18. 4Sum

Medium | Topics | 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]]
```

Constraints:

- `1 <= nums.length <= 200`
- `-10^9 <= nums[i] <= 10^9`
- `-10^9 <= target <= 10^9`

Approach 1:

T(n) : O(n^4)
S(n) : set to check for duplicate quadraplets

Approach 2:

keeping a and b and then doing 2Sum implementation with required value as Target - (a+b)

T(n) : O(n^3)
S(n) : 2sets

This implementation results slower running time which leads to TLE error becoz of sets and their respective function calls and multiple sort() calls.

Approach 3:

Sorting and Two pointers approach

```
class Solution {
public:
    vector<vector<int>> fourSum(vector<int>& nums, int target) {
        vector<vector<int>> ans;
        sort(nums.begin(),nums.end());
        int n=nums.size();
        for(int i=0;i<=n-4;i++){
            if(i>0 && nums[i]==nums[i-1]) continue;
            for(int j=i+1;j<=n-3;j++){
                if(j>i+1 && nums[j]==nums[j-1]) continue;
                int l=j+1,r=n-1;
                long int val=(long int)target-(nums[i]+nums[j]);
                while(l<r){
                    if(nums[l]+nums[r] == val){
                        ans.push_back({nums[i],nums[j],nums[l++],nums[r--]});
                        while(l<r && nums[l]==nums[l-1]) l++;
                        while(l<r && nums[r]==nums[r+1]) r--;
                    }
                    else if(nums[l]+nums[r] < val) l++;
                    else r--;
                }
            }
        }

        return ans;
    }
};
```

let a,b,c,d

skipping duplicate a's

skipping duplicate b's

if we don't typecast, it gives overflow error for large values

skipping duplicate c's

skipping duplicate d's

$T(n): O(n \log n) + O(n^3)$

$S(n):$ no space