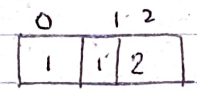
Date: / /
Page:

Day - 92

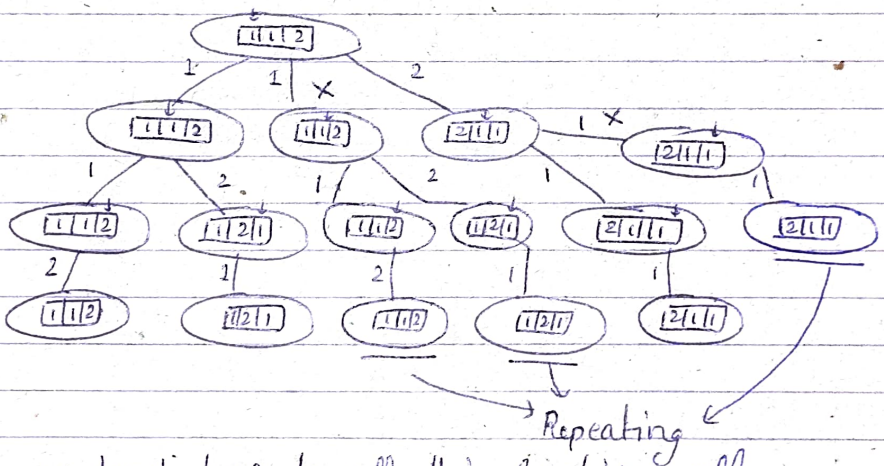
Recursion - 13

* Permutation - 2 :



⇒ We have to return all unique permutation.

⇒ By using old method,



⇒ So, we don't have to call this function call.

⇒ So, we only call function when both the no. are different

⇒ Also, we don't have to call function if the no. is already came.

⇒ So, to check which no. is come or not, we will use an array of size 21. because according to question, the no. in the array is b/w $-10 < num < 10$.

Date: / /
Page:

Code

```
void permut ( vector< int > & arr , vector < vector <int>>
& ans , int index ){
        if( index == arr.size() ){
            ans.push_back( arr );
            return;
        }
    vector<bool> use(21,0);
    for(i= index; i<arr.size(); i++){
        if( use[arr[i] +10 ] == 0 ){
            swap( arr[index] , arr[i] );
            permut( arr, ans, index+1 );
            swap( arr[index] , arr[i] );
            use[ arr[i] +10] = 1;
        }
    }
}
```

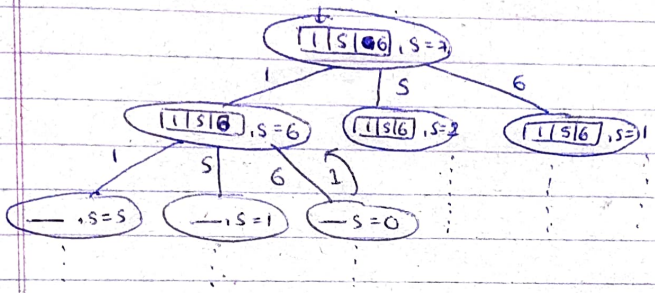
* Ways to sum N:

| 1 | 5 | 6 |

sum = 7

⇒ In this question, orders matter that means —
1 + 6 & 6 + 1
both are different.

⇒ we will consider all the elements every time.



Code

```
int way ( ~~vector~~ int arr[] , int m , int sum ){
    if (sum == 0)
        return 1;
    if (sum < 0 ) return 0;
    int ans = 0;
    for( i= 0; i< m; i++){
        ans += way (arr ,m, sum- arr[i]);
    }
    return ans;
}
```