4 Sum

18. 4Sum

Medium 10.3K 1.2K

Companies

Given an array `nums` of `n` integers, return *an array of all the* ***unique*** *quadruplets* `[nums[a], nums[b], nums[c], nums[d]]` such that:

- `0 <= a, b, c, d < n`
- `a`, `b`, `c`, and `d` are **distinct**.
- `nums[a] + nums[b] + nums[c] + nums[d] == target`

You may return the answer in **any order**.

Example 1:

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

Example 2:

> **Input:** nums = [2,2,2,2,2], target = 8
> **Output:** [[2,2,2,2]]

nums[] = { 1, 0, -1, 0, -2, 2 } target = 0

{ 1, -1, -2, 2 } , { 1, -1, 0, 0 } , { -2, 2, 0, 0 }

0

Brute force:

nested loops → distinct → set → store the values.

```
for(i=0, i<n, i++)
    for(j=i+1; j<n; j++)
        for(k=j+1; k<n; k++)
            for(l=k+1; l<n; l++)
                if(sum == target)
                    set.push(vector);
```

T.C. = O(N^4)

S.C. = O(2N)

We have to return distinct array of the 4 element we

sum = target.

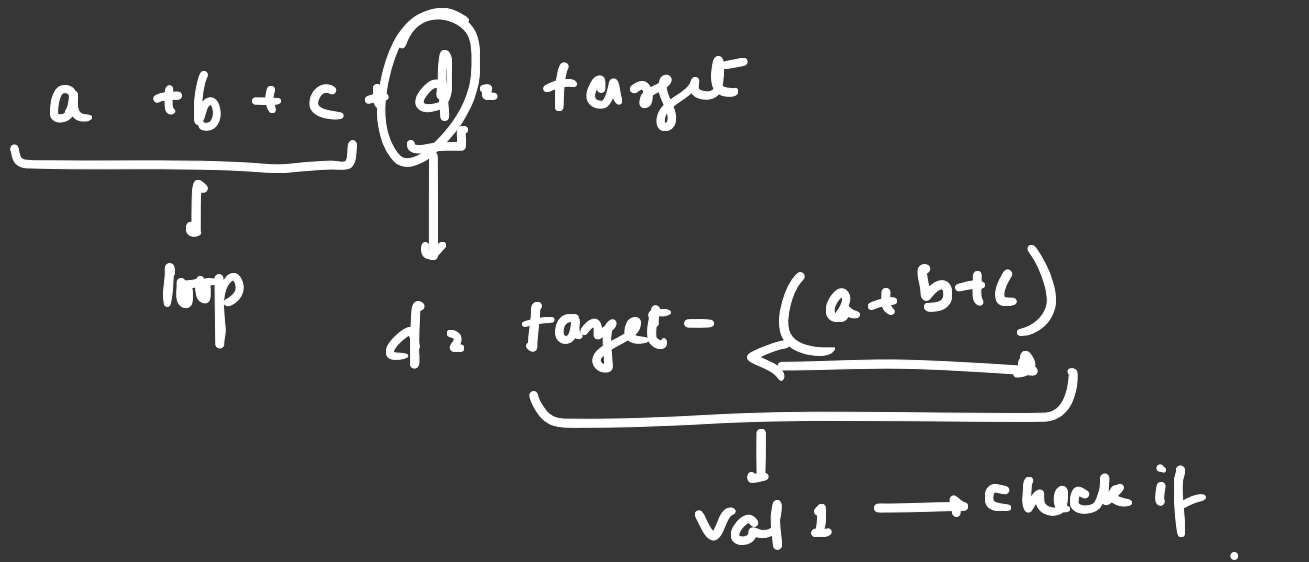
As distinct is asked we store the array in set ~~and~~ to make it distinct. We push the array in sorted form in set to easily distinguish bet^n distinct array.

We we are doing we are iterating at every element in array and summing them up to match the target

Set < vector (int) > st ⟶ Store the distinct quadruplets

T.C. :- $O(N^4)$ ⟶ $O(N^3)$

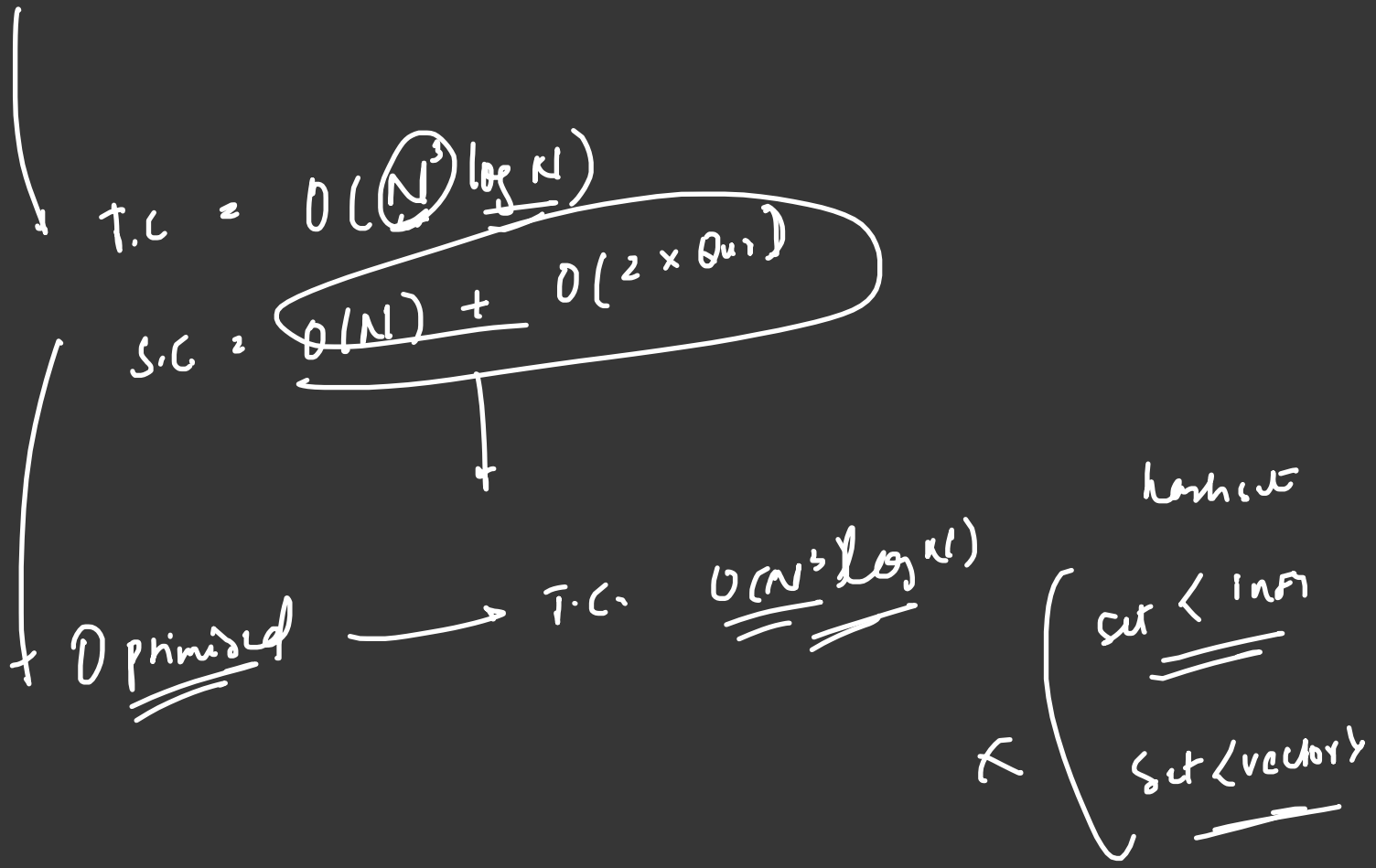
What we can do is



4 1 2 7 3 4 5 target = 16

unordered map → Because we need distinct element

```
for (i=0, i<n, i++)
    for (j=i+1, j<n, j++)                    → unordered map<int, int>
        for (k=j+1, k<n, k++)
            neededvalue = target - (arr[i] + arr[j] + arr[k]).
            if (map found(neededvalue) != map.end())
                vector temp = {arr[i], arr[j], arr[k], neededvalue};
                sort.
                map insert(arr[k]), set. temp.
```


T.C = O(N³ log N)
S.C = O(N) + O(2 × Qu)
Optimized → T.C. O(N³ log N)
Hashset
Set < INT >
Set < vector >

4 1 2 7 3 4 5 target = 16

Sort →

1 2 3 4 4 5 7

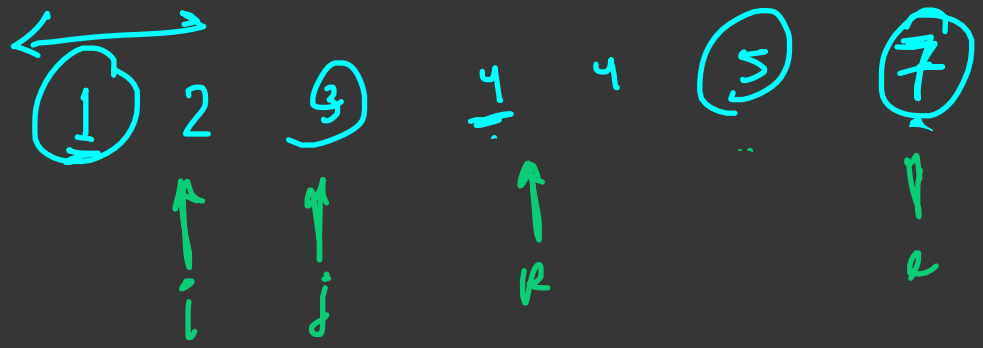
i j k l

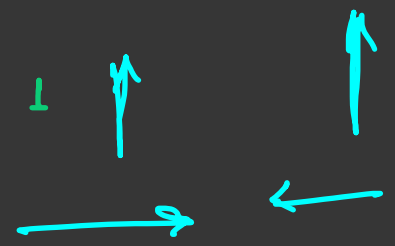
4th value

{7, 1, 4, 4}

{5, 3, 4, 4}

{7, 1, 5, 3}





7+4 = 11

7+5 = 12

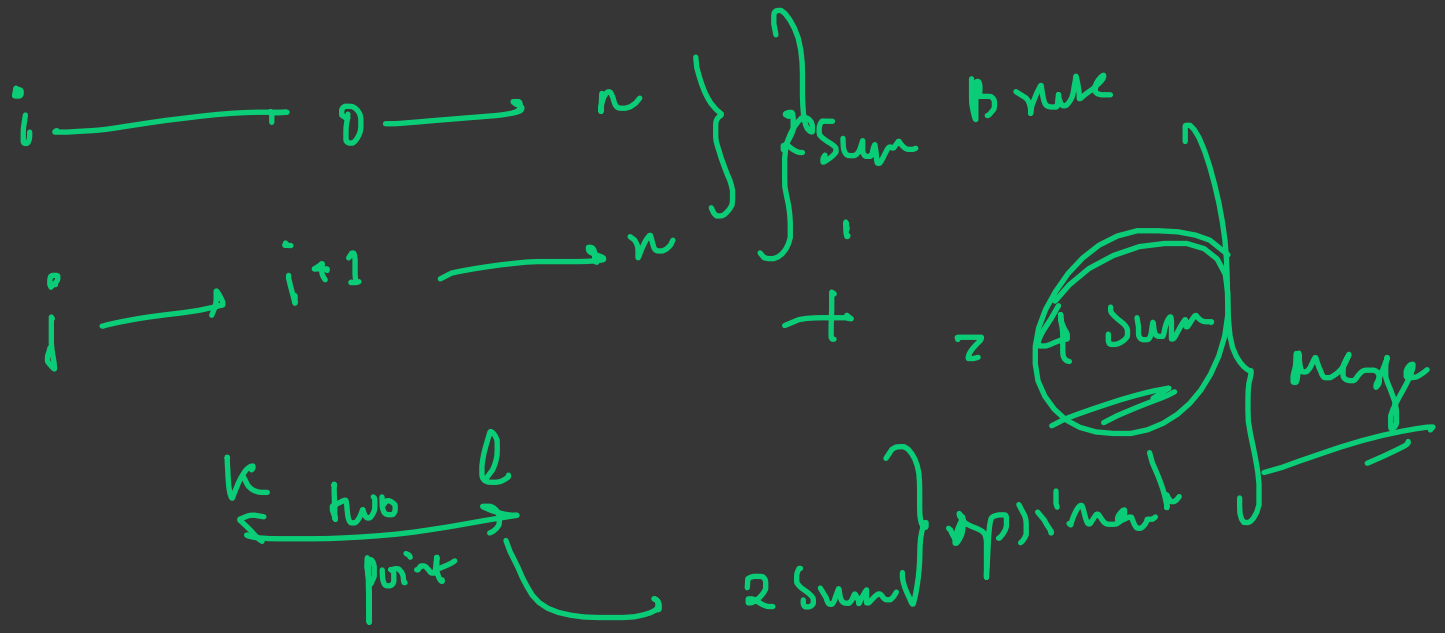
Sum = 1+2 = 3 + 10 (k+1)

Sum = 3+3 = 6

16

{1, 3, 5, 7}

{1, 4, 4, 7}

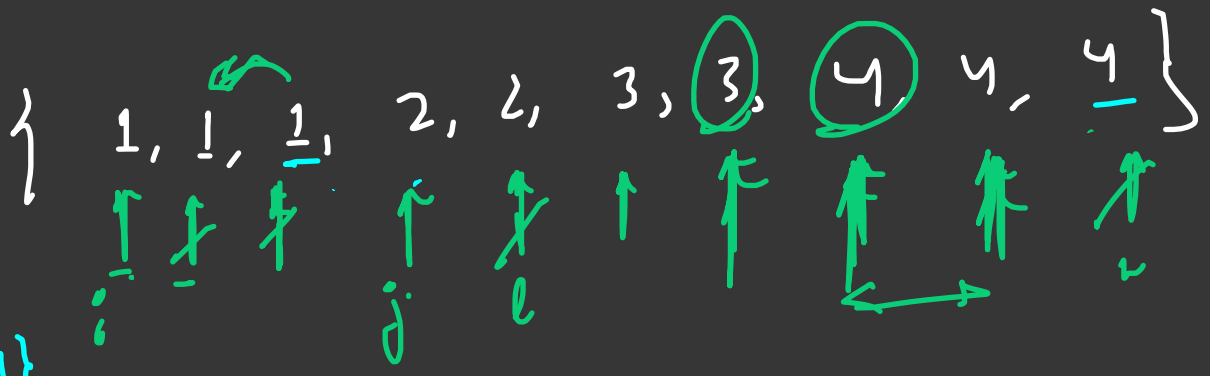


eg. arr[ ] : { 4, 3, 3, 4, 4, 2, 1, 2, 1, 1 }

target = 9

Optimal Approach.

- Sort the array:



sum = {1, 1, 3, 4}

{1, 2, 2, 4}

2 + 4 + 1 = 7

2 + 2 · 4 + 1 · 6 = 1

1 + 2 + 2 + 4 = 9

1 + 2 + 3 + 4 = 10

- To have distinct array we skip the duplicate element in every iterator

Pseudocode

T.C. : $O(N^3)$

S.C. : $O(1)$

```
for (i=0; i<n; i++)
    if(i>0 && arr[i] == arr[i-1]) continue;
    for(j=i+1; j<n; j++)
        if(j>i+1, && arr[j] == arr[j-1]) continue;
        int low = j+1, high = n-1;
        while(low<high)
            sum = ____
            if(sum == target) ans.push(a,b,c,d)
            low++, high--;
            // skip duplicate
            while(low<high && arr[low] == arr[low-1])
                low++;
            while(low<high && arr[high] == arr[high+1])
                high--;
```