# 18. 4Sum

## SOLUTION IN JAVA

class Solution {

public List<List<Integer>> fourSum(int[] nums, int target) {

List<List<Integer>> ans = new ArrayList<>();

Arrays.sort(nums);

nSum(nums, 4, target, 0, nums.length - 1, new ArrayList<>(), ans);

return ans;

}

private void nSum(int[] nums, long n, long target, int l, int r, List<Integer> path,

List<List<Integer>> ans) {

if (r - l + 1 < n || target < nums[l] \* n || target > nums[r] \* n)

return;

if (n == 2) {

while (l < r) {

final int sum = nums[l] + nums[r];

if (sum == target) {

path.add(nums[l]);

path.add(nums[r]);

ans.add(new ArrayList<>(path));

path.remove(path.size() - 1);

path.remove(path.size() - 1);

++l;

--r;

while (l < r && nums[l] == nums[l - 1])

++l;

while (l < r && nums[r] == nums[r + 1])

--r;

} else if (sum < target) {

++l;

} else {

--r;

}

}

return;

}

for (int i = l; i <= r; ++i) {

if (i > l && nums[i] == nums[i - 1])

continue;

path.add(nums[i]);

nSum(nums, n - 1, target - nums[i], i + 1, r, path, ans);

path.remove(path.size() - 1);

}

}

}