Day - 1 05-09-2021

level - Medium

Q1 3Sum

Input : nums = [-1, 0, 1, 2, -1, -4]
output : [[-1, -1, 2], [-1, 0, 1]]

Brute force ($O(n^3)$)

// using 3 pointer i, j and k
// use for loop i=0, j=i+1, k=j+1
// initialize a counter variable where num[i]+num[j]+num[k] == 0, increase counter by 1 //.

Optimal soln

// two pointer algo (low and high)
// sort the vector array
// moves for a ((b+c) == -a)
// for loop upto n-2, because we find triplet.
// if (i == 0 || (i > 0 && num[i] != num[i-1]))
then increase low by 1.

```
⇒ vector<vector<int>> threeSum(vector<int>& num)
  sort(num.begin(), num.end());
  vector<vector<int>> res;

// moves for a
for(int i=0; i<num.size()-2; i++)
{
  if(i==0 || (i>0 && num[i] != num[i-1])) {
    int low = i+1, high = num.size()-1, sum = 0 - num[i];
    while(low<high)
    {
      if(num[low]+num[high] == sum)
```

```
{
    vector<int> temp;
    temp.push_back(nums[i]);
    temp.push_back(nums[low]);
    temp.push_back(nums[high]);
    res.push_back(temp);

    while (low<high && nums[low] == nums[low+1]) low++;
    while (low<high && nums[high] == nums[high-1]) high--;

    low++; high--;
}
    else if (nums[low] + nums[high] < sum)
      low++;
    else high--;
}
  }
    return res;
};
}
```

Again optimal soln (in above there is runtime error)

```
vector<vector<int>> threeSum(vector<int>& nums) {
 vector<vector<int>> output;
 sort(nums.begin(), nums.end());
for(int i=0; i<nums.size(); ++i) {
 if(i!=0 && nums[i]==nums[i-1])
    continue;
  int j=i+1;
  int k=nums.size()-1;
  while(j<k) {
  if(nums[i]+nums[j]+nums[k]==0)
   {
```

```
            output.push_back({nums[i], nums[j], nums[k]});
            ++j;
            while (j<k && nums[j] == nums[j-1])
            ++j;
        }
        else if(nums[i]+nums[j]+nums[k] < 0) {
            ++j;
        } else {
            --k; }
        }
    }
    return output;
  }
};
```

Q-2 4sum - Medium

⟹ 4Sum

Input: nums = [1, 0, -1, 0, -2, 2], target = 0

output: [[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]

```
// Sort the array
// two pointer high and low
// fix high and move low
vector<vector<int>> fourSum(vector<int>& nums, int target
{ vector<vector<int>> res;
  sort(nums.begin(), nums.end());
  int n = nums.size();
  for(int i=0; i<n; i++)
  {
    for(int j=i+1; j<n; j++)
    {
      int tar_2 = target - (nums[i]+nums[j]);
```

```
int low = j+1; int high = n-1;
while (low < high) {
  int result = nums[low] + nums[high];
  if (result < target-2)
    low++;
  else if (result > target-2)
    high--;
  else {
    vector<int> arr(4,0);
    arr[0] = nums[i];
    arr[1] = nums[j];
    arr[2] = nums[low];
    arr[3] = nums[high];
    res.emplace_back(arr);   // emplace work of push back.

// processing the duplicate of num 3
while (low<high && nums[low]==arr[2]) ++low;
// processing the duplicate of num 4, num 2, num 1
while (low<high && nums[high]==arr[3]) --high;
  }
}
while (j+1<n && nums[j+1]==nums[j]) ++j;
}
while (i+1<n && nums[i+1]==nums[i]) ++i;
}
return res;
}
};
```

Recursion

→ function call itself called recursion.
→ It follow PMI (Principal of mathematical Induc.
→ Hypothesis
→ Induction
→ Base condition (use to stop the fn)

Q-1 find sum of element of given vector.

```
// Bottom-up approach

int findSum (vector<int>v, int i)
{
  if (i==n)        <- Base condition
    return 0;
  return v[n] + findSum(v, n-1);   <- Induction, hypothesis
}

// top-down approach

int findSum (vector<int>v, int i)
{
  if (i==n)
    return 0;
  return v[i] + findSum(v, i+1);
}
```

Q2 find sum of first n natural number.

Bottom-up

```
int sumOfNatural (int n) {
if (n <= 1)
    return n;
    return n + sumOfNatural (n-1);
}
```

top-down

```
int SumOfNatural (Vector<int> v, int i)
  if (i <= 1)
    return i;
    return v[i] + SumOfNatural (v, i+1);
```