August 30, 2018 题目：789,830,247,808,686,485,633,194,183,358,618,613,114,546,192,126

789

1. 做过，比较下Manhattan distance即可：

```
class Solution {
public:
    bool escapeGhosts(vector<vector<int>>& ghosts, vector<int>& target) {
        int dis = abs(target[0]) + abs(target[1]);
        for(auto p: ghosts) if(abs(p[0] - target[0]) + abs(p[1]-target[1]) <= dis) return false;
        return true;
    }
};
```

2. Manhattan distance

```
class Solution(object):
    def escapeGhosts(self, ghosts, target):
        targetDist = abs(target[0]) + abs(target[1])
        for g in ghosts:
            if abs(target[0] - g[0]) + abs(target[1] - g[1]) <= targetDist:
                return False
        return True
```

830

1. One pass:

```
class Solution {
public:
    vector<vector<int>> largeGroupPositions(string s) {
        vector<vector<int>> ans;
        for(int i=1, cnt=1; i<=s.size(); ++i){
            if(i<s.size() && s[i]==s[i-1]) ++cnt;
            else{
                if(cnt >= 3) ans.push_back({i-cnt, i-1});
                cnt = 1;
            }
        }
        return ans;
    }
};
```

2. sliding window

```
class Solution(object):
    def largeGroupPositions(self, S):
        ans = []
        if len(S) < 3:
            return ans
        S += '.'
        left = 0
        right = 1

        while right < len(S):
            if S[left] != S[right]:
                if right - left >= 3:
                    ans.append([left, right - 1])
                left = right
            right += 1

        return ans
```

247

1. 这么简单的题做了好多遍才做对：

```
class Solution {
    const string mid = "018";
    const string ref = "16890";
public:
    vector<string> findStrobogrammatic(int n) {
        int l = n/2, prod = max(1, int(pow(5, n/2 - 1)) * 4);
        if(n % 2) prod *= 3;
        cout<< prod << endl;
        vector<string> ans;
        for(int k=0; k<prod; ++k){
            int m = k;
            string s;
            if(n % 2){
                s.push_back(mid[m%3]);
                m /= 3;
            }
            for(int i=0; i<l; ++i){
                s.push_back(ref[m%5]);
                m /= 5;
            }
            string t;
            for(int j=s.size() - 1; j>=n-2*l; --j){
                if(s[j] == '9') t.push_back('6');
                else if(s[j] == '6') t.push_back('9');
                else t.push_back(s[j]);
            }
            ans.push_back(t+s);
        }
        return ans;
    }
};
```

2. recursion

```
class Solution(object):
    def findStrobogrammatic(self, n):
        def helper(curr):
            if curr == 0: return ['']
            if curr == 1: return ['0', '1', '8']

            currList = helper(curr-2)

            ans = []

            for s in currList:
                if curr != n:
                    ans.append('0' + s + '0')
                ans.append('1' + s + '1')
                ans.append('8' + s + '8')
                ans.append('6' + s + '9')
                ans.append('9' + s + '6')
            return ans

        return helper(n)
```

808

1. 什么鬼dp

```
class Solution(object):
    def soupServings(self, N):
        if N > 4800: return 1
        memo = {}
        def dp(a, b):
            if (a, b) in memo: return memo[a, b]
            if a <= 0 and b <= 0: return 0.5
            if a <= 0: return 1
            if b <= 0: return 0
            memo[a, b] = 0.25 * (dp(a - 100, b) + dp(a - 75, b - 25) + dp(a - 50, b - 50) + dp(a - 25, b - 75))
            return memo[a, b]
        return dp(N, N)
```

2. 这nm是什么鬼：

```
class Solution {
    unordered_map<int, unordered_map<int, double>> dp;
    double dfs(int x, int y){
        if(x<=0 && y<=0) return 0.5;
        if(x<=0) return 1.;
        if(y<=0) return 0.;
        if(dp.count(x) && dp[x].count(y)) return dp[x][y];
        return dp[x][y] = 0.25 * (dfs(x-100, y) + dfs(x-75, y-25) + dfs(x-50, y-50) + dfs(x-25, y-75));
    }
public:
    double soupServings(int N) {
        if(N>=4800) return 1.;
        return dfs(N, N);
    }
};
```

686

1. Mingze
   the answer can only be ceil(len(B) / len(A)) or +1 if exact division

```
from math import ceil
```

```
class Solution:
    def repeatedStringMatch(self, A, B):
        ans = ceil(len(B) / len(A))
        if B in (A * ans):
            return ans
        if B in (A * (ans+1)):
            return ans+1
        return -1
```

2. Brute Force:

```
class Solution(object):
    def repeatedStringMatch(self, A, B):
        cnt = int((len(A) + len(B) -1) / len(A))
        if B in cnt * A: return cnt
        if B in (cnt + 1) * A: return cnt + 1
        return -1
```

- [ ] Try kmp @Zebo L

485

1. One Pass :

```
class Solution {
public:
    int findMaxConsecutiveOnes(vector<int>& nums) {
        int n = nums.size(), ans = 0;
        for(int i=0, cnt=0; i<=n; ++i){
            if(i<n && nums[i]) ++cnt;
            else {
                ans = max(ans, cnt);
                cnt = 0;
            }
        }
        return ans;
    }
};
```

633

1. 零 tm 也算啊：

```
class Solution {
public:
    bool judgeSquareSum(int c) {
        for(int i=0; i<=int(sqrt(c/2)); ++i){
            int k = c - i*i, j = int(sqrt(c-i*i));
            if(k == j*j) return true;
        }
        return false;
    }
};
```

2. 同上

```
class Solution(object):
    def judgeSquareSum(self, c):
        for a in range(int(c**0.5)+1):
            left = c - a**2
            b = int(left**0.5)
            if left == b**2:
                return True
        return False
```

194

BASH: 深刻意识到自己什么都不会

183

SQL

358

1. 思路不难，但极容易错：
    - 先填个数为 $\frac{n+k-1}{k}$ 的,从前往后，距离为 k， 最多填 $n\%k$ 个，否则`return "";`
    - 再填个数为n/k的，从后往前，距离为 k，填 $n-n\%k$ 个
    - 最后一次把其他数填进去。g

```
class Solution {
    typedef pair<int, char> ic;
public:
    string rearrangeString(string s, int k) {
        if(k<=1) return s;
        int n = s.size(), i = 0, j = 0, m = (int(s.size()) + k - 1)/k, v = k;
        int l = (n%k? n%k: k);
        unordered_map<char, int> cnt;
        for(char c: s) ++cnt[c];
        set<ic, greater<ic>> reg;
        for(auto p: cnt) reg.insert(ic(p.second, p.first));
        string ans(s);
        for(auto it = reg.begin(); it!=reg.end() && it->first>=n/k; ){
            if(it->first > m) return "";
            if(it->first == m){
                if(i >= l) return "";
                for(int z=0; z<m; ++z) ans[z*k + i] = it->second;
                ++i;
                it=reg.erase(it);
            }
            else if(it->first == n/k && v>l){
                --v;
                for(int z=0; z<n/k; ++z) ans[z*k + v] = it->second;
                it=reg.erase(it);
            }
            else ++it;
        }
        for(auto p: reg){
            for(int t=0; t<p.first; ++t){
                ans[j*k + i] = p.second;
                ++j;
                if(i<l && j>= (n+k-1)/k){
                    ++i;
                    j = 0;
                }
                else if(i >= l && j >= n/k){
                    ++i;
                    j = 0;
                }
            }
        }
        return ans;
    }
};
```

618

HARD SQL

613

SQL

114

1. Iterative with the help of stack:

```
class Solution {
public:
    void flatten(TreeNode* r) {
        stack<TreeNode *> S;
        while(r) {
            if(r->left){
                if(r->right) S.push(r->right);
                r->right = r->left;
                r->left = NULL;
            }
            else if(!r->right && !S.empty()){
                r->right = S.top();
                S.pop();
            }
            r = r->right;
        }
    }
};
```

546

192

BASH: 自己真是啥都不会

126

1. 用了九牛二虎之力才做出来：
    a. Build Edges （这一步也许可以省）
    b. Build Next（一个图，用BFS建立一个网络，使得每个节点的下一个节点即为通向end的路径的一部分）
    c. dfs，利用已经建立好的Next图，生成所有最小路径

```
class Solution {
    typedef pair<string, int> si;
    unordered_map<string, vector<string>> E, next;
    void init(string b, string e, unordered_set<string> &W){
        W.insert(b);
        W.insert(e);
        int l = b.size();
        for(auto it=W.begin(); it!=W.end(); ++it){
            string s = *it;
            for(int i=0; i<l; ++i) {
                string t(s);
                for(char c='a'; c<='z'; ++c) if(c!=s[i]){
                    t[i] = c;
                    if(W.count(t)) E[s].push_back(t);
                }
            }
        }
    }
    int buildLink(string b, string e, unordered_set<string>&
W){
        unordered_map<string, int> cnt;
        int L = -1;
        queue<string> Q;
        cnt[e] = 0;
        Q.push(e);
        while(!Q.empty()){
            string s = Q.front();
            Q.pop();
            if(s==b && L==-1) L = cnt[s];
            if(L!=-1 && cnt[s] > L) return L;
            int step = cnt[s];
            for(string t: E[s]){
                if(!cnt.count(t)){
                    cnt[t] = step + 1;
                    Q.push(t);
                }
                else if(cnt[t] == step-1){
                    next[s].push_back(t);
                }
            }
        }
        return L;
    }
    void dfs(vector<vector<string>> &ans, vector<string> cur, string tmp){
        cur.push_back(tmp);
        if(next[tmp].empty()) ans.push_back(cur);
        for(string s: next[tmp]) dfs(ans, cur, s);
    }
public:
    vector<vector<string>> findLadders(string beginWord, string endWord, vector<string>& wordList) {
        unordered_set<string> WL(wordList.begin(), wordList.end());
        vector<vector<string>> ans;
        if(!WL.count(endWord)) return ans;
        init(beginWord, endWord, WL);
        int l = buildLink(beginWord, endWord, WL);
        if(l == -1) return ans;
        dfs(ans, vector<string>(), beginWord);
        return ans;
    }
};
```

- [ ] 空间够的话这种方法也可以：https://leetcode.com/problems/word-ladder-ii/discuss/40434/C++-solution-using-standard-BFS-method-no-DFS-orbacktracking @Zebo L