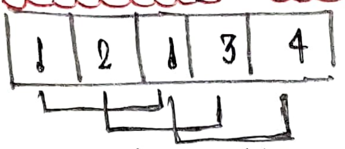
Two Pointer

01 subarrays with distinct integer

Eg



$\rightarrow$ atmost 3

1 + 2 + 3 + 4 + 3

= 13.

Now we use

atmost (3) − atmost (2) = ans

= 13 − 10

= $\boxed{8}$.

Eg.

1 2 1 3

$\hookrightarrow$ (1 + 2 + 3 + 4) = 10

$\hookrightarrow$ 1 3 4

(3) = 8

$\rightarrow$ atmost 2

= (10)

~~+~~ (18)

total

02 Two sum (without map).

Eg [2, 6, 5, 8, 11] target = 14.

so, sort [2 5 6 8 11]

left right

2 + 11 = 13 < 14 left ++.

5 + 11 = 16 > 14 right −−

5 + 8 = 13 < 14 left ++.

6 + 8 = 14 = 14 yes.

• yeh yes or no yaha puche uske liye ye best way hai.

$\rightarrow$ Now use hashing (map)

arr [] = [2 | 6 | 5 | 8 | 11]

target = 14

0 1 2 3

2 (6) 5 (8) 11

2 12 X

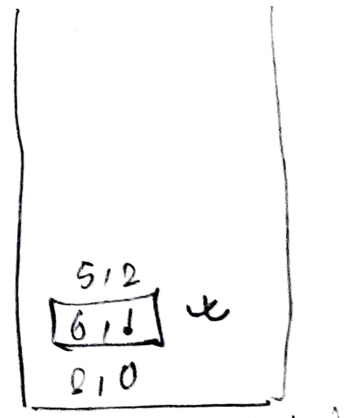
6 8 8 X

5 9 9 X

8 6 ✓ yes

TC = O(n log n)

SC = O(n).



return

[1 | 3]

yes

03 3 sum

arr [] = [−1, 0, 1, 2, −1, −4]

ans = [−1, 2, −1]

[0, 1, −1] ans

return unique triplets.

arr [i] + arr [j] + arr [k] = 0

i != j != k.

que

arr[] = [-2, -2, -2, -1, -1, -1, 0, 0, 0, 2, 2, 2, 2]

i   i   j   j   j   k   k k

Fixed   sorted order

$-2 -2 + 2 = -2 < 0$

$-1 -2 + 2 = -1 < 0$ → [-2, 0, 2]

$-2 + 0 + 2 = \boxed{0} = 0.$ → [-1 -1 2].

if (k < j) loop ended.   → [0 0 2].

now move i

04   4 sum

arr[] = after sorting.   target = 8

[1 1 1 2 2 2 3 3 3 4 4 4 5 5]

i j k   l

* num[i] + num[j] + num[k] + num[l] == target.

* i < j < k < l.

05   4 sum II

- Yaha ye logic lagana hai   A[i] + B[j] = -(C[k] + D[l]).
- here we put a part in map and find it in that map of that -ve part.

06   3 sum closest

arr = [-1 -3 2 4 5]

sort it [-3 -1 2 4 5]

sum = 2.   i j   k   difference

$-3 -1 + 5 = +1 < 2$ = 1

$-3 + 2 + 5 = 4 > 2$ = 2

$-3 + 2 + 4 = 3 > 2$ = 1

TWO POINTER BEST QUESTIONS

1.) SUBARRAY SUM EQUALS K

```
class Solution {
public:
    int subarraySum(vector<int>& nums, int goal) {
        int ans=0;
        int psum=0;
        unordered_map<int,int> mpp;
        mpp[0]=1;
        for(auto it:nums){
            psum=psum+it;
            if(mpp.find(psum-goal)!=mpp.end()){
                ans=ans+mpp[psum-goal];
            }
            mpp[psum]++;
        }
        return ans;
    }
};
```

2.) BINARY SUBARRAYS WITH SUM

FIRST APPROACH

```
class Solution {
public:
    int numSubarraysWithSum(vector<int>& nums, int goal) {
        long long int i=0;
        long long int j=0;
        long long int sum=0;
        long long int ans=0;
        while(j<nums.size()){
            sum=sum+nums[j];
            while(i<=j && sum>goal){
                sum=sum-nums[i];
                i++;
            }
            ans=ans+(j-i+1);
            j++;
        }
        return ans;
    }
};
```

SECOND APPROACH

```
class Solution {
public:
    int func(vector<int>& nums, int goal) {
        long long int i=0;
        long long int j=0;
        long long int ans=0;
        long long int sum=0;
        while(j<nums.size()){
            sum=sum+nums[j];
            while(i<=j && sum>goal){
                sum=sum-nums[i];
                i++;
            }
            ans=ans+(j-i+1);
            j++;
        }
        return ans;
    }
    int numSubarraysWithSum(vector<int>& nums, int goal) {
        return func(nums,goal)-func(nums,goal-1);
    }
};
```

3.) TWO SUM

CODE

```
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        unordered_map<int,int> mp;
        vector<int> ans;
        for(int i=0;i<nums.size();i++){
            int rest=target-nums[i];
            if(mp.find(rest)!=mp.end()){
                ans.push_back(i);
                ans.push_back(mp[rest]);
                return ans;
            }
            mp[nums[i]]=i;
        }
        return {};
    }
};
```

4.) 3 SUM

CODE

```cpp
class Solution {
public:
    vector<vector<int>> threeSum(vector<int>& nums) {
        vector<vector<int>> ans;
        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){
                int sum=nums[i]+nums[j]+nums[k];
                if(sum<0){
                    j++;
                }
                else if(sum>0){
                    k--;
                }
                else{
                    vector<int> temp={nums[i], nums[j], nums[k]};
                    ans.push_back(temp);
                    j++;
                    k--;
                    while(j<k && nums[j]==nums[j-1])
                    j++;
                    while(j<k && nums[k]==nums[k+1])
                    k--;
                }
            }
        }
        return ans;
    }
};
```

5.) 4 SUM

CODE

```
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {
        vector<vector<int>> ans;
        sort(nums.begin(), nums.end());
        for(int i=0;i<nums.size();i++){
            if(i>0 && nums[i]==nums[i-1])
            continue;

            for(int j=i+1;j<nums.size();j++){
                if(j!=i+1 && nums[j]==nums[j-1])
                continue;

                int k=j+1;
                int l=nums.size()-1;
                while(k<l){
                    long long int sum=nums[i];
                    sum+=nums[j];
                    sum+=nums[k];
                    sum+=nums[l];
                    if(sum==target){
                        vector<int> temp={nums[i], nums[j], nums[k], nums[l]};
                        ans.push_back(temp);
                        k++;
                        l--;
                        while(k<l && nums[k]==nums[k-1])
                        k++;
                        while(k<l && nums[l]==nums[l+1])
                        l--;
                    }
                    else if(sum<target){
                        k++;
                    }
                    else{
                        l--;
                    }
                }
            }
        }
        return ans;
    }
};
```

6.) 4 SUM II

CODE

```
class Solution {
public:
    int fourSumCount(vector<int>& A, vector<int>& B, vector<int>& C,
vector<int>& D) {
        map<int,int> mp;
        for(int i:A){
            for(int j:B){
                int sum=i+j;
                mp[-sum]++;
            }
        }
        int count=0;
        for(int k:C){
            for(int l:D){
                int sum=k+l;
                count=count+mp[sum];
            }
        }
        return count;
    }
};
```

7.) TWO SUM II – INPUT ARRAY IS SORTED

CODE

```
class Solution {
public:
    int threeSumClosest(vector<int>& nums, int target) {
        sort(nums.begin(), nums.end());
        int diff=INT_MAX;
        int ans=0;
        for(int i=0;i<nums.size();i++){
            int first=nums[i];
            int s=i+1;
            int e=nums.size()-1;
            while(s<e){
                if(first+nums[s]+nums[e]==target){
                    return target;
                }
                else if(abs(first+nums[s]+nums[e]-target)<diff){
                    diff=abs(first+nums[s]+nums[e]-target);
                    ans=first+nums[s]+nums[e];
                }
                if(first+nums[s]+nums[e]>target)
                e--;
                else
                s++;
            }
        }
        return ans;
    }
};
```

8.) 3 SUM CLOSEST

CODE

```
class Solution {
public:
    int threeSumClosest(vector<int>& nums, int target) {
        sort(nums.begin(), nums.end());
        int diff=INT_MAX;
        int ans=0;

        for(int i=0;i<nums.size();i++){
            int first=nums[i];
            int s=i+1;
            int e=nums.size()-1;

            while(s<e){
                if(first+nums[s]+nums[e]==target){
                    return target;
                }
                else if(abs(first+nums[s]+nums[e]-target)<diff){
                    diff=abs(first+nums[s]+nums[e]-target);
                    ans=first+nums[s]+nums[e];
                }
                if(first+nums[s]+nums[e]>target)
                e--;
                else
                s++;
            }
        }
        return ans;
    }
};
```

9.) MINIMIZE MAXIMUM PAIR SUM IN ARRAY

CODE

```
class Solution {
public:
    int minPairSum(vector<int>& nums) {
        sort(nums.begin(), nums.end());
        int ans=INT_MIN;
        for(int i=0;i<nums.size()/2;i++){
            ans=max(ans, nums[i]+nums[nums.size()-1-i]);
        }
        return ans;
    }
};
```