Greedy for practice:

Sharing good Greedy problems for practice:

Sort/Array
https://leetcode.com/problems/jump-game/
https://leetcode.com/problems/jump-game-ii/
https://leetcode.com/problems/gas-station/
https://leetcode.com/problems/candy/
https://leetcode.com/problems/remove-k-digits/
https://leetcode.com/problems/wiggle-subsequence/
https://leetcode.com/problems/assign-cookies/
https://leetcode.com/problems/boats-to-save-people/
https://leetcode.com/problems/bag-of-tokens/
https://leetcode.com/problems/number-of-burgers-with-no-waste-of-ingredients/
https://leetcode.com/problems/queue-reconstruction-by-height/
https://leetcode.com/problems/play-with-chips/
https://leetcode.com/problems/previous-permutation-with-one-swap/
https://leetcode.com/problems/lemonade-change/
https://leetcode.com/problems/bag-of-tokens/

Hash/Multi-set:
https://leetcode.com/problems/task-scheduler/
https://leetcode.com/problems/partition-labels/
https://leetcode.com/problems/car-pooling/
https://leetcode.com/problems/divide-array-in-sets-of-k-consecutive-numbers/
https://leetcode.com/problems/group-the-people-given-the-group-size-they-belong-to/
https://leetcode.com/problems/cinema-seat-allocation/
https://leetcode.com/problems/construct-k-palindrome-strings/
https://leetcode.com/problems/advantage-shuffle/

Strings:
https://leetcode.com/problems/reorganize-string/
https://leetcode.com/problems/string-without-aaa-or-bbb/

https://leetcode.com/problems/check-if-a-string-can-break-another-string/
https://leetcode.com/problems/remove-duplicate-letters/

Heap:
https://leetcode.com/problems/last-stone-weight/
https://leetcode.com/problems/reduce-array-size-to-the-half/

Stack:
https://leetcode.com/problems/minimum-add-to-make-parentheses-valid/

Sharing solutions for little tricky problems:

https://leetcode.com/problems/divide-array-in-sets-of-k-consecutive-numbers/

```
class Solution {

public:

	bool isPossibleDivide(vector<int>& nums, int k) {


		int n = nums.size();

		if (n % k != 0) return false;

		int ssize = n/k;


		map<int, int>hm;

		for (int i = 0; i < n; i++)

			hm[nums[i]]++;


		for (auto it = hm.begin(); it != hm.end(); it++) {

			if (hm[it->first] > 0) {

				for (int i = k-1; i >= 0; i--) {
```

```
                                hm[it->first+i] -= hm[it->first];
                                if (hm[it->first+i] < 0)
                                        return false;
                        }
                }
        }

        return true;
    }
};
```

https://leetcode.com/problems/car-pooling/

```
class Solution {
public:
    bool carPooling(vector<vector<int>>& trips, int capacity) {

        int trip_len = 1001;
        vector<int>stops(trip_len, 0);

        for (int i = 0; i < trips.size(); i++) {
            stops[trips[i][1]] += trips[i][0];
            stops[trips[i][2]] -= trips[i][0];
        }

        for (int i = 0; i < trip_len; i++) {
```

```
                        if (i != 0) stops[i] += stops[i-1];
                        if (stops[i] > capacity)
                                return false;
                }



                return true;
        }
};
```

https://leetcode.com/problems/reorganize-string/

```
class Solution {
        static bool compare(pair<char, int>p1, pair<char, int>p2) {
                return p1.second > p2.second;
        }
public:
        string reorganizeString(string S) {

                int n = S.length();

                unordered_map<char, int>m;
                vector<pair<char, int>>v;

                for (int i = 0; i < n; i++)
                        m[S[i]]++;
```

```
for(auto it = m.begin(); it != m.end(); it++) {
	if (it->second > (n+1)/2)
		return "";
	v.push_back(make_pair(it->first, it->second));
}

sort(v.begin(), v.end(), compare);
string str;
for (int i = 0; i < v.size(); i++) {
	while (v[i].second--)
		str += v[i].first;
}

string ans;
int size = str.size();
int i = 0, j = (size-1)/2+1;

while (i < (size-1)/2+1) {
	ans += str[i];
	ans += str[j];
	i++; j++;
}
```

```
            return ans;
        }
};
```

https://leetcode.com/problems/candy/

```
class Solution {
public:
        int candy(vector<int>& ratings) {

            int n = ratings.size();

            vector<int>left(n, 1);
            for (int i = 1; i < n; i++) {
                if (ratings[i] > ratings[i-1])
                    left[i] = left[i-1]+1;
            }

            int sum = left[n-1];
            for (int i = n-2; i >= 0; i--) {
                if (ratings[i] > ratings[i+1])
                    left[i] = max(left[i], left[i+1]+1);
                sum += left[i];
            }

            return sum;
```

```
        }
};
```