分治模板：

```
C/C++
int divide_conquer(Problem *problem, int params) {
  // recursion terminator
  if (problem == nullptr) {
    process_result
    return return_result;
  }

  // process current problem
   subproblems = split_problem(problem, data)
   subresult1 = divide_conquer(subproblem[0], p1)
   subresult2 = divide_conquer(subproblem[1], p1)
   subresult3 = divide_conquer(subproblem[2], p1)
   ...

   // merge
   result = process_result(subresult1, subresult2, subresult3)
   // revert the current level status

   return 0;
 }
```

分治作业题：

多数元素（众数）

```
class Solution {
 int count_in_range(vector<int>& nums, int target, int lo, int hi) {
 int count = 0;
 for (int i = lo; i <= hi; ++i)
 if (nums[i] == target)
 ++count;
 return count;
 }
 int majority_element_rec(vector<int>& nums, int lo, int hi) {
  if (lo == hi)
  return nums[lo];
  int mid = (lo + hi) / 2;
  int left_majority = majority_element_rec(nums, lo, mid); // 不断到子问
题，即每一层的数组的左半数组
```

```
        // 的左边的多数元素;
 int right_majority = majority_element_rec(nums, mid + 1, hi); // 同理,
每一层也去找右半数组的多数元素

 // 终止条件:
        // 上面两个递归肯定能找到众数，然后我们就根据这些数去判断是否他们是众数
 if (count_in_range(nums, left_majority, lo, hi) > (hi - lo + 1) / 2)
 return left_majority;
 if (count_in_range(nums, right_majority, lo, hi) > (hi - lo + 1) / 2)
 return right_majority;
 return -1;
 }
// 这个模板好像超哥讲的那个生成括号的题，虽然我还没写过那道题，但是感觉基本是一
个思路
public:
 int majorityElement(vector<int>& nums) {
 return majority_element_rec(nums, 0, nums.size() - 1);
 }
};
```

柠檬水找零：
这道题刚开始我在想能否用栈来解决，但是其实在题目给的这个例子中，自己不自觉就用了贪心的方式找零，即如果我们有3张5块，一张10块，然后当前顾客给了我们20时，我倾向于用10+5去换零，这就是贪心思维

```
class Solution {
public:
    bool lemonadeChange(vector<int>& bills) {
        int five = 0, ten = 0;
        for (auto bill : bills) {
            if (bill == 5) {
                five ++;
            }
            else if (bill == 10) {
                if (five == 0) return false;
                five --;
                ten ++;
             }
             else {
                if (five > 0 && ten > 0) {
                    five --;
                    ten --;
                }
                else if
                    (five >=3) five -= 3;
                else
                    return false;
            }
        }
        return true;
    }
};
```

分发饼干

```
class Solution {
public:
    int findContentChildren(vector<int>& g, vector<int>& s) {
        sort(g.begin(), g.end());
        sort(s.begin(), s.end());
        int res;
        int gi = 0, si = 0;
        while (gi < g.size() && si < s.size()) {
            if (g[gi] <= s[si]) {
                gi++;
            }
            si++;
        }
        return gi;
    }
};
```

买股票最佳时机：

又是贪心算法解——只在当前数-前一个数大于0的时候加

```
class Solution {
public:
    int maxProfit(vector<int>& prices) {
        int res = 0;
        for (int i = 1; i < prices.size(); i ++) {
            int tmp = prices[i] - prices[i - 1];
            if (tmp > 0) res += tmp;
        }
        return res;
    }
};
```

模拟行走机器人

```
class Solution {
public:
 int robotSim(vector<int>& commands, vector<vector<int>>& obstacles) {

 // map存储某个方向(key)对应的 {x方向移动，y方向移动，当前方向的左侧，当前方向的右侧} (val)
 unordered_map<string, vector<string>> mymap = {{"up", {"0", "1", "left", "right"}},{"down", {"0", "-1", "right", "left"}},{"left", {"-1", "0", "down", "up"}},{"right", {"1", "0", "up", "down"}}};

 unordered_set<string> obstacles_set;
 for(int i = 0; i < obstacles.size(); ++i){
  obstacles_set.insert(to_string(obstacles[i][0]) + " " + to_string(obstacles[i][1]));
  }

  int x = 0, y = 0;
  string dir = "up";
  int res = 0;

  for(auto command : commands){
  if(command > 0){
  for(int i = 0; i < command; ++i){
  int temp_x = x + stoi(mymap[dir][0]);
  int temp_y = y + stoi(mymap[dir][1]);

  string temp = to_string(temp_x) + " " + to_string(temp_y);

  if(obstacles_set.find(temp) == obstacles_set.end()){
  x += stoi(mymap[dir][0]);
  y += stoi(mymap[dir][1]);
  res = max(res, x * x + y * y);
  }
  else
  break;
  }
  }
  else{
  if(command == -2){
  dir = mymap[dir][2];
  }
  if(command == -1){
  dir = mymap[dir][3];
  }
  }
  }
  return res;
  }
  };.
```

二分做法：

```
class Solution:
 def robotSim(self, commands: List[int], obstacles: List[List[int]]) -> int:
 dirs = [(0, 1), (1, 0), (0, -1), (-1, 0)]
 cnt_dir = 0
 ans = 0
 x = y = 0

 ob_x_dict = {}
 ob_y_dict = {}
 for ob in obstacles:
 ob_x_dict.setdefault(ob[0], [])
 ob_x_dict[ob[0]].append(ob[1])
 ob_y_dict.setdefault(ob[1], [])
 ob_y_dict[ob[1]].append(ob[0])

 for v in ob_x_dict.values():
 v.sort()
 for v in ob_y_dict.values():
 v.sort()

 for cmd in commands:
 if cmd == -1:
 cnt_dir = (cnt_dir + 1) % 4
```

```
elif cmd == -2:
cnt_dir = (cnt_dir - 1 + 4) % 4
else:
new_x = dirs[cnt_dir][0] * cmd + x
new_y = dirs[cnt_dir][1] * cmd + y
if new_x != x and y in ob_y_dict:
idx = bisect.bisect_left(ob_y_dict[y], x)
if new_x > x:
if idx < len(ob_y_dict[y]) and x < ob_y_dict[y][idx] <= new_x:
new_x = ob_y_dict[y][idx] - 1
else:
if idx > 0:
idx -= 1
if new_x <= ob_y_dict[y][idx] < x:
new_x = ob_y_dict[y][idx] + 1
elif new_y != y and x in ob_x_dict:
idx = bisect.bisect_left(ob_x_dict[x], y)
if new_y > y:
if idx < len(ob_x_dict[x]) and y < ob_x_dict[x][idx] <= new_y:
new_y = ob_x_dict[x][idx] - 1
else:
if idx > 0:
idx -= 1
if new_y <= ob_x_dict[x][idx] < y:
new_y = ob_x_dict[x][idx] + 1

cnt_ans = new_x ** 2 + new_y ** 2
ans = max(ans, cnt_ans)
x, y = new_x, new_y

return ans
```

Pow(x, n)

```
class Solution {
public:

    double quickMul(double x, long long N) {
        if (N==0) {
            return 1.0;
        }
        double y = quickMul(x, N/2);
        return N % 2 == 0 ? y * y : y * y * x;
    }

    double myPow(double x, int n) {
        long long N = n;
        return N > 0 ? quickMul(x, N) : 1.0 / quickMul(x, -N);
    }
};
```