**BIT MANIPULATION**

Table of Contents

[**Important Codes** 1](#_Toc171628510)

[Binary Search Tree Implementation: 1](#_Toc171628511)

[LEVEL 1: **EASY** 2](#_Toc171628512)

[1. Question 2](#_Toc171628513)

[2. Question 2](#_Toc171628514)

[LEVEL 2: **Medium** 3](#_Toc171628515)

[1. Array Removal 3](#_Toc171628516)

[LEVEL 3: **Difficult** 4](#_Toc171628517)

[**SOLUTIONS:** 5](#_Toc171628518)

[**LEVEL 1:** 5](#_Toc171628519)

[**LEVEL 2:** 6](#_Toc171628520)

# **Important Codes**

## Binary Search Tree Implementation:

# LEVEL 1: **EASY**

### Question

Link: <Link>

### Question

Link: <Link>

# LEVEL 2: **Medium**

### Array Removal

Link: <https://www.codechef.com/problems/ARRREM>

# LEVEL 3: **Difficult**

# **SOLUTIONS:**

## **LEVEL 1:**

1. Same tree

class Solution:

    def isSameTree(self, p: Optional[TreeNode], q: Optional[TreeNode]) -> bool:

        if p is None and q is None:

            return True

        if (p s None) or (q is None):

            return False

        if p.val!=q.val:

            return False

        return self.isSameTree(p.left,q.left) and self.isSameTree(p.right,q.right)

dd

## **LEVEL 2:**

1. Array removal

Problem 2, understand this and make your own explanation

[](https://www.youtube.com/watch?v=IqKZE6nwzko)

for \_ in range(int(input())):

    n = int(input())

    arr = list(map(int,input().split()))

    ans=float('inf')

    for i in range(32):

        val = (1<<i)-1

        num\_taken = 0

        bit\_or = 0

        for num in arr:

            if num>val:

                continue

            num\_taken+=1

            bit\_or = bit\_or | num

        if(bit\_or == val):

            num\_to\_remove = n - num\_taken

            ans = min(ans,num\_to\_remove)

    print(ans)