August 19, 2018 题目：790,23,463,422,719,709,93,430,318,303,216,300,390,39

790

1. Easy dp (Can be `O(1)` space)

```
class Solution {
    typedef vector<long> vl;
    #define M (1000000007)
public:
    int numTilings(int N) {
        vector<vl> dp(N+1, vl(3, 0L));
        dp[0][0] = dp[1][0] = dp[1][1] = dp[1][2] = 1L;
        for(int j=2; j<=N; ++j){
            dp[j][0] = (dp[j-1][0] + dp[j-2][0] + dp[j-2][1] +
dp[j-2][2])%M;
            dp[j][1] = (dp[j-1][0] + dp[j-1][2]) %M;
            dp[j][2] = (dp[j-1][0] + dp[j-1][1]) %M;
        }
        return dp[N][0];
    }
};
```

2. Java version

```
 class Solution {
    final int number = 1000000007;
    public int numTilings(int N) {
        if (N == 0) return 0;
        if (N < 4) {
            switch(N) {
                case 0 : return 1;
                case 1 : return 1;
                case 2 : return 2;
                case 3 : return 5;
            }
        }
        int[] dp = new int[N + 1];
        dp[0] = 1;
        dp[1] = 1;
        dp[2] = 2;
        dp[3] = 5;
        for (int i = 4; i <= N; i++) {
            dp[i] = ((2 * dp[i - 1]) % number + dp[i - 3]) % n
umber;
        }
        return dp[N];
    }
}
```

3. 同上

```
class Solution {
    public int numTilings(int N) {
        if (N == 0) return 0;
        else if (N == 1) return 1;
        else if (N == 2) return 2;
        else if (N == 3) return 5;
        int[] dp = new int[N + 1];
        int number = 1000000007;
        dp[0] = 1;
        dp[1] = 1;
        dp[2] = 2;
        dp[3] = 5;
        for (int i = 4; i <= N; i++) {
            dp[i] = (((dp[i - 1] * 2) % number) + dp[i - 3]) %
number;
        }
        return dp[N];

    }
}
```

23

1. 典型merge SORT：

```
class Solution {
    ListNode* mergeTwo(ListNode *l1, ListNode *l2){
        ListNode lead(0);
        for(auto p=&lead; l1||l2; p=p->next){
            if(!l2) {
                p->next = l1;
                l1 = l1->next;
            }
            else if(!l1){
                p->next = l2;
                l2 = l2->next;
            }
            else if(l1->val < l2->val){
                p->next = l1;
                l1 = l1->next;
            }
            else{
                p->next = l2;
                l2 = l2->next;
            }
        }
        return lead.next;
    }
public:
    ListNode* mergeKLists(vector<ListNode*>& lists) {
        if(lists.empty()) return NULL;
        int N = lists.size();
        while(N>1){
            for(int i=0; i<N/2; ++i) lists[i] = mergeTwo(lists
[i], lists[N-i-1]);
            N = (N+1)/2;
            lists.resize(N);
        }
        return lists[0];
    }
};
```

463

1. 一个一个算就行了：

```
class Solution {
    int d[4] = {1, -1, 0, 0};
public:
    int islandPerimeter(vector<vector<int>>& grid) {
        if(grid.empty() || grid[0].empty()) return 0;
        int n = grid.size(), m = grid[0].size(), ans=0;
        for(int i=0; i<n; ++i) for(int j=0; j<m; ++j) if(grid
[i][j]) for(int k=0; k<4; ++k) {
            int i1 = i + d[k], j1 = j + d[3-k];
            if(i1<0 || i1>=n || j1<0 || j1>=m || !grid[i1][j
1]) ++ans;
        }
        return ans;
    }
};
```

422

1. 刚做过：

```
class Solution {
public:
    bool validWordSquare(vector<string>& W) {
        if(W.empty() || W[0].empty()) return false;
        int n = W.size();
        for(int i=0; i<n; ++i) {
            int m = W[i].size();
            if(m > n) return false;
            for(int j=0; j<m; ++j){
                if(W[j].size()<= i || W[i][j] != W[j][i]) retu
rn false;
            }
        }
        return true;
    }
};
```

719

1. 二分：给定m，计算差值比m小的个数。设 answer 为 `ans`, 则差值比 `ans` 小的pair的个数 `<k`, 差值比`ans+1` 小的pair个数 `>=k`

```
class Solution {
    typedef vector<int> vi;
    int smallerPair(vi &A, int m){
        int n = A.size(), cnt = 0;
        for(int i=0; i<n-1; ++i){
            if(A[i+1] - A[i] >= m) continue;
            int l = i+1, r = n;
            while(l<r-1){
                int c = (l+r)/2;
                if(A[c] - A[i] >= m) r = c;
                else l = c;
            }
            cnt += l-i;
        }
        return cnt;
    }
public:
    int smallestDistancePair(vector<int>& nums, int k) {
        int n = nums.size();
        sort(nums.begin(), nums.end());
        int l = -1, r = nums[n-1] - nums[0] + 1;
        while(l<r-1){
            int c = (l+r)/2;
            if(smallerPair(nums, c) < k) l = c;
            else r = c;
        }
        return l;
    }
};
```

709

1.

```
class Solution {
    public String toLowerCase(String str) {
        StringBuilder sb = new StringBuilder();
        for (char c : str.toCharArray()) {
            char t = (c - 'A') < 26 && (c - 'A') >= 0 ? (char)
('a' + (c - 'A')) : c;
            sb.append(t);
        }
        return sb.toString();
    }
}
```

2. Short is Beauty 系列：

```
class Solution(object):
    def toLowerCase(self, str):
        return str.lower()
```

93

1. DFS:

```
class Solution {
    #define CI(c) int((c) - '0')
    typedef vector<string> vs;
    void dfs(vs &ans, string cur, int i, int k, string& s){
        if(k==4 || i==s.size()){
            if(i == s.size() && k==4){
                cur.pop_back();
                ans.push_back(cur);
            }
            return;
        }
        if(s[i] == '0'){
            dfs(ans, cur+"0.", i+1, k+1, s);
        }
        else{
            int x = 0, j = i;
            while(j<s.size() && (x = x*10 + CI(s[j])) <= 255){
                dfs(ans, cur+to_string(x) + ".", j+1, k+1, s);
                ++j;
            }
        }
    }
public:
    vector<string> restoreIpAddresses(string s) {
        vs ans;
        dfs(ans, "", 0, 0, s);
        return ans;
    }
};
```

430

1. recrusion

```
class Solution {
    Node pre;
    public Node flatten(Node head) {
        Node node = head;
        while (node != null) {
            if (node.child != null) {
                Node next = node.next;
                node.child.prev = node;
                node.next = flatten(node.child);
                node.child = null;
                if (next != null) {
                    pre.next = next;
                    next.prev = pre;
                }
            }
            pre = node;
            node = node.next;
        }
        return head;
    }
}
```
2. 跟 BT 的 preorder traversal 一摸一样，可以用stack，记得改指针就行了：
```
class Solution {
public:
    Node* flatten(Node* head) {
        Node *p = head, *pre = NULL;
        stack<Node *> S;
        while(p||!S.empty()){
            while(p){
                p->prev = pre;
                if(pre) pre->next = p;
                if(p->next) S.push(p->next);
                pre = p;
                p = p->child;
                pre->child = NULL;
            }
            if(!S.empty()){
                p = S.top();
                S.pop();
            }
        }
        return head;
    }
};
```

318

1. bit

```
class Solution {
    public int maxProduct(String[] words) {
        Map<String, Integer> map = new HashMap<>();
        for (String word : words) {
            int k = 0;
            for (char c : word.toCharArray()) {
                k |= 1 << (c - 'a');
            }
            map.put(word, k);
        }
        int res = 0;
        for (int i = 0; i < words.length; i++) {
            int k = map.get(words[i]);
            for (int j = i + 1; j < words.length; j++) {
                if ((k & map.get(words[j])) == 0) {
                    res = Math.max(res, words[i].length() * wo
rds[j].length());
                }
            }
        }
        return res;
    }
}
```

2. 同上：

```
class Solution {
    #define CI(c) int((c) - 'a')
public:
    int maxProduct(vector<string>& words) {
        unordered_map<int, int> L;
        for(string &s: words){
            int tmp = 0;
            for(char &c: s) tmp |= (1<<CI(c));
            L[tmp] = max(L[tmp], (int)s.size());
        }
        int ans = 0;
        for(auto it=L.begin(); it!=L.end(); ++it) for(auto ir=
it; ir!=L.end(); ++ir) if(!(it->first & ir->first)){
            ans = max(ans, it->second * ir->second);
        }
        return ans;
    }
};
```

303

1. 毕竟Easy:

```
class NumArray {
    vector<int> P;
public:
    NumArray(vector<int> nums): P(vector<int>(nums.size() + 1,
0)) {
        for(int i=1; i<=nums.size(); ++i) P[i] = P[i-1] + nums
[i-1];
    }
    int sumRange(int i, int j) { return P[j+1] - P[i]; }
};
```

216

1. backtracking

```
class Solution {
    public List<List<Integer>> combinationSum3(int k, int n) {
        List<List<Integer>> res = new ArrayList<>();
        helper(res, new ArrayList<>(), k, n, 1);
        return res;
    }

    private void helper(List<List<Integer>> res, List<Integer> 
list, int k, int n, int pos) {
        if (list.size() == k && n == 0) {
            res.add(new ArrayList<>(list));
            return;
        }

        for (int i = pos; i <= 9; i++) {
            list.add(i);
            helper(res, list, k, n - i, i + 1);
            list.remove(list.size() - 1);
        }
    }
}
```

2. 这个是带重复的：

```
class Solution {
    typedef vector<int> vi;
    void dfs(vector<vi> &ans, vi cur, int i, int j, int sum, i
nt&k, int&n){
        int x = n - sum, y = k - j;
        if(!x){
            ans.push_back(cur);
            return;
        }
        if(i == 9){
            if(y*i == x){
                for(int k=0; k<y; ++k) cur.push_back(i);
                ans.push_back(cur);
            }
            return;
        }
        if(9 * y < x) return;
        int l = max(0, y*i+y-x), r = min(x/i, min((9*y-x)/(9-
i), y));
        if(r < l) return;
        for(int z=0; z<l; ++z){
            cur.push_back(i);
            ++j;
            sum += i;
        }
        for(int z=l; z<=r; ++z){
            dfs(ans, cur, i+1, j, sum, k, n);
            cur.push_back(i);
            ++j;
            sum += i;
        }
    }
public:
    vector<vector<int>> combinationSum3(int k, int n) {
        vector<vi> ans;
        dfs(ans, vi(), 1, 0, 0, k, n);
        return ans;
    }
};
```
3. 同1:
```
class Solution {
    typedef vector<int> vi;
    void dfs(vector<vi> &ans, vi cur, int i, int x, int y){
        if(!y || i==10 || i>x) return;
        cur.push_back(i);
        x -= i;
        --y;
        if(!x){
            if(!y) ans.push_back(cur);
            return;
        }
        for(int k=i+1; k<10; ++k) dfs(ans, cur, k, x, y);
    }
public:
    vector<vector<int>> combinationSum3(int k, int n) {
        vector<vi> ans;
        for(int i=1; i<10; ++i) dfs(ans, vi(), i, n, k);
        return ans;
    }
};
```

300

1. LIS
```
class Solution {
    public int lengthOfLIS(int[] nums) {
        List<Integer> list = new ArrayList<>();
        if (nums == null || nums.length == 0) return 0;
        list.add(nums[0]);
        for (int num : nums) {
            int k = list.get(list.size() - 1);
            if (num > k) {
                list.add(num);
            } else {
                int n = binarySearch(list, num);
                list.set(n, num);
            }
        }
        return list.size();
    }

    public int binarySearch(List<Integer> list, int target) {
        int i = 0, j = list.size() - 1;
        while (i < j) {
            int mid = i + (j - i) / 2;
            if (list.get(mid) < target) {
                i = mid + 1;
            } else {
                j = mid;
            }
        }
        return i;
    }
}
```

2. LIS :

```
class Solution {
public:
    int lengthOfLIS(vector<int>& nums) {
        if(nums.empty()) return 0;
        vector<int> head{nums[0]};
        for(int i=1; i<nums.size(); ++i){
            if(nums[i] > head.back()) head.push_back(nums[i]);
            else{
                int l = -1, r = head.size()-1;
                while(l<r-1){
                    int c = (l+r)/2;
                    if(head[c] < nums[i]) l = c;
                    else r = c;
                }
                head[r] = nums[i];
            }
        }
        return head.size();
    }
};
```

390

1.

```
class Solution {
    public int lastRemaining(int n) {
        int step = 1;
        int k = 1;
        boolean left = true;

        while (n > 1) {
            if (left || n % 2 == 1) {
                k += step;
            }
            n = n / 2;
            step = step * 2;
            left = !left;
        }
        return k;
    }
}
```
2. 逆向变换：看经过一次筛选后，下一轮第`i`个元素，对应原来序列的那一个即可:
```
class Solution {
public:
    int lastRemaining(int n) {
        stack<int> S;
        while(n>1){
            S.push(n);
            n /= 2;
        }
        int k = 1;
        while(!S.empty()){
            k = (S.top()/2)*2 - 2*(k-1);
            S.pop();
        }
        return k;
    }
};
```
- [ ] 把 1 弄懂 @Zebo L

39

1. DFS: Why only beat `35%`
```
class Solution {
    typedef vector<int> vi;
    void dfs(vector<vi> &ans, vi cur, int x, vi&C, int i){
        if(!x || i==C.size()){
            if(!x) ans.push_back(cur);
            return;
        }
        do{
            dfs(ans, cur, x, C, i+1);
            cur.push_back(C[i]);
            x -= C[i];
        }while(x>=0);
    }
public:
    vector<vector<int>> combinationSum(vector<int>& candidate
s, int target) {
        sort(candidates.begin(), candidates.end(), greater<int
>());
        vector<vi> ans;
        dfs(ans, vi(), target, candidates, 0);
        return ans;
    }
};
```

2. 多向别人学back tracking，自己总写不好:

```
class Solution {
    typedef vector<int> vi;
    void dfs(vector<vi> &ans, vi cur, int x, vi&C, int i){
        if(!x){
            ans.push_back(cur);
            return;
        }
        for(int j=i; j<C.size() && x>=C[j]; ++j){
            cur.push_back(C[j]);
            dfs(ans, cur, x-C[j], C, j);
            cur.pop_back();
        }
    }
public:
    vector<vector<int>> combinationSum(vector<int>& candidate
s, int target) {
        vector<vi> ans;
        sort(candidates.begin(), candidates.end());
        dfs(ans, vi(), target, candidates, 0);
        return ans;
    }
};
```

- [ ] 好好研究一下 Back Track @Zebo L