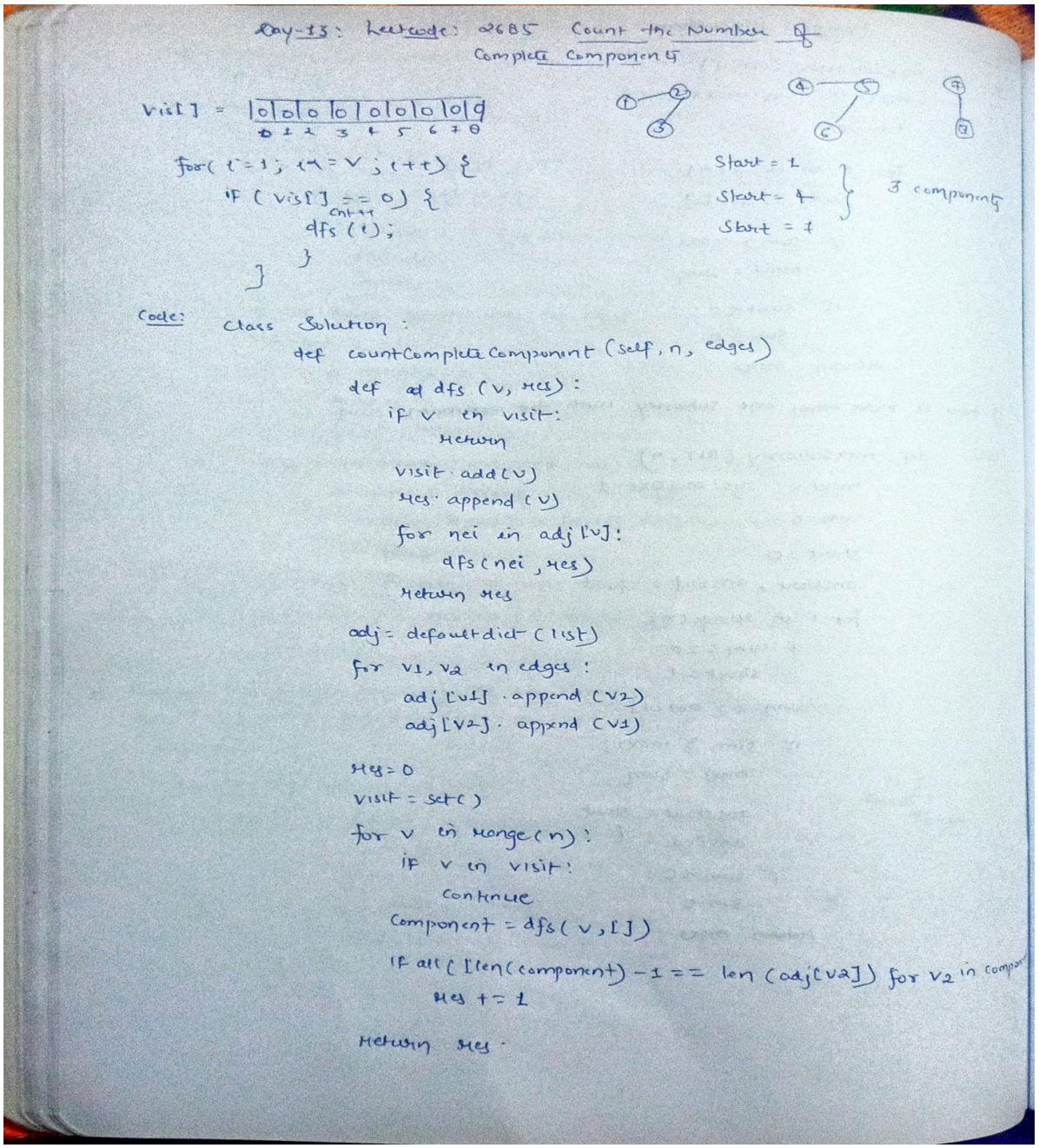
Day_13: DSA



Day-13: leetcode: 2685 Count the Number of Complete Components

vis[] = |0|0|0|0|0|0|0|0|0|0|0|
0 1 2 3 4 5 6 7 8


```
for(i=1; i<=V; i++) {
    if (vis[i] == 0) {
        cnt++
        dfs(i);
    }
}
```

Start = 1
Start = 4
Start = 7
} 3 components

Code:

```
class Solution:
    def countCompleteComponent(self, n, edges)
        def dfs(v, res):
            if v in visit:
                return
            visit.add(v)
            res.append(v)
            for nei in adj[v]:
                dfs(nei, res)
            return res

        adj = defaultdict(list)
        for v1, v2 in edges:
            adj[v1].append(v2)
            adj[v2].append(v1)

        res = 0
        visit = set()
        for v in range(n):
            if v in visit:
                continue
            component = dfs(v, [])
            if all([len(component) - 1 == len(adj[v2]) for v2 in comp
                res += 1

        return res
```

SOL

- Sort an array 0's, 1's and 2's

arr[] = | 0 | 1 | 2 | 0 | 1 | 2 | 1 | 2 | 0 | 0 | 0 |

→ Better Soln

```
cnt0=0  cnt1=0  , cnt2=2
for(i=0 ; i<n; i++)
   { if (a[i]==0) cnt0++ ;
     elseif (a[i] == 1) cnt1++;          TC - O(2N)
     else  cnt2++;
   }
  for (i=0 ; i<cnt0 ; i++) a[i]=0
  for ( i=cnt0 ; i<cnt0+cnt1; i++) a[i]=1
  for( i=cnt0+cnt1 ; i<n ; i++) a[i]=2
```

Optimal Soln :

```
def sortArray(arr):
    low=0
    mid = 0
    high = len(arr)-1
    while mid <= high:
        if arr[mid] == 0
        arr[low] ,arr[mid] = arr[mid] ,arr[low]
        low += 1
        mid += 1
        elif arr[mid] == 1:
            mid += 1
        else:
            arr[mid] , arr[high] = arr[high] , arr[mid]
            high -= 1
```