}

return res[];
```

T.C → O(N)
S.C → O(1)