July 23, 2018 题目：543,729,321,363,317,735,105,90,255,617,746,209

543

1. Recursion:

```
class Solution {
    int res = 0;
    int depth(TreeNode* root){
        if(!root) return 0;
        int l = depth(root->left), r = depth(root->right);
        res = max(res, l+r);
        return 1 + max(l, r);
    }
public:
    int diameterOfBinaryTree(TreeNode* root) {
        depth(root);
        return res;
    }
};
```

2. 没你们厉害的 recursion:

```
class Solution {

    int hel = 1;

    public int helper(TreeNode root){
        if(root == null){
            return 0;
```

```
        }
        int left = helper(root.left);
        int right = helper(root.right);
        hel = Math.max(hel,1 + left + right);
        return Math.max(left+1,right+1);
    }

    public int diameterOfBinaryTree(TreeNode root) {
        helper(root);
        return hel - 1;

    }
}
```

3. 和1一样

```
class Solution {
    int ans = 0;
    public int diameterOfBinaryTree(TreeNode root) {
        if (root == null) return 0;
        length(root);
        return ans;
    }

    public int length(TreeNode root) {
        if (root == null) return 0;
        int left = length(root.left);
        int right = length(root.right);
        ans = Math.max(left + right, ans);
        return Math.max(left, right) + 1;
    }
```

729

1. Map maintain range:

```
class MyCalendar {
    map<int, int> range;
public:
    MyCalendar() {
        range[-2] = -1;
        range[int(1E9+7)] =int(1E9+9);
    }

    bool book(int start, int end) {
        auto it = range.upper_bound(start);
        if(end > it->first) return false;
        --it;
        if(it->second > start) return false;
        if(it->second == start) start = it->first;
        if(range.count(end)){
            int tmp = range[end];
            range.erase(end);
            end = tmp;
        }
        range[start] = end;
        return true;
    }
};
```

2. Python Brute

```
class MyCalendar:


    def __init__(self):
```

```
        self.calender = []

    def check_conflict(self,start,end):
        for date in self.calender:
            if start < date[1] and date[0] < end:
                return True
        return False

    def book(self, start, end):
        """
        :type start: int
        :type end: int
        :rtype: bool
        """
        if self.check_conflict(start,end):
            return False
        else:
            self.calender.append((start,end))
            return True
```

3. 同上

```
class MyCalendar {

    List<int[]> list;
    public MyCalendar() {
        list = new ArrayList<>();
    }

    public boolean book(int start, int end) {
        for (int[] l : list) {
            if (Math.max(l[0], start) < Math.min(l[1], end)) {
```

```
                return false;
            }
        }
        list.add(new int[]{start, end});
        return true;
    }
}
```

321

1.

```
class Solution {
    public int[] maxNumber(int[] nums1, int[] nums2, int k) {
        int[] res = new int[k];
    for (int i = Math.max(k - nums2.length, 0); i <= Math.min
(k, nums1.length); i++) {
      int[] res1 = new int[i];
      int[] res2 = new int[k - i];
      res1 = max(nums1, i);
      res2 = max(nums2, k - i);

      int[] tmp = new int[k];
      int p1 = 0, p2 = 0, p = 0;
      while (res1.length > 0 && res2.length > 0 && p1 < res1.l
ength && p2 < res2.length) {
        if (compare(res1, res2, p1, p2)) {
          tmp[p++] = res1[p1++];
        } else {
          tmp[p++] = res2[p2++];
        }
```

```
      }

      while (p1 < res1.length) {
        tmp[p++] = res1[p1++];
      }
      while (p2 < res2.length) {
        tmp[p++] = res2[p2++];
      }

      if (!compare(res, tmp, 0, 0)) {
        res = tmp;
      }
    }
    return res;
  }

  public int[] max(int[] nums, int k) {
    int[] res = new int[k];
    int len = 0;
    for (int i = 0; i < nums.length; i++) {
      while (len > 0 && nums.length - k > i - len && res[len -
1] < nums[i]) {
        len--;
      }
      if (len < k) res[len++] = nums[i];
    }
    return res;
  }
```

```
  public boolean compare(int[] nums1, int[] nums2, int p1, int
p2) {
    for (; p1 < nums1.length && p2 < nums2.length; p1++, p2++)
{
      if (nums1[p1] < nums2[p2]) return false;
      else if (nums1[p1] > nums2[p2]) return true;
    }
    return p1 != nums1.length;
  }
}
```

- [ ] Solve it @Zebo L

363

1. 想了好久好久的solution，参考了一下discussion修改了一下
    a. 我也是这个solution

```
class Solution {
    public int maxSumSubmatrix(int[][] matrix, int k) {
        int row = matrix.length;
        if (row == 0) return 0;
        int col = matrix[0].length;
        int m = Math.min(row, col);
        int n = Math.max(row, col);

        boolean flag = col > row;
        int res = Integer.MIN_VALUE;

        for (int i = 0; i < m; i++) {
            int[] tmp = new int[n];
            for (int j = i; j >= 0; j--) {
                int num = 0;
                TreeSet<Integer> set = new TreeSet<>();
```

```
                set.add(0);
                for (int index = 0; index < n; index++) {
                    tmp[index] += (flag ? matrix[j][index] : m
atrix[index][j]);
                    num += tmp[index];
                    Integer sub = set.ceiling(num - k);
                    if (sub != null) {
                        res = Math.max(res, num - sub);
                    }
                    set.add(num);
                }
            }
        }
        return res;
    }
}
```

2. 直接Brute Force 过了：

```
class Solution {
    typedef vector<int> vi;
public:
    int maxSumSubmatrix(vector<vector<int>>& matrix, int K) {
        int n = matrix.size(), m = matrix[0].size();
        vector<vi> sum(n+1, vi(m+1, 0));
        for(int i=1; i<=n; ++i) for(int j=1; j<=m; ++j){
            sum[i][j] = sum[i][j-1] + sum[i-1][j] - sum[i-1][j
-1] + matrix[i-1][j-1];
        }
        int ans = INT_MIN;
        for(int i=0; i<n; ++i) for(int j=i+1; j<=n;++j) for(in
t k=0; k<m; ++k) for(int l=k+1; l<=m; ++l){
```

```
            int area = sum[j][l] - sum[i][l] - sum[j][k] + sum
[i][k];
            if(area <= K) ans = max(ans, area);
        }
        return ans;
    }
};
```

- [x] ~~学习上面两位大神的解法~~ ~~@Zebo L~~
  - [x] ~~复杂度能降到~~ `O(nnm*log(m))` ~~, Do it later (Why so slow)~~
    - Note: `swap(m, n)` will make things much faster.

```
class Solution {
    typedef vector<int> vi;
public:
    int maxSumSubmatrix(vector<vector<int>>& matrix, int K) {
        int n = matrix.size(), m = matrix[0].size();
        vector<vi> sum(n+1, vi(m+1, 0));
        for(int i=1; i<=n; ++i) for(int j=1; j<=m; ++j){
            sum[i][j] = sum[i][j-1] + sum[i-1][j] - sum[i-1][j
-1] + matrix[i-1][j-1];
        }
        long ans = INT_MIN;
        for(int i=0; i<n; ++i) for(int j=i+1; j<=n;++j) {
            set<long> pool{long(INT_MIN), long(INT_MAX)};
            for(int k=m;k>=0;--k){
                int tmp = sum[j][k] - sum[i][k];
                if(pool.count(tmp+K)) return K;
                auto it = --pool.upper_bound(tmp+K);
                ans = max(ans, *it - tmp);
                pool.insert(tmp);
            }
        }
```

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

- 加一句，intuition 其实就是把问题压倒一个dimension 的array 然后再解决问题

317

1. Brute Force 可以过：注意要有个variable 标记哪些点是可以到达所有 Buildings 的

```
class Solution {
    int d[4] = {1, -1, 0, 0};
    typedef pair<int, int> ii;
public:
    int shortestDistance(vector<vector<int>>& grid) {
        int n = grid.size(), m = grid[0].size(), ans = INT_MI
N, cnt = 0;
        vector<vector<int>> pass(n, vector<int>(m, 0));
        for(int i=0; i<n; ++i) for(int j=0; j<m; ++j) if(grid
[i][j]==1){
            ++cnt;
            queue<ii> Q;
            unordered_set<int> S;
            Q.push(ii(i*m + j, 0));
            S.insert(i*m + j);
            while(!Q.empty()){
                int x = Q.front().first/m, y = Q.front().firs
t%m, step = Q.front().second;
                grid[x][y] -= step;
                Q.pop();
                for(int k=0; k<4; ++k){
                    int z = x + d[k], w = y + d[3-k];
                    if(z>=0 && z<n && w>=0 && w<m && grid[z]
[w]<=0 && !S.count(z*m + w)){
```

```
                        S.insert(z*m + w);
                        Q.push(ii(z*m+w, step+1));
                        ++pass[z][w];
                    }
                }
            }
        }
        for(int i=0; i<n; ++i) for(int j=0; j<m; ++j) if(grid
[i][j]<0 && pass[i][j]==cnt){
            ans = max(ans, grid[i][j]);
        }
        return (ans==INT_MIN? -1:-ans);
    }
};
```

- [ ] Looking at LeetCode discussion for faster solution @Zebo L
  - MS 大家都是这么做的

735

1. stack 不过写的很不clean

```
class Solution:
    def asteroidCollision(self, asteroids):
        stack = []
        size = 0
        for x in asteroids:
            if x > 0 or size == 0:
                stack.append(x)
                size+=1
            else:
                while size > 0 and abs(x) >= stack[-1]:
                    if stack[-1] < 0:
                        stack.append(x)
```

```
                        size+=1
                        break;
                    elif abs(x) == stack[-1]:
                        stack.pop()
                        size-=1
                        x = 0;
                        break;
                    else:
                        stack.pop()
                        size -= 1
                if x != 0 and size == 0:
                    size+=1
                    stack.append(x)

        return stack
```

2. 同上：

```
class Solution {
public:
    vector<int> asteroidCollision(vector<int>& asteroids) {
        vector<int> ans;
        for(int n: asteroids){
            if(n>0) ans.push_back(n);
            while(!ans.empty() && ans.back()>0 && ans.back()<-
n) ans.pop_back();
            if(ans.empty() || ans.back()<0) ans.push_back(n);
            else if(ans.back()==-n) ans.pop_back();
        }
        return ans;
    }
};
```

105

1. Slow as fuck... 有人能帮我加快吗？

```
def build(preorder, inorder):
    if preorder == [] or inorder == []:
        return None

    curr_val = preorder[0];

    k = 0;
    for x in range(len(inorder)):
        if inorder[x] == curr_val:
            k = x;
            break;

    tree = TreeNode(curr_val)
    tree.left = build(preorder[1:k+1],inorder[0:k]);
    tree.right = build(preorder[k+1:],inorder[k+1:]);

    return tree

class Solution:
    def buildTree(self, preorder, inorder):
        """
        :type preorder: List[int]
        :type inorder: List[int]
        :rtype: TreeNode
        """
        return build(preorder,inorder)
```

2. 上面的加速版 @Derek S
    a. 哦对直接存index少copy傻逼了 谢谢提醒 @Zebo L

```
class Solution:
    _pos = {}
    def _getTree(self, p, i, l1, r1, l2, r2):
        if l1 > r1:
            return None
        root = TreeNode(p[l1])
        k = self._pos[p[l1]]
        root.left = self._getTree(p, i, l1+1, l1+k-l2, l2, k-
1)
        root.right = self._getTree(p, i, l1+1+k-l2, r1, k+1, r
2)
        return root

    def buildTree(self, preorder, inorder):
        """
        :type preorder: List[int]
        :type inorder: List[int]
        :rtype: TreeNode
        """
        for i in range(len(inorder)):
            self._pos[inorder[i]] = i
        return self._getTree(preorder, inorder, 0, len(preorde
r)-1, 0, len(inorder)-1)
```

3. Recursion: 事先建一个map，这样就不用每次通过loop 找 root 了

```
#include<cassert>
class Solution {
    unordered_map<int, int> pos;
    TreeNode* bT(vector<int>& preorder, vector<int>& inorder,
int l1, int r1, int l2, int r2){
        if(l1 > r1) return NULL;
        TreeNode *root = new TreeNode(preorder[l1]);
```

```
        int k = pos[preorder[l1]];
        assert(k<=r2);
        root->left = bT(preorder, inorder, l1+1, l1+k-l2, l2, k-1);
        root->right = bT(preorder, inorder, l1+1+k-l2, r1, k+1, r2);
        return root;
    }
public:
    TreeNode* buildTree(vector<int>& preorder, vector<int>& inorder) {
        assert(preorder.size() == inorder.size());
        for(int i=0; i<inorder.size(); ++i) pos[inorder[i]] = i;
        return bT(preorder, inorder, 0, preorder.size()-1, 0, inorder.size()-1);
    }
};
```

90

1. DFS

```
class Solution {
    public List<List<Integer>> subsetsWithDup(int[] nums) {
        List<List<Integer>> res = new ArrayList<>();
        if (nums == null || nums.length == 0) {
            return res;
        }
        Arrays.sort(nums);
        dfs(nums, res, new ArrayList<Integer>(), 0);
        return res;
    }
```

```
    void dfs(int[] nums, List<List<Integer>> res, ArrayList<In
teger> list, int idx) {
        if (idx == nums.length) {
            res.add(new ArrayList<>(list));
            return;
        }

        list.add(nums[idx]);
        dfs(nums, res, list, idx + 1);
        list.remove(list.size() - 1);

        int temp = idx;
        while(idx < nums.length && nums[idx] == nums[temp] ) {
            idx++;
        }
        dfs(nums, res, list, idx);
    }
}
```

2. 同上：

```
class Solution {
    typedef vector<int> vi;
    map<int, int> cnt;
    void dfs(vector<vi> &ans, vi cur, map<int, int>::iterator
it, vi &nums){
        if(it == cnt.end()){
            ans.push_back(cur);
            return;
        }
        int n = it->second, k = it->first;
        ++it;
        dfs(ans, cur, it, nums);
```

```
        for(int i=0; i<n; ++i){
            cur.push_back(k);
            dfs(ans, cur, it, nums);
        }
    }
public:
    vector<vector<int>> subsetsWithDup(vector<int>& nums) {
        vector<vi> ans;
        for(int k: nums) ++cnt[k];
        dfs(ans, vi(), cnt.begin(), nums);
        return ans;
    }
};
```

255

1. 奇技淫巧，find out the lower and upper bound by using a stack, or the previous track, like a 单调递增栈
2. 上面的实现，注意边界别overflow

```
class Solution {
public:
    bool verifyPreorder(vector<int>& preorder) {
        stack<int> S;
        long lower_bound = long(INT_MIN)-1;
        for(auto n: preorder){
            if(long(n) <= lower_bound) return false;
            while(!S.empty() && n>S.top()) {
                lower_bound = S.top();
                S.pop();
            }
            S.push(n);
        }
```

```
        return true;
    }
};
```

617

1.

```
class Solution {
    public TreeNode mergeTrees(TreeNode t1, TreeNode t2) {
        return merge(t1, t2);
    }

    public TreeNode merge(TreeNode t1, TreeNode t2) {
        if (t1 == null && t2 == null) {
            return null;
        }
        int val = 0;
        if (t1 != null) val += t1.val;
        if (t2 != null) val += t2.val;

        TreeNode node = new TreeNode(val);
        node.left = merge(t1 == null ? null : t1.left, t2 == n
ull ? null : t2.left);
        node.right = merge(t1 == null ? null : t1.right, t2 ==
null ? null : t2.right);
        return node;

    }
}
```

2. 和上面一样

```
class Solution {
    public TreeNode mergeTrees(TreeNode t1, TreeNode t2) {
```

```
        if(t1 == null && t2 == null){
            return null;
        }
        if(t1 == null){
            return t2;
        }
        if(t2 == null){
            return t1;
        }
        TreeNode left = mergeTrees(t1.left,t2.left);
        t1.val = t1.val + t2.val;
        t1.left = left;
        t1.right = mergeTrees(t1.right,t2.right);
        return t1;
    }
}
```

3. 同上recursion，但注意这个解法虽然简洁，但是用了已有`nodes`，面试的时候最好make copy:

```
class Solution {
public:
    TreeNode* mergeTrees(TreeNode* t1, TreeNode* t2) {
        if(!t1 && !t2) return NULL;
        if(!t1) return t2;
        if(!t2) return t1;
        TreeNode *root = new TreeNode(t1->val + t2->val);
        root->left = mergeTrees(t1->left, t2->left);
        root->right = mergeTrees(t1->right, t2->right);
        return root;
    }
};
```

746

1. DP

```
class Solution {
    public int minCostClimbingStairs(int[] cost) {
        int[] dp = new int[cost.length];
        dp[0] = 0;
        dp[1] = Math.min(cost[0],cost[1]);

        for(int x = 2; x < cost.length; x ++){
            dp[x] = Math.min(cost[x] + dp[x-1],dp[x-2] + cost
[x-1]);
        }
        return dp[dp.length - 1];
    }
}
```

2. 跟楼上基本相同，但：`O(1) space`

```
class Solution {
public:
    int minCostClimbingStairs(vector<int>& cost) {
        int a = 0, b = 0;
        for(int i=2; i<=cost.size(); ++i){
            int c = min(a + cost[i-2], b + cost[i-1]);
            a = b;
            b = c;
        }
        return b;
    }
};
```

209

1. sliding window

```
class Solution {

    public int minSubArrayLen(int s, int[] nums) {

        int sum = 0;
        int min_count = nums.length + 1;
        int j = 0;
        for(int i = 0; i < nums.length; i++){
            sum = sum + nums[i];
            while(sum >= s){
                min_count = Math.min(min_count,i+1 - j);
                sum = sum - nums[j];
                j++;
            }
        }
        min_count = (min_count == nums.length + 1) ? 0 : min_c
ount;
        return min_count;

    }
}
```

2. 同上

```
class Solution {
public:
    int minSubArrayLen(int s, vector<int>& nums) {
        if(!s) return 0;
        int i = 0, j = 0, sum = 0, n = nums.size(), ans = num
s.size();
        while(i<n && j<n){
            while(j<n && sum<s) sum += nums[j++];
```

```
            if(i==0 && j==n && sum<s) return 0;
            while(i<n && sum-nums[i]>=s) sum -= nums[i++];
            ans = min(ans, j-i);
            sum -= nums[i++];
        }
        return ans;
    }
};
```