→ 2 min

[a ↔ b]

Sum All Numbers in a Range

We'll pass you an array of two numbers. Return the sum of those two numbers plus the sum of all the numbers between them. The lowest number will not always come first.

For example, `sumAll([4,1])` should return `10` because sum of all the numbers between 1 and 4 (both inclusive) is `10`.

Run the Tests (Ctrl + Enter)

Reset All Code

Get Help

Tests

- `sumAll([1, 4])` should return a number.
- `sumAll([1, 4])` should return 10.
- `sumAll([4, 1])` should return 10.
- `sumAll([5, 10])` should return 45.
- `sumAll([10, 5])` should return 45.

1+2+3+4 = 10

[10,5] → 10+9+8+7+6+5 = 45

☑ Input and output

☑ Write steps in English to solve

Step1 – ☐ compare a and b

☐ if a is smaller
↳ iterate a–b and sum

☐ else
↳ iterate b–a and sum

☐ return Sum

☑ writing the code (n) X

☐ understand T C → O(abs(a-b))

☐ Can we improve TC.

```
1  function sumAll(arr) {
2      let a = arr[0];
3      let b = arr[1];
4      let sum = 0;
5      // compare
6      if(a < b){
7        for(let i  = a;i<=b;i++){
8          sum += i;
9        }
10     }
11     else{
12       for(let i  = b;i<=a;i++){
13         sum += i;
14       }
15     }
16    return sum;
17
18 }
19
20 sumAll([1, 4]);
```

→ O(1)

→ O(a-b)

a → b

sum(1,b)

→ a-b

[5-10] → (5+6+7+8+9+10)

→ [1 — 10] → 1+2+3+4 .. 10

✓ [1-4] = 1+2+3+4

[5,10]

5 10

[1-10] ⇒ AP ⇒ $\frac{(n)(n+1)}{2}$

$1+2+3+4 \ldots\ldots n-3+n-2+n-1+n$

$\rightarrow (n+1), (n+1), (n+1), (n+1) \ldots$

$\rightarrow \left(\frac{n}{2}\right)(n+1)$

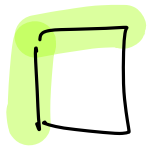
$[1-10] = \frac{(n)}{2} \times (n+1) = \frac{10 \times 11}{2} = 55$

$[1-4] = \frac{n}{2} \times (n+1) = \frac{4}{2} \times 5 = 10$

$= 55 - 10 = 45$

```
function sum1toN(x){
  return ((x)*(x+1))/2;
}


function sumAll(arr) {
    let a = arr[0];
    let b = arr[1];
    let sum = 0;
    // compare
    if(a < b){
      sum = sum1toN(b) - sum1toN(a-1)
    }
    else{
      sum = sum1toN(a) - sum1toN(b-1)
    }
  return sum;

}

sumAll([1, 4]);
```

3 min



TC → O(1)

Diff Two Arrays

Compare two arrays and return a new array with any items only found in one of the two given arrays, but not both. In other words, return the symmetric difference of the two arrays.

Note: You can return the array with its elements in any order.

Run the Tests (Ctrl + Enter)

Reset All Code

Get Help

Tests

- `diffArray([1, 2, 3, 5], [1, 2, 4, 5])` should return an array.
- `["diorite", "andesite", "grass", "dirt", "pink wool", "dead shrub"], ["diorite", "andesite", "grass", "dirt", "dead shrub"]` should return `["pink wool"]`.
- `["diorite", "andesite", "grass", "dirt", "pink wool", "dead shrub"], ["diorite", "andesite", "grass", "dirt", "dead shrub"]` should return an array with one item.
- `["andesite", "grass", "dirt", "pink wool", "dead shrub"], ["diorite", "andesite", "grass", "dirt", "dead shrub"]` should return `["diorite", "pink wool"]`.
- `["andesite", "grass", "dirt", "pink wool", "dead shrub"], ["diorite", "andesite", "grass", "dirt", "dead shrub"]` should return an array with two items.
- `["andesite", "grass", "dirt", "dead shrub"], ["andesite", "grass", "dirt", "dead shrub"]` should return `[]`.

☑ input and output

☐ writing the solution in english.

Step → * iterate on array 1
↳ check if ele is present in second array
NO
add in ans arr

→ * iterate on array 2 ✓
↳ same as above

- [ ] Write code ~ 9:18

```
function diffArray(arr1, arr2) {
  let diff = [];
  for(let i = 0;i<arr1.length;i++){
    if(arr2.includes(arr1[i]) == false){
      diff.push(arr1[i]);
    }
  }
  for(let i = 0;i<arr2.length;i++){
    if(arr1.includes(arr2[i]) == false){
      diff.push(arr2[i]);
    }
  }
  return diff;
}

diffArray([1, 2, 3, 5], [1, 2, 3, 4, 5]);
```

- [ ] TC → $O(n^2)$ ✓ , $O(n \log n)$ ✓

- [ ] Optimise → $O(n)$

```
if(arr2.includes(arr1[i]) == false){
  diff.push(arr1[i]);
}
```

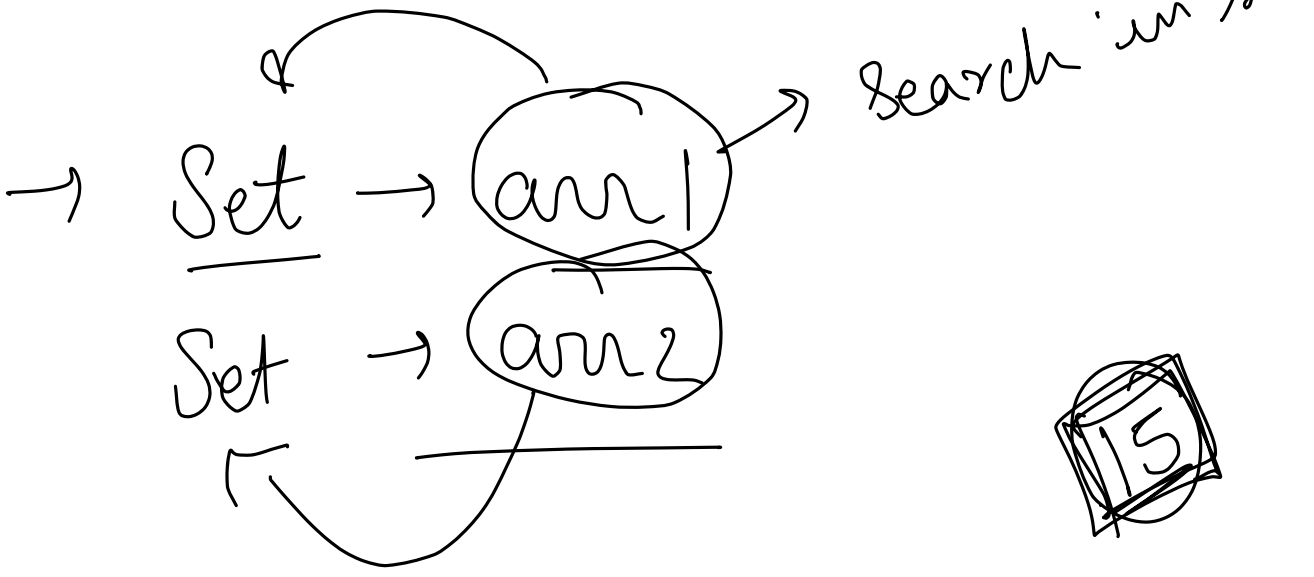
So many time

$O(n)$

yes

Set ~ Collection of element ☑ all element are unique

arr2
↳ Set
↳ element is present



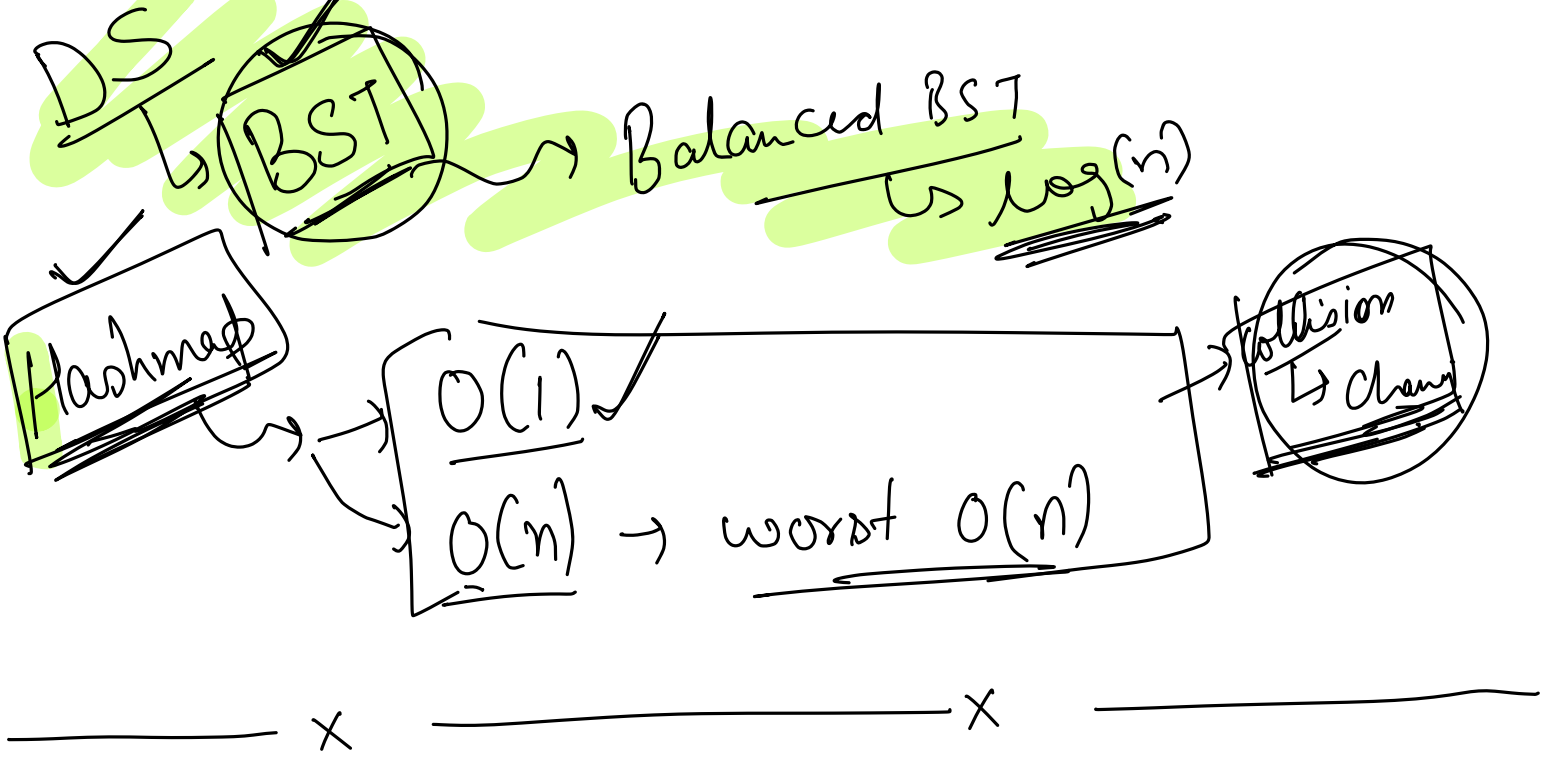

```
1   function diffArray(arr1, arr2) {
2     let diff = [];
3     let set1 = new Set(arr1);
4     let set2 = new Set(arr2);
5     for(let i = 0;i<arr1.length;i++){
6       if(set2.has(arr1[i]) == false){
7         diff.push(arr1[i]);
8       }
9     }
10    for(let i = 0;i<arr2.length;i++){
11      if(set1.has(arr2[i]) == false){
12        diff.push(arr2[i]);
13      }
14    }
15    return diff;
16  }
17
18  diffArray([1, 2, 3, 5], [1, 2, 3, 4, 5]);
```

n → log(n) → n x log(n)

n → log(n) → n log(n)

set

O(n log(n))



Break

Seek and Destroy

You will be provided with an initial array (the first argument in the `destroyer` function), followed by one or more arguments. Remove all elements from the initial array that are of the same value as these arguments.

Note: You have to use the `arguments` object.

Run the Tests (Ctrl + Enter)

Reset All Code

Get Help

Tests

- `destroyer([1, 2, 3, 1, 2, 3], 2, 3)` should return `[1, 1]`.
- `destroyer([1, 2, 3, 5, 1, 2, 3], 2, 3)` should return `[1, 5, 1]`.
- `destroyer([3, 5, 1, 2, 2], 2, 3, 5)` should return `[1]`.
- `destroyer([2, 3, 2, 3], 2, 3)` should return `[]`.
- `destroyer(["tree", "hamburger", 53], "tree", 53)` should return `["hamburger"]`.

☐ understand the Question →

☐ input, output clear

☐ f (arr, arg1, arg2 ...)

→ ? → Rest operator

☐ iterate over arr

check if ele is present restOfArgs

yes ← present restOfArgs → No

X

add in ansArr

(arg1, arg2, arg3 ....) ⇒

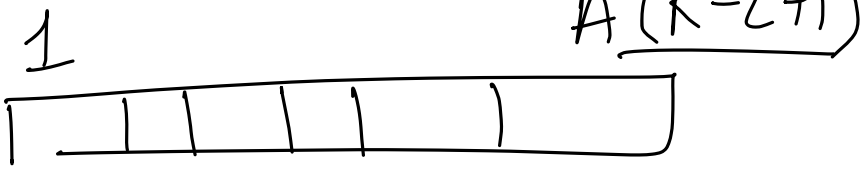
→ ... → array

```
1   function destroyer(arr,...rest) {
2     let set1 = new Set(rest);
3     let ansArr = [];
4     for(let i = 0;i<arr.length;i++){
5       if(set1.has(arr[i]) == false){
6         ansArr.push(arr[i])
7       }
8     }
9
10    return ansArr;
11  }
12
13  destroyer([1, 2, 3, 1, 2, 3], 2, 3 , 4, 5 , 6, 7 );
```

5 min

L R

Problem Statement



Amy has an array A of R-L+1 integers such that A1 = L, A2 = L+1, …. , AR-L+1 = R.

Anne gives her Q queries. Each query consists of two integers X and Y. Anne wants Amy to check if there exists at least one subsequence in A, such that the sum of subsequence lies between X and Y (both inclusive).

You are given T independent test cases.

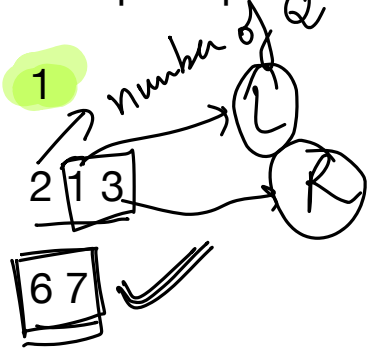
NOTE: Subsequence of an array can be obtained by erasing some (possibly zero) elements from the array. You can erase any elements, not necessarily going successively. The remaining elements preserve their order.

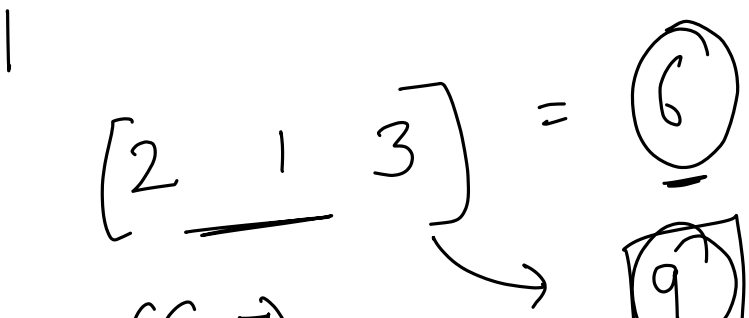
Constraints

1 <= T <= 3
1 <= Q <= 10^5
1 <= L <= R <= 10^9
1 <= X<= Y <= 10^18
All input values are integers.
Input Format

First-line contains T.
First line of each test case consists of a three space separated integers integer Q, L and R.
Next Q lines of each test case consists of two space separated integers X and Y.
Output Format

Print in a newline for each query: 1 if there exists a required subsequence otherwise print 0.
Sample Input 1





Sample Output 1

1

0

Explanation of Sample 1

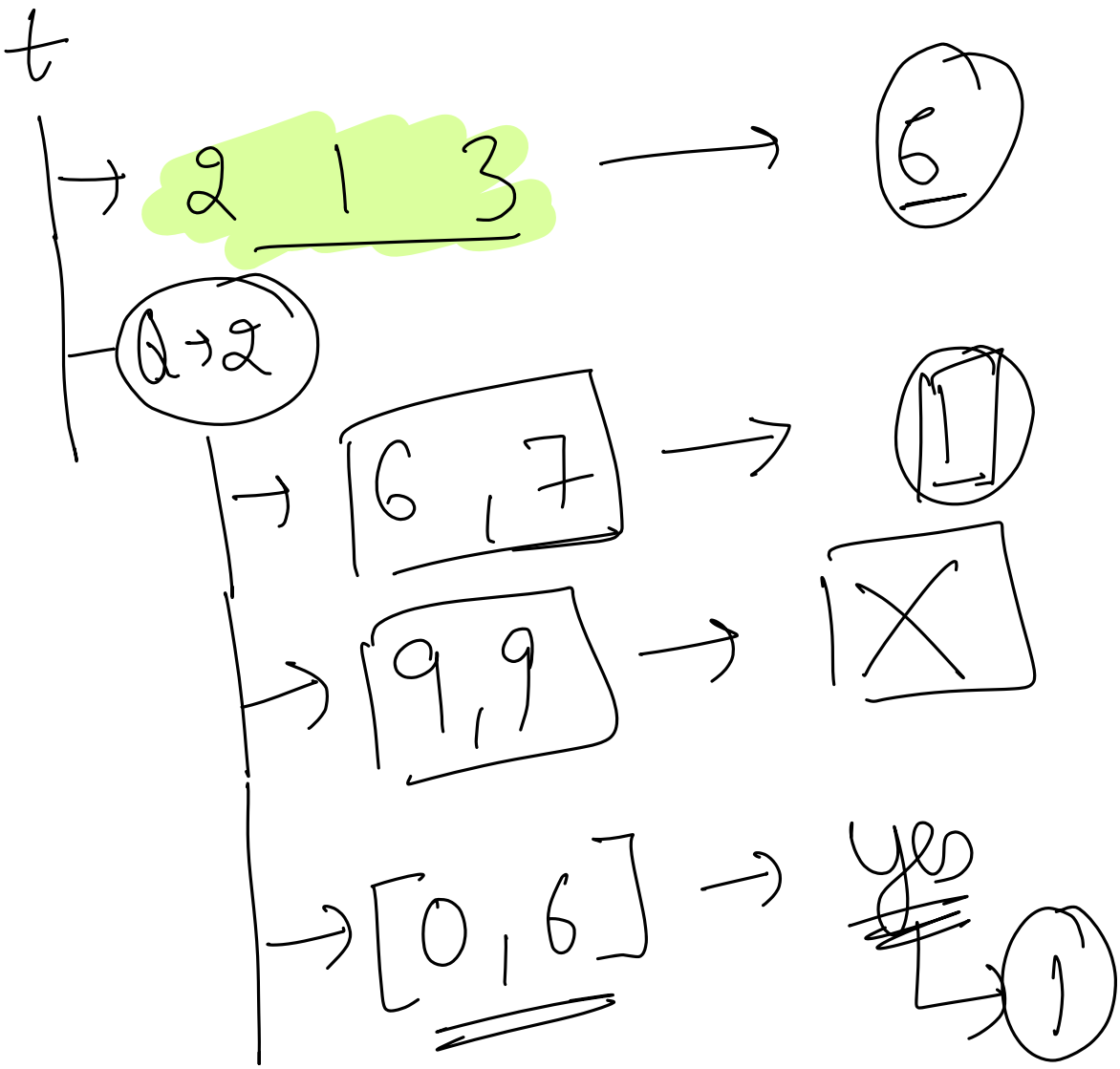
For the first query, subsequence S = { A1, A2, A3} has sum 6 (A1+A2+A3 = 1+2+3 = 6). So, there is one possible subsequence whose sum lies between X = 6 and Y = 7.

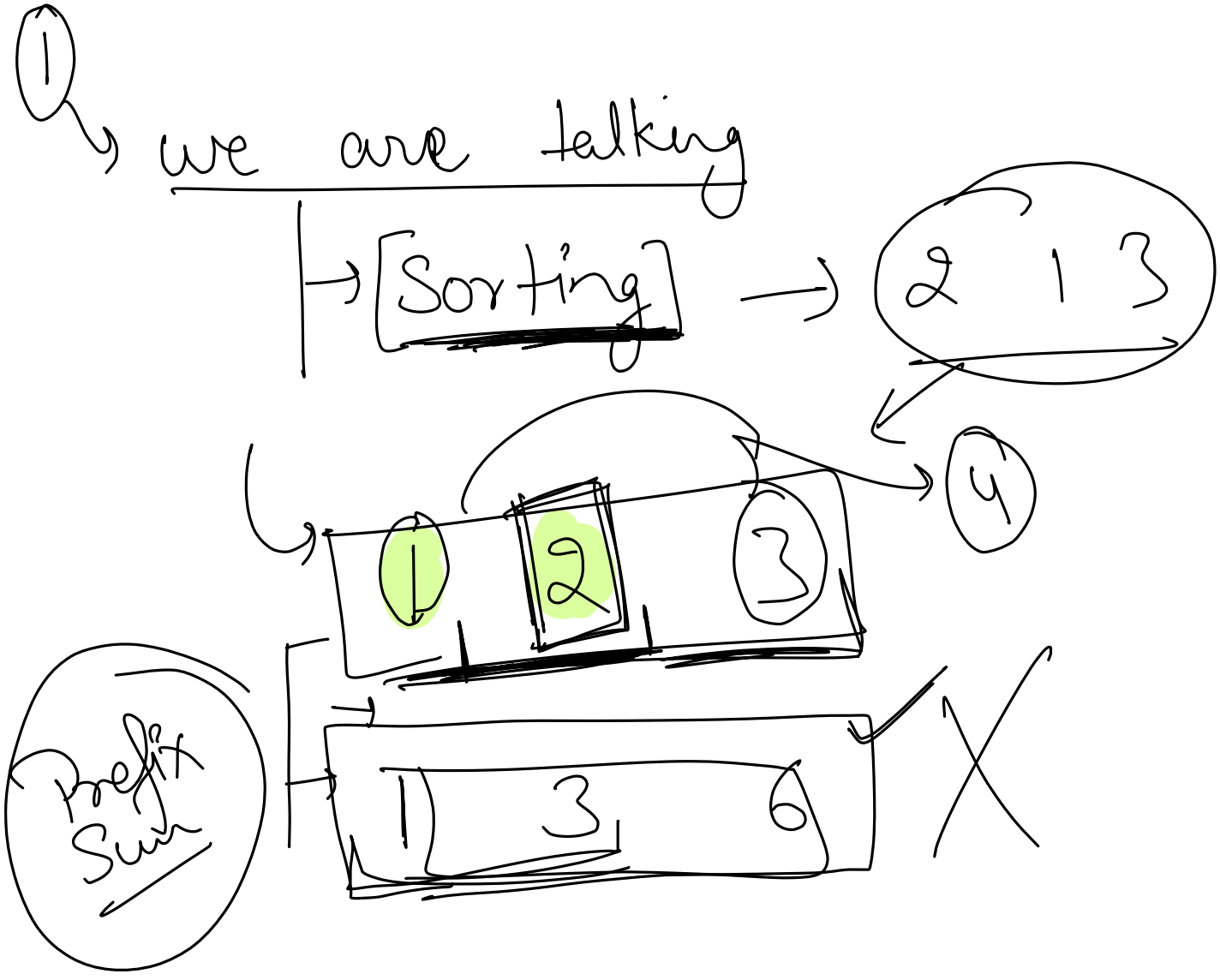
For 2nd query, there is no possible subsequence.

///////other solution for this problem/////
// your code goes here

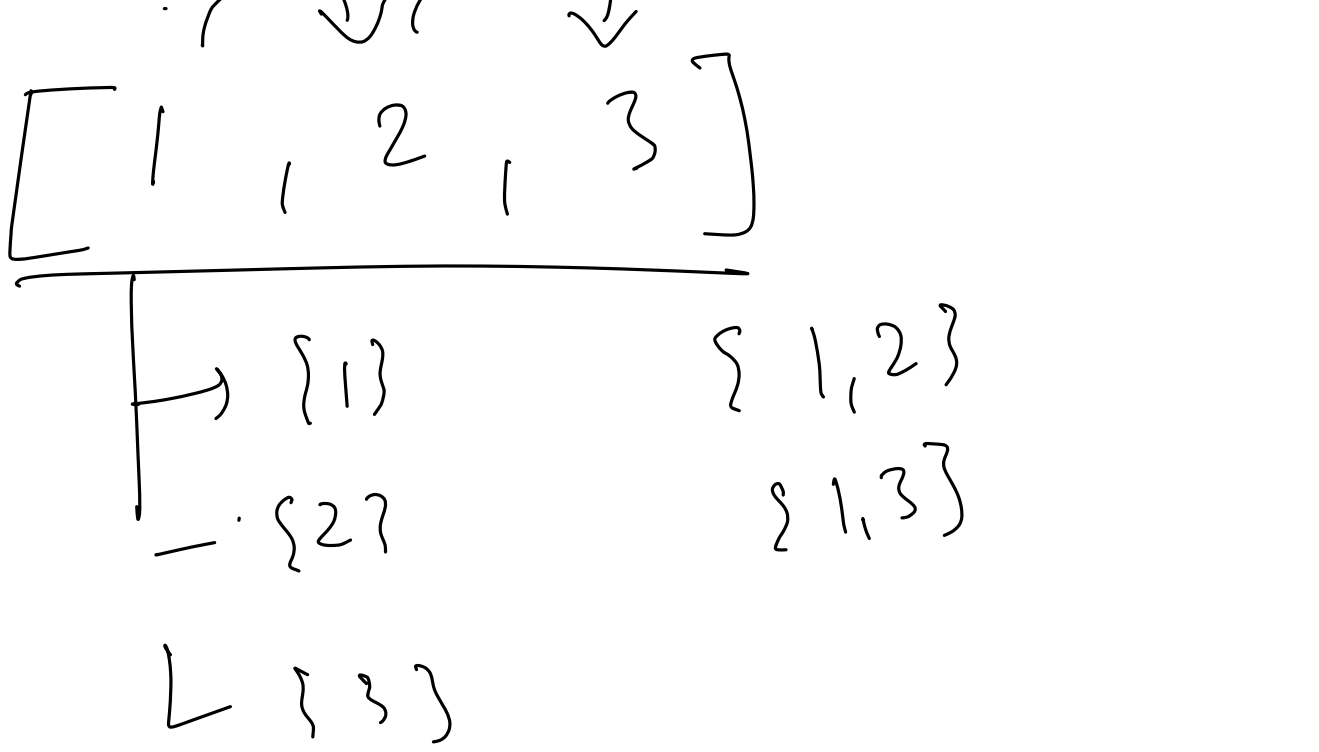
```
function driver(l, r, queries){
    var sum = 0
    var res = []
    for(var i = l; i <= r; i++){
        sum += i
    }
    queries.forEach(query => {
        if(sum >= query[0] || l >= query[0]) res.push(1)
        else res.push(0)
    });
    return res
}

let tc = parseInt(readline());
while(tc--){

      let queries = readline().split(" ").map(Number);
    let r = parseInt(readline());
      let l = parseInt(readline());
  console.log(driver(l, r, queries))
```

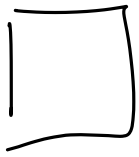
```
}
```


t
2 1 3
6
a→2
6 , 7
1
9 , 9
X
[0 , 6]
yes
1

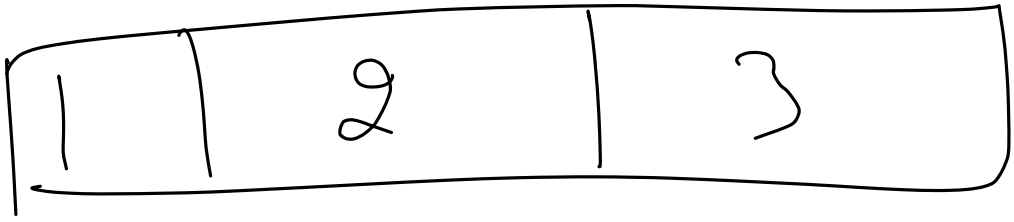


any Subsequence where Sum is



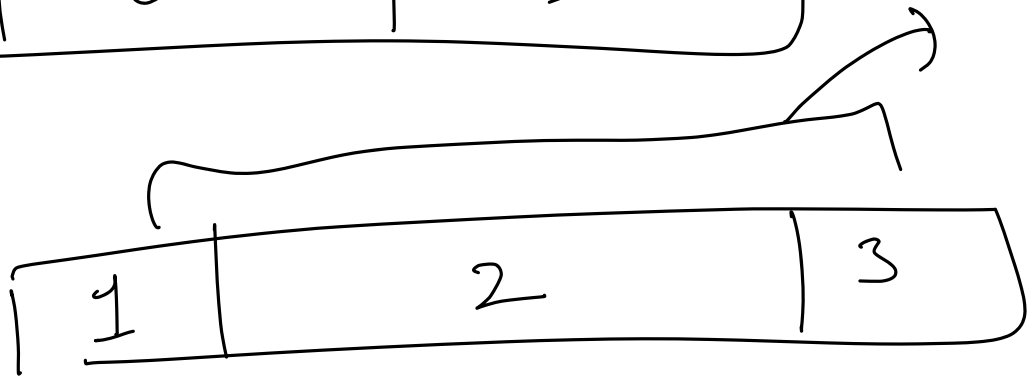


1 3

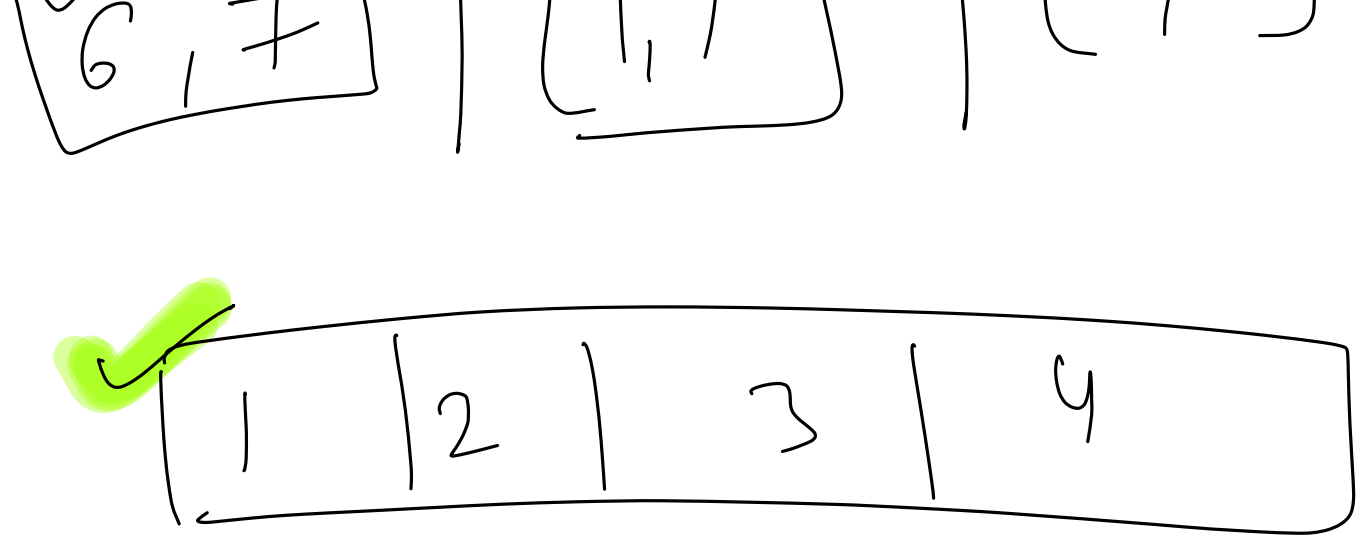


Q → ~~3~~

L → 1



R → 3



- [ ] find sum of all subsequences
  - ↳ put them in Set
- [ ] Matrix → H.W