August 6, 2018 题目：710,837,240,731,484,30,574,463,381,267,139,570,461

710

1. Move the last few number to those in blacklist

```
class Solution {
    Map<Integer, Integer> map;
    int M;
    Random r;
    public Solution(int N, int[] blacklist) {
        map = new HashMap<>();
        r = new Random();
        for (int b : blacklist) {
            map.put(b, -1);
        }
        M = N - map.size();
        for (int b : blacklist) {
            if (b >= M) continue;
            while (map.containsKey(N - 1)) {
                N--;
            }
            map.put(b, N - 1);
            N--;
        }
    }

    public int pick() {
        int k = r.nextInt(M);
        if (map.containsKey(k)) {
            return map.get(k);
        }
        return k;
    }
}
```

2. Same as above
3. 最菜的二分：

```
class Solution {
    vector<int> B, P;
    int n;
public:
    Solution(int N, vector<int> blacklist): n(N), B(blacklist)
{
        sort(B.begin(), B.end());
        P.resize(B.size() + 1, 0);
        for(int i=1; i<=B.size(); ++i) P[i] += P[i-1] + (i>1?
(B[i-1]-B[i-2]-1): B[i-1]);
    }
    int pick() {
        int K=rand()%(n - (int)B.size()), l=0, r=B.size()+1;
        while(l<r-1){
            int c = (l+r)/2;
            if(P[c]<=K) l = c;
            else r = c;
        }
        if(l==0) return K;
        return B[l-1] + 1 + K- P[l];
    }
};
```

- [x] ~~Implement the above~~

4. 楼上的方法：

```
class Solution {
    unordered_map<int, int> M;
    int n;
public:
    Solution(int N, vector<int> blacklist): n(N) {
        set<int> S(blacklist.begin(), blacklist.end());
        vector<int> B(S.begin(), S.end());
        int j = B.size()-1, i=0;
        while(i<=j && n > 0){
            while(j>=i && n-1 == B[j]){
                --j;
                --n;
            }
            if(i>j) break;
            M[B[i++]] = (--n);
        }
    }
    int pick() {
        int K=rand()%n;
        if(M.count(K)) K = M[K];
        return K;
    }
};
```

837

1. dp, 概率，数学

- [ ] 遗留问题 @Zebo L

240

1. binary search

```
class Solution {
    public boolean searchMatrix(int[][] matrix, int target) {
        if (matrix == null || matrix.length == 0) return false;
        int row = 0, col = matrix[0].length - 1;
        while (col >= 0 && row < matrix.length) {
            if (matrix[row][col] == target) return true;
            else if (matrix[row][col] < target) row++;
            else col--;
        }
        return false;
    }
}
```

2. 只想到2轮binary search， 楼上牛逼
3. Zigzag bisection search: Guess Complexity `O(n+m)`

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        if(matrix.empty() || matrix[0].empty()) return false;
        int n = matrix.size(), m = matrix[0].size(), i = n-1, j = 0;
        if(matrix[0][0]>target || matrix[n-1][m-1]<target) return false;
        bool update = true;
        while(update){
            update = false;
            int r = i+1;
            i = -1;
            while(i < r-1){
                int c = (i + r)/2;
                if(matrix[c][j]<=target) i = c;
                else {
                    update = true;
                    r = c;
                }
            }
            if(i<0) return false;
            if(matrix[i][j] == target) return true;
            int l = j-1;
            j = m;
            while(l < j - 1){
                int c = (l + j) / 2;
                if(matrix[i][c] >= target) j = c;
                else{
                    l = c;
                    update = true;
                }
            }
            if(j >= m) return false;
            if(matrix[i][j] == target) return true;
        }
        return false;
    }
};
```
4. Complexity `N*log(M)`:

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target)
{
        if(matrix.empty() || matrix[0].empty()) return false;
        int n = matrix.size(), m = matrix[0].size();
        for(int i=0, j=m, l=-1; i<n; ++i){
            for(l=-1; l<j-1;){
                int c = (j+l)/2;
                if(matrix[i][c]>target) j = c;
                else l = c;
            }
            if(l == -1) return false;
            if(matrix[i][l] == target) return true;
        }
        return false;
    }
};
```

5. Complexity `O(N+M)`: 好写，还快

```
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target)
{
        if(matrix.empty() || matrix[0].empty()) return false;
        int n = matrix.size(), m = matrix[0].size();
        for(int i=0, j=m-1; i<n; ++i){
            while(j>=0 && matrix[i][j] > target) --j;
            if(j<0) return false;
            if(matrix[i][j] == target) return true;
        }
        return false;
    }
};
```

1. 做过，纯练手：

```
class MyCalendarTwo {
    map<int, int> one, two;
public:
    MyCalendarTwo() {
        one[-5] = two[-5] = -5;
        one[2*int(1E9)] = two[2*int(1E9)] = 2*int(1E9);
    }

    void insr(int s, int e){
        auto it = --one.upper_bound(s);
        if(it->second < s) ++it;
        while(it->first <= e){
            if(max(it->first, s) < min(e, it->second)){
                two[max(it->first, s)] = min(e, it->second);
            }
            s = min(it->first, s);
            e = max(it->second, e);
            it = one.erase(it);
        }
        one[s] = e;
    }
    bool canInsr(int s, int e){
        auto it = --two.upper_bound(s);
        if(it->second > s) return false;
        ++it;
        if(it->first < e) return false;
        return true;
    }
    bool book(int start, int end) {
        if(canInsr(start, end)){
            insr(start, end);
            return true;
        }
        return false;
    }
};
```

484

1.

```
class Solution {
    public int[] findPermutation(String s) {
        int len = s.length() + 1;
        int[] res = new int[len];
        for (int i = 1; i <= res.length; i++) {
            res[i - 1] = i;
        }
        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) == 'D') {
                int lo = i;
                while (i < s.length() && s.charAt(i) == 'D') {
                    i++;
                }
                reverse(res, lo, i);
            }
        }
        return res;
    }

    public void reverse(int[] res, int i, int j) {
        while (i < j) {
            int t = res[i];
            res[i] = res[j];
            res[j] = t;
            i++;
            j--;
        }
    }
}
```

2. 受楼上启发，扔掉`priority_queue`:

```
class Solution {
public:
    vector<int> findPermutation(string s) {
        int n = s.size() + 1, i = 0, j = 0, m = 1;
        vector<int> ans(n, 0);
        priority_queue<int, vector<int>, greater<int>> Q;
        while(i<n){
            while(i<n-1 && s[i]=='I') ans[i++] = m++;
            j = s.find('I', i);
            if(j<0) j = s.size();
            for(int k=j; k>=i; --k) ans[k] = m++;
            i = j + 1;
        }
        return ans;
    }
};
```

30

1. 烦得一B

```
class Solution {
public:
    vector<int> findSubstring(string s, vector<string>& words)
{
        if(words.empty() || s.empty()) return vector<int>();
        int l = words[0].size(), n = s.size();
        unordered_map<string, int> H;
        for(string w: words) H[w]++;
        vector<int> ans;
        for(int x = 0; x < l; ++x){
            unordered_map<string, int> res;
            string cur = "";
            int i = x, j = x, cnt = 0;
            while(i+l<=n && j+l<=n){
                while(j+l<=n && cnt < words.size()){
                    cur = s.substr(j, l);
                    if(H.count(cur)){
                        res[cur]++;
                        ++cnt;
                        j+=l;
                        if(res[cur]>H[cur]) break;
                    }
                    else break;
                }
                if(!H.count(cur)){
                    i = j = j + l;
                    cnt = 0;
                    res.clear();
                    continue;
                }
                else if(cnt == words.size() && res[cur]<=H[cur]){
                    ans.push_back(i);
                    string tmp = s.substr(i, l);
                    res[tmp]--;
                    if(!res[tmp]) res.erase(tmp);
                    --cnt;
                    i += l;
                }
                else if(j+l>n) break;
                else{
                    while(res[cur] > H[cur]){
                        string tmp = s.substr(i, l);
                        res[tmp]--;
                        if(!res[tmp]) res.erase(tmp);
                        i += l;
                        --cnt;
                    }
                }
            }
        }
        return ans;
    }
};
```

- [ ] Review it and Use Hash to solve it @Zebo L

574

SQL

463

1. BFS:

```
class Solution {
    typedef pair<int, int> ii;
    int d[4] = {1, -1, 0, 0};
    int BFS(int i, int j, vector<vector<int>>&G){
        int N = G.size(), M = G[0].size(), ans = 0;
        queue<int> Q;
        Q.push(i*M + j);
        unordered_set<int> S{i*M + j};
        while(!Q.empty()){
            int x = Q.front()/M, y = Q.front()%M;
            Q.pop();
            for(int k=0; k<4; ++k){
                int x1 = x + d[k], y1 = y + d[3-k];
                if(x1<0 || x1>=N || y1<0 || y1>=M || G[x1][y1]
==0) ++ans;
                else if(!S.count(x1*M + y1)){
                    Q.push(x1*M + y1);
                    S.insert(x1*M + y1);
                }
            }
        }
        return ans;
    }
public:
    int islandPerimeter(vector<vector<int>>& grid) {
        if(grid.empty() || grid[0].empty()) return 0;
        for(int i=0; i<grid.size(); ++i) for(int j=0; j<grid
[0].size(); ++j) if(grid[i][j]) return BFS(i, j, grid);
        return 0;
    }
};
```

2. 直接One pass 简单而且更快：

```
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;
    }
};
```

381

1. Use map to keep track of the indexes for each element

```
class RandomizedCollection {
    Map<Integer, Set<Integer>> map;
    List<Integer> list;
    /** Initialize your data structure here. */
    public RandomizedCollection() {
        map = new HashMap<>();
        list = new ArrayList<>();
    }

    /** Inserts a value to the collection. Returns true if the
collection did not already contain the specified element. */
```

```
    public boolean insert(int val) {
        boolean res = false;
        list.add(val);
        if (!map.containsKey(val)) {
            res = true;
            map.put(val, new HashSet<>());
        }
        map.get(val).add(list.size() - 1);
        return res;
    }

    /** Removes a value from the collection. Returns true if t
he collection contained the specified element. */
    public boolean remove(int val) {
        if (!map.containsKey(val)) return false;

        int pos = map.get(val).iterator().next();
        map.get(val).remove(pos);
        int last = list.get(list.size() - 1);
        list.set(pos, last);
        map.get(last).add(pos);
        map.get(last).remove(list.size() - 1);
        list.remove(list.size() - 1);

        if (map.get(val).isEmpty()) map.remove(val);

        return true;
    }

    /** Get a random element from the collection. */
    public int getRandom() {
        Random r = new Random();
        int n = r.nextInt(list.size());
        return list.get(n);
    }
}
```

2. 同上，赐题比之前的medium要简单：

```
class RandomizedCollection {
public:
    /** Initialize your data structure here. */
    vector<int> C;
    unordered_map<int, unordered_set<int>> pos;
    RandomizedCollection() {}
    bool insert(int val) {
        bool ans = !pos.count(val);
        pos[val].insert((int)C.size());
        C.push_back(val);
        return ans;
    }
    bool remove(int val) {
        if(C.empty() || !pos.count(val)) return false;
        if(C.back() == val){
            C.pop_back();
            pos[val].erase((int)C.size());
            if(pos[val].empty()) pos.erase(val);
            return true;
        }
        int j = C.size() - 1, i = *pos[val].begin();
        pos[C[j]].erase(j);
        pos[C[j]].insert(i);
        pos[val].erase(i);
        if(pos[val].empty()) pos.erase(val);
        swap(C[i], C[j]);
        C.pop_back();
        return true;
    }
    int getRandom() {
        return C[rand()%int(C.size())];
    }
};
```

267

1. backtracking

```
class Solution {
    public List<String> generatePalindromes(String s) {
        List<String> res = new ArrayList<>();
        int[] counts = new int[128];
        for (char c : s.toCharArray()) {
            counts[c]++;
        }

        int odd = -1;
        int n = 0;
        for (int i = 0; i < 128; i++) {
            if (counts[i] % 2 != 0) {
                odd = i;
                if (n == 1) return res;
                n++;

            }
        }
        helper(res, counts, odd != -1 ? ((char) odd) + "" :
"", s.length());
        return res;
    }

    public void helper(List<String> list, int[] counts, String
s, int length) {
        if (s.length() == length) {
            list.add(s);
            return;
        } else {
            for (int i = 0; i < counts.length; i++) {
            if (counts[i] > 1) {
                counts[i] -= 2;
                helper(list, counts, ((char) i) + s + ((char)
i), length);
                counts[i] += 2;
                }
            }
        }

    }
}
```

2. 同上，注意dfs过程中不要用`map.erase()`，否则会有奇怪的bug：

```
class Solution {
    int L;
    string mid;
    void dfs(vector<string> &ans, string cur, unordered_map<ch
ar, int> &rest){
        if(cur.size() == L){
```

};

```
            string s = cur;
            reverse(cur.begin(), cur.end());
            s += mid + cur;
            ans.push_back(s);
            return;
        }

        for(auto p: rest) if(p.second>0){
            --rest[p.first];
            cur += p.first;
            dfs(ans, cur, rest);
            cur.pop_back();
            rest[p.first]++;
        }
    }
public:
    vector<string> generatePalindromes(string s) {
        vector<string> ans;
        unordered_map<char, int> rest;
        for(char c: s) ++rest[c];
        for(auto p: rest) if(p.second%2) mid+=p.first;
        if(mid.size()>1) return ans;
        L = (int)s.size()/2;
        for(auto &p: rest) p.second/=2;
        dfs(ans, "", rest);
        return ans;
    }
};
```

139

1. 这层dp 必不可少
    a. 一开始以为遇到`True`的时候 `dfs` 就开始返回，所以 `dp` 没什么用
    b. 当发现如果 search results 一直是 `False` ，没有 `dp` 就会超时

```
class Solution {
    #define CI(c) int((c) - 'a')
    struct T{
        bool isW;
        vector<T *> ch;
        T(): ch(vector<T *>(26, NULL)), isW(false) {}
    };
    void ins(T *r, string w){
        for(char c: w){
            if(!r->ch[CI(c)]) r->ch[CI(c)] = new T();
            r = r->ch[CI(c)];
        }
        r->isW = true;
    }
    unordered_map<int, int> dp;
    bool dfs(T*root, int k, string &s){
        if(k == s.size()) return true;
        if(dp.count(k)) return dp[k];
        int i = k;
        T *r = root;
        while(i<s.size()){
            if(!r->ch[CI(s[i])]) return dp[k]=false;
            r = r->ch[CI(s[i++])];
            if(r->isW && dfs(root, i, s)) return dp[k]=true;
        }
        return dp[k]=false;
    }
public:
    bool wordBreak(string s, vector<string>& wordDict) {
        T *root = new T();
        for(auto w: wordDict) ins(root, w);
        return dfs(root, 0, s);
    }
};
```

570

SQL

461

1. Straight forward

```
class Solution {
public:
    int hammingDistance(int x, int y) {
        int ans = 0;
        for(int i=0; i<32; ++i) if((1&(x>>i))^(1&(y>>i))) ++an
s;
        return ans;
    }
};
```