39 & 49 Combination Sum / II

Category	
Difficulty	
Note	
AC & Time	
Property	
related	

39. **Combination Sum**

Backtracking

```
vector<vector<int>> combinationSum(vector<int>& candidates, int target) {
        // clearfy the input , number range, order, how many time to use, type, sum range??

        vector<vector<int>> res;
        vector<int> path;
        sort(candidates.begin(), candidates.end());
        backtracking(candidates, target, 0, 0, path, res);
        return res;
    }

private:
    void backtracking(vector<int>& nums, int target, int sum, int indx, vector<int>& path, vector<vector<int>>& res) {
        if (sum == target) {
            res.push_back(path);
            return;
        }

        for (int i = indx; i < nums.size() && sum + nums[indx] <= target; i++) {
            sum += nums[i];
            path.push_back(nums[i]);
            backtracking(nums, target, sum, i, path, res);
            path.pop_back();
            sum -= nums[i];
        }
    }
O(n^2)
```

40. **Combination Sum II**

```
vector<vector<int>> combinationSum2(vector<int>& candidates, int target) {
        // use ones //deal with duplication;
        vector<vector<int>> res;
        vector<int> path;
        sort (candidates.begin(), candidates.end());//O(nlog(n))
        combinationSumHelper(candidates, target, 0, 0, path, res);
        return res;
    }


private:
    void combinationSumHelper(vector<int>& nums, int target, int sum, int startId, vector<int>& path, vector<vector<int>>& res) {
        if (sum == target) {
            res.push_back(path);
            return;
```

```
        }

        for (int i = startId; i < nums.size(); ++i) { //(n^2)
            if (sum + nums[i] > target) break;
            if (i && nums[i] == nums[i - 1] && i > startId) continue; // deal with dumplicaton
            sum += nums[i];
            path.push_back(nums[i]);
            combinationSumHelper(nums, target, sum, i + 1, path, res);
            path.pop_back();
            sum -= nums[i];

        }

    }
```