July 18, 2018 题目：4,10,864,859,857,188,552,161,144,824,457

4

1. classic problem. Shall use the idea of kth largest element. Watch out for the index moving issue. Use Kth element instead of the index to avoid confusion.

```
class Solution {
    public double findMedianSortedArrays(int[] nums1, int[] nums2) {
        int len1 = nums1.length;
        int len2 = nums2.length;
        int left = (len1 + len2 + 1) / 2;
        int right = (len1 + len2 + 2) / 2;
        return (find(nums1, nums2, left) + find(nums1, nums2, right)) / 2.0;
    }
    int find(int[] nums1, int[] nums2, int k) {
        int len1 = nums1.length;
        int len2 = nums2.length;
        if (len2 == 0) {
            return nums1[k - 1];
        }
        if (len1 == 0) {
            return nums2[k - 1];
        }
        if (k == 1) {
            return Math.min(nums1[0], nums2[0]);
        }
```

```
        int i = Math.min(len1, k / 2);
        int j = Math.min(len2, k / 2);
        if (nums1[i - 1] < nums2[j - 1]) {
            return find(Arrays.copyOfRange(nums1, i, len1), nu
ms2, k - i);
        }
        return find(nums1, Arrays.copyOfRange(nums2, j, len2),
k - j);
    }
}
```

2. Function `getKthNumber` cannot be avoided. (一直比较 `A[i]` and `B[k-i]`):

```
class Solution {
    typedef vector<int> vi;
    int findK(vi &A, vi&B, int k){
        int n = A.size(), m = B.size();
        int l = max(0, k-m), r = k;
        if(!n || (k<m && A[0]>=B[k])) return B[k];
        if(!m || (k<n && A[k]<=B[0])) return A[k];
        while(l < r - 1){
            int c = (l + r)/2;
            if(A[c] < B[k-c]) l = c;
            else r = c;
        }
        return max(A[l], B[k-l-1]);
    }
public:
    double findMedianSortedArrays(vector<int>& nums1, vector<i
nt>& nums2) {
        if(nums1.size() < nums2.size()) swap(nums1, nums2);
        int n1 = nums1.size(), n2 = nums2.size();
```

```
        return 0.5 * double(findK(nums1, nums2, (n1+n2-1)/2) +
findK(nums1, nums2, (n1+n2)/2));
    }
};
```

3.

```
public double findMedianSortedArrays(int[] nums1, int[] nums2)
{
        int m = nums1.length, n = nums2.length;
        if ((m + n) % 2 == 0) {
            return (double) (helper(nums1, nums2, 0, 0, (m +
n) / 2 + 1) + helper(nums1, nums2, 0, 0, (m + n) / 2)) / 2;
        }
        return helper(nums1, nums2, 0, 0, (m + n) / 2 + 1);
    }

    private int helper(int[] nums1, int[] nums2, int i, int j,
int k) {
        if (i >= nums1.length) return nums2[j + k - 1];
        if (j >= nums2.length) return nums1[i + k - 1];

        if (k == 1) {
            return Math.min(nums1[i], nums2[j]);
        }

        int a = i + k / 2 - 1 >= nums1.length ? Integer.MAX_VA
LUE : nums1[i + k / 2 - 1];
        int b = j + k / 2 - 1 >= nums2.length ? Integer.MAX_VA
LUE : nums2[j + k / 2 - 1];

        if (a > b) {
            return helper(nums1, nums2, i, j + k / 2, k - k /
2);
```

```
        } else {
            return helper(nums1, nums2, i + k / 2, j, k - k / 2);
        }
    }
```

- [ ] See LeetCode Discussion @Zebo L

10

1. 经典2D DP

```
public boolean isMatch(String s, String p) {
        // if p is null, return false;
        if (p == null || s == null) return false;
        boolean[][] dp = new boolean[s.length() + 1][p.length() + 1];
        dp[0][0] = true;
        for (int i = 0; i < p.length(); i++) {
            if (p.charAt(i) == '*') {
                dp[0][i + 1] = dp[0][i - 1];
            }
        }
        for (int i = 0; i < s.length(); i++) {
            for (int j = 0; j < p.length(); j++) {
                if (s.charAt(i) == p.charAt(j) || p.charAt(j) == '.') {
                    dp[i + 1][j + 1] = dp[i][j];
                }
                if (p.charAt(j) == '*') {
                    // a* count for empty.
                    if (p.charAt(j - 1) != s.charAt(i) && p.charAt(j - 1) != '.') {
                        dp[i + 1][j + 1] = dp[i + 1][j - 1];
```

```
                    } else {
                        dp[i + 1][j + 1] = dp[i][j + 1] || dp
[i + 1][j] || dp[i + 1][j - 1];
                    }
                }
            }
        }
        return dp[s.length()][p.length()];
    }
```

2. 小细节特别容易错

```
class Solution {
    public boolean isMatch(String s, String p) {
        if (s == null || p == null) {
            return false;
        }
        boolean[][] dp = new boolean[s.length() + 1][p.length
() + 1];
        char[] sc = s.toCharArray();
        char[] pc = p.toCharArray();
        dp[0][0] = true;
        for (int i = 1; i <= p.length(); i++) {
            if (pc[i - 1] == '*') {
                dp[0][i] = dp[0][i - 2];
            }
        }

        for (int i = 1; i < sc.length + 1; i++) {
            for (int j = 1; j < pc.length + 1; j++) {
                if (sc[i - 1] == pc[j - 1] || pc[j - 1] ==
'.') {
                    dp[i][j] = dp[i - 1][j - 1];
```

```
                } else if (pc[j - 1] == '*') {
                        if (pc[j - 2] == sc[i - 1] || pc[j - 2] == '.') {
                            dp[i][j] = dp[i][j - 2] || dp[i - 1][j];
                        } else {
                            dp[i][j] = dp[i][j - 2];
                        }
                }
            }
        }
        return dp[s.length()][p.length()];
    }
}
```

3. details

```
class Solution {
public:
    bool isMatch(string s, string p) {
        int n = s.size(), m = p.size();
        vector<vector<bool>> dp(s.size()+1, vector<bool>(p.size() + 1, false));
        dp[n][m] = true;
        for(int i=n; i>=0; --i) for(int j=m-1; j>=0; --j){
            if(i==n) {
                if(j<m-1 && p[j+1] == '*') dp[i][j] = dp[i][j+2];
                else dp[i][j] = false;
            }
            else{
                if(p[j] == '*') dp[i][j] = false;
                if(p[j] == '.') {
```

```
                    if(j<m-1 && p[j+1] == '*'){
                        dp[i][j] = dp[i][j+2];
                        for(int k=i; k<n && !dp[i][j]; ++k) dp
[i][j] = dp[k+1][j+2];
                    }
                    else dp[i][j] = dp[i+1][j+1];
                }
                else {
                    if(j<m-1 && p[j+1] == '*'){
                        dp[i][j] = dp[i][j+2];
                        for(int k=i; k<n && s[k]==p[j] && !dp
[i][j]; ++k) dp[i][j] = dp[k+1][j+2];
                    }
                    else dp[i][j] = (p[j] == s[i]) && dp[i+1]
[j+1];
                }
            }
        }
        return dp[0][0];
    }
};
```

See LeetCode Discussion @Zebo L

864

1. BFS with bit mask

```
class Solution {
    class Status {
        int x;
        int y;
        int keys;
        public Status (int x, int y, int keys) {
```

```
            this.x = x;
            this.y = y;
            this.keys = keys;
        }
    }

    public int shortestPathAllKeys(String[] grid) {
        int m = grid[0].length(), n = grid.length;
        int startX = -1, startY = -1;
        int steps = 0;
        int maxKey = 0;
        int[] dx = {1, -1, 0, 0};
        int[] dy = {0, 0, 1, -1};
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                char c = grid[j].charAt(i);
                if (c == '@') {
                    startX = i;
                    startY = j;
                }
                if (c >= 'a' && c <= 'f') {
                    maxKey = Math.max(maxKey, c - 'a' + 1);
                }
            }
        }
        Queue<Status> q = new LinkedList<>();
        q.offer(new Status(startX, startY, 0));
        Set<String> visited = new HashSet<>();
        visited.add(parse(startX, startY, 0));
        while (!q.isEmpty()) {
```

```
            int size = q.size();
            for (int i = 0; i < size; i++) {
                Status cur = q.poll();
                int x = cur.x;
                int y = cur.y;
                if (cur.keys == ((1 << maxKey) - 1)) return st
eps;

                for (int k = 0; k < 4; k++) {
                    int keys = cur.keys;
                    int ux = dx[k] + x;
                    int uy = dy[k] + y;
                    if (ux < 0 || uy < 0 || ux >= m || uy >=
n) continue;
                    char c = grid[uy].charAt(ux);
                    if (c == '#') continue;
                    if (c >= 'a' && c <= 'f') keys |= 1 << (c
- 'a');
                    if (c >= 'A' && c <= 'F' && ((keys >> (c -
'A')) & 1) == 0) continue;
                    if (!visited.contains(parse(ux, uy, key
s))) {
                        visited.add(parse(ux, uy, keys));
                        q.offer(new Status(ux, uy, keys));
                    }

                }
            }
            steps++;
        }
        return -1;
```

```
    }
    private String parse(int i, int j, int keys) {
        return i + "," + j + "," + keys;
    }
}
```

2. 世界真奇妙！！！
3. BFS:

```
class Solution {
    typedef pair<int, int> ii;
public:
    int shortestPathAllKeys(vector<string>& grid) {
        int n = grid.size(), m = grid[0].size(), fin = 0, d[4]
={1,-1,0,0};
        queue<ii> Q;
        unordered_set<int> S;
        fin = 0;
        for(int i=0;i<n;++i) for(int j=0;j<m;++j){
            if(grid[i][j] == '@'){
                Q.push(ii(32*i + j, 0));
                S.insert(32*i + j);
            }
            else if(grid[i][j] >= 'a' && grid[i][j] <= 'f'){
                int k = int(grid[i][j] - 'a');
                fin |= (1<<k);
            }
        }
        while(!Q.empty()){
            int M = Q.front().first, step = Q.front().second;
            Q.pop();
            int stat = M/1024, i = (M%1024)/32, j = (M%1024)%3
2;
```

```
            if(stat == fin) return step;
            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]
[j1]=='#') continue;
                if(grid[i1][j1] == '.' || grid[i1][j1] == '@')
{
                    int new_stat = i1*32 + j1 + 1024 * stat;
                    if(!S.count(new_stat)){
                        S.insert(new_stat);
                        Q.push(ii(new_stat, step+1));
                    }
                }
                else if(grid[i1][j1]>='a' && grid[i1][j1]<
='f'){
                    int new_stat = i1*32 + j1, r = int(grid[i
1][j1] - 'a');
                    new_stat += 1024 * (stat|(1<<r));
                    if(!S.count(new_stat)){
                        S.insert(new_stat);
                        Q.push(ii(new_stat, step+1));
                    }
                }
                else{
                    int new_stat = i1*32 + j1 + 1024 * stat, r
= int(grid[i1][j1] - 'A');
                    if((1&(stat>>r)) && !S.count(new_stat)){
                        S.insert(new_stat);
                        Q.push(ii(new_stat, step+1));
                    }
                }
```

```
            }
        }
        return -1;
    }
};
```

- [ ] See LeetCode Discussion @Zebo L

859

1. 本来用两个for loop但是会超时。需要先考虑两个string相同的情况。

```
public boolean buddyStrings(String A, String B) {
        if (A.length() != B.length()) return false;

        if (A.equals(B)) {
            Set<Character> set = new HashSet<>();
            for (char c : A.toCharArray()) set.add(c);
            return set.size() < A.length();
        }
        List<Integer> list = new ArrayList<>();
        for (int i = 0; i < A.length(); i++) {
            if (A.charAt(i) != B.charAt(i)) list.add(i);
        }
        return list.size() == 2 && A.charAt(list.get(0)) == B.
charAt(list.get(1)) && A.charAt(list.get(1)) == B.charAt(list.
get(0));
    }
```

2. Be careful about several cases, 1) when 2 are actually equal, look for same char, 2) when not equal, look for two indices.
3. 计数问题：

```
class Solution {
public:
    bool buddyStrings(string A, string B) {
        if(A.size() != B.size()) return false;
```

```
        bool same = false;
        map<char, int> cnt;
        for(auto c: A) {
            ++cnt[c];
            if(cnt[c]>=2) same = true;
        }
        for(auto c:B){
            --cnt[c];
            if(cnt[c]<0) return false;
        }
        if(same && A==B) return true;
        int dif = 0;
        for(int i=0;i<A.size() && dif<=2; ++i) if(A[i]!=B[i])
++dif;
        return dif==2;
    }
};
```

857

1. 按照ratio从小到大排序，保证最小化wage。再用heap记录quality，如果quality过大就删除，保证wage最小化。

```
class Solution {
    class Worker {
        int quality;
        int wage;
        double ratio;
        public Worker(int quality, int wage) {
            this.quality = quality;
            this.wage = wage;
            this.ratio = (double) wage / quality;
        }
```

```
    }

    public double mincostToHireWorkers(int[] quality, int[] wa
ge, int K) {
        PriorityQueue<Worker> pq = new PriorityQueue<Worker>
(1, new Comparator<Worker>(){
            public int compare(Worker a, Worker b) {
                return b.quality - a.quality;
            }
        });
        List<Worker> workers = new ArrayList<>();
        for (int i = 0; i < quality.length; i++) {
            workers.add(new Worker(quality[i], wage[i]));
        }
        Collections.sort(workers, (a, b) -> Double.compare(a.r
atio, b.ratio));
        double sum = 0;
        double res = Double.MAX_VALUE;
        for (Worker w : workers) {
            sum += w.quality;
            pq.offer(w);
            if (pq.size() > K) {
                sum -= pq.poll().quality;
            }
            if (pq.size() == K) res = Math.min(res, sum * w.ra
tio);

        }
        return res;
    }
}
```

2. 同上:

```
class Solution {
    typedef pair<int, int> ii;
    static bool cmp(const ii&a, const ii&b){
        if(a.first * b.second == a.second * b.first) return a.
first > b.first;
        return a.first * b.second < a.second * b.first;
    }
public:
    double mincostToHireWorkers(vector<int>& quality, vector<i
nt>& wage, int K) {
        vector<ii> W;
        int n = wage.size(), sum = 0;
        for(int i=0;i<n;++i) W.push_back(ii(wage[i], quality
[i]));
        sort(W.begin(), W.end(), cmp);
        priority_queue<int> Q;
        for(int i=0;i<K;++i){
            sum += W[i].second;
            Q.push(W[i].second);
        }
        double ans = double(sum) * W[K-1].first / W[K-1].secon
d;
        for(int i=K;i<n;++i){
            sum -= Q.top();
            Q.pop();
            sum += W[i].second;
            Q.push(W[i].second);
            ans = min(ans, double(sum) * W[i].first / W[i].sec
ond);
        }
```

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

188

1. 经典递归：

```
class Solution {
public:
    int maxProfit(int k, vector<int>& prices) {
        int n = prices.size(), ans = 0;
        if(k > (n+1)/2){
            for(int i=1; i<n; ++i) if(prices[i] > prices[i-1])
ans += prices[i] - prices[i-1];
            return ans;
        }
        vector<int> dp(n, 0);
        for(int r=0; r<k; ++r){
            for(int j=1, m=dp[0]-prices[0]; j<n; ++j){
                m = max(m, dp[j]-prices[j]);
                dp[j] = max(dp[j-1], prices[j] + m);
            }
        }
        return dp[n-1];
    }
};
```

552

1. dp

```
public int checkRecord(int n) {
        if (n == 1) return 3;
        if (n == 2) return 8;
```

```
        int m = 1000000007;
        int[] P = new int[n];
        int[] A = new int[n];
        int[] L = new int[n];
        P[0] = 1;
        L[0] = 1;
        L[1] = 3;
        A[0] = 1;
        A[1] = 2;
        A[2] = 4;

        for (int i = 1; i < n; i++) {
            P[i - 1] %= m;
            A[i - 1] %= m;
            L[i - 1] %= m;

            P[i] = ((P[i - 1] + L[i - 1]) % m + A[i - 1]) % m;
            if (i > 1) L[i] = ((A[i - 1] + P[i - 1]) % m + (A
[i - 2] + P[i - 2]) % m) % m;
            if (i > 2) A[i] = ((A[i - 1] + A[i - 2]) % m + A[i
- 3]) % m;
        }
        return ((A[n - 1] + P[n - 1]) % m + L[n - 1]) % m;
    }
}
```

2. 同上

```
class Solution {
    int M = 1E9 + 7;
    void add(int &x, int &y){
        x += y;
        if(x >= M) x -= M;
```

```
    }
public:
    /*
    i / 3: # of A
    i % 3: # of L
    */
    int checkRecord(int n) {
        vector<int> dp{1, 0, 0, 0, 0, 0};
        int ans = 0;
        for(int j=0; j<n; ++j){
            vector<int> tmp(6, 0);
            for(int i=0; i<6; ++i){
                if(i%3 == 0) for(int k=0; k<i+3; ++k) add(tmp
[i], dp[k]);
                else add(tmp[i], dp[i-1]);
            }
            swap(dp, tmp);
        }
        for(int i=0; i<6; ++i) {
            cout<<dp[i]<<endl;
            add(ans, dp[i]);
        }
        return ans;
    }
};
```

161

1.

```
public boolean isOneEditDistance(String s, String t) {
        for (int i = 0; i < Math.min(s.length(), t.length());
i++) {
```

```
            if (s.charAt(i) != t.charAt(i)) {
                if (s.length() == t.length()) return s.substri
ng(i + 1).equals(t.substring(i + 1));
                else if (s.length() < t.length()) return s.sub
string(i).equals(t.substring(i + 1));
                else return s.substring(i + 1).equals(t.substr
ing(i));
            }
        }
        return Math.abs(s.length() - t.length()) == 1;
    }
```

2. 直接比就行了:

```
class Solution {
public:
    bool isOneEditDistance(string s, string t) {
        if(abs((int)s.size() - (int)t.size()) > 1) return fals
e;
        if(s.size() > t.size()) swap(s, t);
        int cnt = 0;
        for(int i=0, j=0; i<s.size()&&cnt < 2; ){
            if(s[i] == t[j]) ++i, ++j;
            else{
                ++cnt;
                if(s.size() < t.size()) ++j;
                else ++i, ++j;
            }
        }
        return (cnt==1) || (cnt==0 && t.size()>s.size());
    }
};
```

144

1. Nothing to say

```
class Solution {
public:
    vector<int> preorderTraversal(TreeNode* root) {
        stack<TreeNode *> S;
        vector<int> ans;
        while(root || !S.empty()){
            while(root){
                if(root->right) S.push(root->right);
                ans.push_back(root->val);
                root = root->left;
            }
            if(!S.empty()){
                root = S.top();
                S.pop();
            }
        }
        return ans;
    }
};
```

824

1.

```
class Solution {
    Set<Character> set = new HashSet<>(Arrays.asList('a', 'o',
'e', 'u', 'i', 'A', 'O', 'E', 'U', 'I'));
    public String toGoatLatin(String S) {
        StringBuilder sb = new StringBuilder();
        String[] arr = S.split(" ");
        int i = 1;
```

```
        for (String str : arr) {
            char c = str.charAt(0);
            if (set.contains(c)) {
                str += "ma";
            } else {
                str = str.substring(1, str.length()) + c + "m
a";
            }
            for (int k = 0; k < i; k++) {
                str += "a";
            }
            i++;
            sb.append(str);
            sb.append(" ");
        }
        return sb.toString().trim();
    }
}
```

2. Python one line:

```
class Solution(object):
    def toGoatLatin(self, S):
        """
        :type S: str
        :rtype: str
        """
        return ' '.join([
            x+'ma' + (i+1)*'a' if x[0].lower() in "aeiou" else
x[1:] + x[0] + "ma" + (i+1)*'a' for i, x in enumerate(S.split
())
        ])
```

457

1. 并查集

```
class Solution {
    vector<int> P;
    int fr(int i){
        if(P[i] == i) return i;
        return P[i] = fr(P[i]);
    }
public:
    bool circularArrayLoop(vector<int>& nums) {
        int n = nums.size();
        P.resize(n);
        for(int i=0; i<n; ++i) P[i] = i;
        for(int i=0; i<n; ++i) if(nums[i]>0){
            int j = ((i + nums[i]) % n + n) % n;
            if(j == i) continue;
            if(fr(i) == fr(j)) return true;
            P[fr(i)] = fr(j);
        }
        for(int i=0; i<n; ++i) P[i] = i;
        for(int i=0; i<n; ++i) if(nums[i]<0) {
            int j = ((i + nums[i]) % n + n) % n;
            if(j == i) continue;
            if(fr(i) == fr(j)) return true;
            P[fr(i)] = fr(j);
        }
        return false;
    }
};
```