/// <summary>

/// Leet code #18. 4Sum

/// Given an array S of n integers, are there elements a, b, c, and d in S

/// such that a + b + c + d = target?

/// Find all unique quadruplets in the array which gives the sum of target.

/// Note: The solution set must not contain duplicate quadruplets.

/// For example, given array S = [1, 0, -1, 0, -2, 2], and target = 0.

///

/// A solution set is:

/// [

/// [-1, 0, 0, 1],

/// [-2, -1, 1, 2],

/// [-2, 0, 0, 2]

/// ]

/// </summary>

vector<vector<int>> LeetCode::fourSum(vector<int>& nums, int target)

{

sort(nums.begin(), nums.end());

vector<vector<int>> result;

unordered\_map<int, vector<pair<int, int>>> two\_sum;

for (int i = nums.size() - 1; i > 2; i--)

{

if (4 \* nums[i] < target) break;

// dedup

if (i < nums.size() - 1 && nums[i] == nums[i + 1]) continue;

for (int j = i - 1; j > 1; j--)

{

// if (nums[i] + nums[j] \* 3 < target) break;

// dedup

if (j < (i - 1) && nums[j] == nums[j + 1]) continue;

two\_sum[nums[i] + nums[j]].push\_back({ j, i });

}

}

for (size\_t i = 0; i < nums.size() - 3; i++)

{

if (4 \* nums[i] > target) break;

// dedup

if (i > 0 && nums[i] == nums[i - 1]) continue;

for (size\_t j = i + 1; j < nums.size() - 2; j++)

{

// if (nums[i] + nums[j] \* 3 > target) break;

// dedup

if (j >(i + 1) && nums[j] == nums[j - 1]) continue;

int value = target - (nums[i] + nums[j]);

if (two\_sum.count(value) == 0) continue;

for (int k = two\_sum[value].size() - 1; k >= 0; k--)

{

if (two\_sum[value][k].first <= j) continue;

result.push\_back({ nums[i], nums[j], nums[two\_sum[value][k].first], nums[two\_sum[value][k].second] });

}

}

}

return result;

}