airbnb面试题汇总

2017-06-01 | 面试汇总 |

前段时间面试了airbnb北京，顺便整理了下网上能搜到的面经，刷了一遍，主要是coding部分。

Palindrome Pairs

warm up：is_palindrome

```
bool isPalindrome (string s) {
    int left = 0, right = s.size() - 1;
    while (left < right) {
        if (s[left++] != s[right--]) return false;
    }
    return true;
}
```

给定一个字符串数组，找出所有的字符串对，该字符串对拼接起来是回文字符串（https://leetcode.com/problems/palindrome-pairs/?tab=Description）

```
#include <iostream>
#include <unordered_map>
#include <set>
#include <vector>
using namespace std;

bool isPalindrome(string word, int left, int right) {
    while (left < right)
        if(word[left++] != word[right--]) return false;
    return true;
}
   vector<vector<int> > palindromePairs(vector<string>& words) {
    unordered_map<string, int> idx;
    set<int> st;
    vector<vector<int> > ans;
    for (int i = 0; i < words.size(); ++i) {
        idx[words[i]] = i;
        st.insert(words[i].length());
    }
    for (int i = 0 ; i < words.size(); ++i) {
        string tmp = words[i];
        int len = tmp.length();
        reverse(tmp.begin(), tmp.end());
        if (idx.count(tmp) && idx[tmp] != i)
            ans.push_back({i, idx[tmp]});
        auto end = st.find(len);
        for (auto it = st.begin(); it != end; ++it) {
            if (idx.count(tmp.substr(len - *it)) && isPalindrome(tmp, 0, len - *it - 1))
                ans.push_back({i, idx[tmp.substr(len - *it)]});
            if (idx.count(tmp.substr(0, *it)) && isPalindrome(tmp, *it, len - 1))
                ans.push_back({idx[tmp.substr(0, *it)], i});
        }
    }
    return ans;
}
```

```
int main() {
    vector<string> words = {"bat", "tab", "cat"};
    auto ans = palindromePairs(words);
    for (auto pr : ans) {
        cout << pr[0] << " "<< pr[1] <<endl;
    }
    return 0;
}
```

Round numbers

When you book on airbnb the total price is:

Total price = base price + service fee + cleaning fee + ...

```
input : array of decimals ~ X
output : array of int ~ Y
```

But they need to satisfy the condition:

```
sum(Y) = round(sum(x))
minmize (|y1-x1| + |y2-x2| + ... + |yn-xn|)
Example1:
input = 30.3, 2.4, 3.5
output = 30 2 4

Example2:
input = 30.9, 2.4, 3.9
output = 31 2 4
```

先将所有floor(x)加起来统计出如果所有都floor的话还差多少，按照ceil以后需要加的价格排序，贪心取最小的补齐即可。代码如下：

```
# python
def roundNum(self, input):
    output = map(lambda x: floor(x), input)
    remain = int(round(sum(input)) - sum(output))
    it = sorted(enumerate(input), key=lambda x: x[1] - floor(x[1]))
    for _ in xrange(remain):
        output[it.pop()[0]] += 1
    return output

//c++
vector<int> roundNumber(vector<double>& prices) {
    vector<int> ans;
    int got = 0;
    double all = 0.0;
    vector<pair<double, int> > s_prices;
    for (int i = 0; i < prices.size(); ++i) {
        double price = prices[i];
        int tmp = int(floor(price));
        got += tmp;
        ans.push_back(tmp);
        all += price;
        s_prices.push_back(make_pair(price, i));
    }
    sort (s_prices.begin(), s_prices.end(),
         [](pair<double, int> a, pair<double, int> b)
         { return a.first - floor(a.first) > b.first - floor(b.first);});
    for (int i = 0; i < int(round(all)) - got; ++i) {
        ans[s_prices[i].second]++;
    }
    return ans;
}
```

2D itertaor + remove()

leetcode 251（https://leetcode.com/problems/flatten-2d-vector）

实现二维数组的迭代器，加上remove操作。代码如下：

```
class Vector2D {
private:
    vector<vector<int> >::iterator row, iBegin, iEnd;
    vector<int>::iterator col;
public:
    Vector2D(vector<vector<int> > &nums) {
    row = nums.begin();
    iBegin = nums.begin();
    iEnd = nums.end();
    if (!nums.empty()) col = row->begin();
    }
    int next() {
        if (hasNext()) {
           int val = *col;
           col++;
           return val;
       }
       throw "It's empty already!";
    }
    bool hasNext() {
        while (row != iEnd && col == row->end()) {
            ++row;
            if(row != iEnd)
                col = row->begin();
        }
        return row != iEnd;
    }
    void remove() {
        if (col == row->begin()) {
            auto pre = prev(row);
            while (pre != iBegin && (*pre).empty())
                pre = prev(pre);
            if (!(*pre).empty()) {
                (*pre).erase(prev((*pre).end()));
            } else {
                throw "Should call next() first!";
            }
        } else {
            (*row).erase(prev(col));
            col--;
        }
    }
};
```

ip2cidr

给出一个ipv4的range，找出最少的cidr可以覆盖这个range内的所有ip。

参考：

背景介绍https://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing

这个是个online转化工具http://www.ipaddressguide.com/cidr

大概的思路是group as much IPs as you can.

描述起来还真的麻烦呢，建议跑几个case，就理解了

code: http://stackoverflow.com/questions/33443914/how-to-convert-ip-address-range-to-cidr-in-java

解释： ——代表end-start能覆盖到的二进制位

start：xxxxxxx100000

end： xxxxxx——-这种情况下，先找出可以覆盖住xxxxxxx100000~xxxxxxx111111的cidr，start变为xxxxxxx100000 + 100000

end： xxxxxxxxx—-这种情况下，先找出可以覆盖住xxxxxxx100000~xxxxxxx101111的cidr，start变为xxxxxxx100000 + 10000

```
def ipToVal(ip):
    ip = ip.split(".")
    val = 0
    for x in ip:
        val = (val << 8) + int(x)
    return val


def ValToIp(val):
    ip, i = ["0"] * 4, 3
    while val:
        ip[i] = str(val % (1 << 8))
        val /= (1 << 8)
        i -= 1
    return ".".join(ip)
```

```
def range2cidr(start, end):
    if not start or not end or start.count('.') != 3 or end.count('.') != 3:
        return None
    start, end = ipToVal(start), ipToVal(end)
    if start > end:
        return None
    ans = []
    while start <= end:
        firstOne = start & (-start)
        maxMask = 32 - int(log(firstOne, 2))
        maxDiff = 32 - int(floor(log(end - start + 1, 2)))
        maxMask = max(maxMask, maxDiff)
        ip = ValToIp(start)
        ans.append(ip + "/" + str(maxMask))
        start += 2 ** (32 - maxMask)
    return ans

//C++
long ipToVal(string ip) {
    long val = 0;
    int i = 0;
    for (int j = 0; i < 4 && j < ip.length(); ++i) {
        auto nx = ip.find('.', j);
        if (nx == ip.npos) {
            val = (val << 8) + atoi(ip.substr(j).c_str());
            ++i;
            break;
        }
        val = (val << 8) + atoi(ip.substr(j, nx - j).c_str());
        j = nx + 1;
    }
    if (i != 4) throw "The ip is incorrect";
    return val;
}

string valToIp(long val) {
    string ip = "";
    for (int i = 0; i < 4; ++i) {
        ip = to_string(val % 256) + "." + ip;
        val /= 256;
    }
    ip.pop_back();
    return ip;
}
```

```
vector<string> range2cidr(string start, string end) {
    // try...catch
    long st = ipToVal(start), ed = ipToVal(end);
    vector<string> ans;
    while (st <= ed) {
        int lastOne = st & (-st);
        int maxMask = 32 - int((log(lastOne)/log(2)));
        int maxDiff = 32 - int(floor(log(ed - st + 1)/log(2)));
        maxMask = max(maxMask, maxDiff);
        string ip = valToIp(st);
        ans.push_back(ip + "/" + to_string(maxMask));
        st += int(pow(2, 32 - maxMask));
    }
    return ans;
}
```

Display Page list

用一个set来保存是否出现，加入以后删除原有元素，不够的话顺序补充

```
vector<vector<string> > paging(vector<string> items, int size) {
    vector<vector<string> > ans;
    const int n = items.size();
    for (int i = 0; i <= (n - 1) / size; ++i) {
        vector<string> tmp;
        unordered_set<string> st;
        for (auto it = items.begin(); it != items.end() && tmp.size() < size;) {
            if (st.count(*it)) {
                ++it;
                continue;
            }
            st.insert(*it);
            tmp.push_back(*it);
            items.erase(it);
        }
        for (auto it = items.begin(); it != items.end() && tmp.size() < size;) {
            tmp.push_back(*it);
            items.erase(it);
        }
        ans.push_back(tmp);
    }
    return ans;
}
```

menu order

点菜，菜价格为double，问如何正好花完手里的钱

解法：把菜单价格*100转成整数，题目转换成leetcode 40.Combination Sum II（https://leetcode.com/problems/combination-sum-ii/?tab=Description）

```
void helper(vector<int> &dish, vector<vector<double> > &ans, vector<double> &cur, int idx,
    if (money == 0) {
        ans.push_back(cur);
        return;
    }
    if (idx == dish.size() || money < 0 || money < dish[idx])
        return;
    int cnt = 1;
    while (cnt * dish[idx] <= money) {
        cur.push_back(dish[idx] / 100.0);
        helper(dish, ans, cur, idx + 1, money - cnt * dish[idx]);
        ++cnt;
    }
    while (--cnt) cur.pop_back();
    helper(dish, ans, cur, idx + 1, money);
```

```
}

vector<vector<double> > menuOrder(vector<double>& dish, double money) {
    vector<int> tmp;
    vector<vector<double> > ans;
    vector<double> cur;
    for (auto d : dish)
        tmp.push_back(int(d * 100));
    sort(tmp.begin(), tmp.end());
    helper(tmp, ans, cur, 0, int(money * 100));
    return ans;
}
```

Hilbert Curve

Hilbert Curve（http://bit-player.org/extras/hilbert/hilbert-construction.html）

Hilbert曲线可以无限阶下去，从1阶开始，落在一个矩阵里，让你写个function，三个参数（x,y,阶数），return 这个点（x,y）是在这阶curve里从原点出发的第几步

```
int hilbertCurve(int x, int y, int iter) {
    if (iter == 0) return 1;
    int areaCnt = (1 << (iter * 2 - 2)); // 每一块区域边长的边界值
    int borderLen = (1 << (iter - 1)); // 区域移动的长度

    if (x >= borderLen && y >= borderLen) //右上角区域 = 前一阶往右上角移动borderLen
        return areaCnt * 2 + hilbertCurve(x - borderLen, y - borderLen, iter - 1);
    else if (x < borderLen && y >= borderLen) //左上角区域 = 前一阶往上移动borderLen
        return areaCnt + hilbertCurve(x, y - borderLen, iter - 1);
    else if (x < borderLen && y < borderLen) //右下角区域 = 前一阶按照y=x对称
        return hilbertCurve(y, x, iter - 1);
    else //右下角区域 = 前一阶按照y=-x对称，然后右移2*borderLen - 1，上移borderLen - 1
    // 设原来坐标(a,b) => (-b, -a) => (2*borderLen - 1 - b, borderLen - 1 - a) = (x, y)
    // => a = borderLen - 1 - y, b = 2*borderLen - 1 - x
        return areaCnt * 3 + hilbertCurve(borderLen - 1 - y, 2 * borderLen - 1 - x, iter -
}
```

Alien Dictionary

leetcode 269

```
string alienOrder(vector<string> &words) {
    unordered_map<char, unordered_set<char> > mp;
    unordered_map<char, int> idx;
    unordered_set<char> st;
    queue<char> q;
    string ans = "";
    for (auto word : words) st.insert(word.begin(), word.end());
    for (int i = 0; i < words.size() - 1; ++i) {
        int j = 0, end = min(words[i].size(), words[i + 1].size());
        for (; j < end; ++j) {
            if (words[i][j] == words[i + 1][j]) continue;
            mp[words[i][j]].insert(words[i + 1][j]);
            break;
        }
        if (j == end && words[i].size() > words[i + 1].size())
            return ans;
    }
    for (auto m : mp) {
        for (auto s : m.second) {
            idx[s]++;
        }
    }
    for (auto s : st) {
        if (!idx.count(s)) {
            q.push(s);
```

```
            ans += s;
        }
    }
    while (!q.empty()) {
        char c = q.front();
        q.pop();
        auto next = mp[c];
        for (auto s : next) {
            if (--idx[s] == 0) {
                q.push(s);
                ans += s;
            }
        }
    }
    return ans.size() == st.size() ? ans : "";
}
```

有向图 求最少的点可以遍历所有点

（https://instant.1point3acres.com/thread/201190，https://instant.1point3acres.com/thread/197716）

解法，scc（strongly connected component）缩点 + 拓扑排序，太尼玛复杂了。。

解scc用了Kosaraju算法， 拓扑排序，代码如下：

```
void dfs(const vector<vector<bool> > &edges, vector<bool> &visited, vector<unordered_set<in
    visited[idx] = true;
    if (components.size() == id) components.push_back({});
    components[id].insert(idx);
    for (int i = 0; i < edges.size(); ++i) {
        if (edges[i][idx] && !visited[i]) dfs(edges, visited, components, i, id);
    }
}

void preDFS(const vector<vector<bool> > &edges, vector<bool> &visited, vector<int> &orders,
    visited[idx] = true;
    orders.push_back(idx);
    for (int i = 0; i < edges.size(); ++i) {
        if (edges[idx][i] && !visited[i]) preDFS(edges, visited, orders, i);
    }
}

vector<int> traverse(vector<vector<bool>> &edges) {
    const auto n = edges.size();
    vector<int> ans;
    if (n == 0) return ans;
    vector<bool> visited(n, false);
    vector<int> orders;// 记录伪拓扑排序的顺序
    vector<unordered_set<int> > components;//记录scc都包含哪些元素
    int id = 0;
    for (int i = 0; i < n; ++i) {
        if (!visited[i]) {
            preDFS(edges, visited, orders, i);
        }
    }
    fill(visited.begin(), visited.end(), false);
    for (int i = n - 1; i >= 0; --i) {
        if (!visited[orders[i]]) {
            dfs(edges, visited, components, orders[i], id);
            ++id;
        }
    }

    unordered_map<int, int> in;
    unordered_map<int, unordered_set<int> > next;
    for (int from = 0; from < id; ++from) {
        for (int to = 0; to < id; ++to) {
            if (from == to) continue;
```

```
                bool found = false;
                for (auto x : components[from]) {
                    if (found) break;
                    for (auto y : components[to])
                        if (edges[x][y]) {
                            in[to]++;
                            next[from].insert(to);
                            found = true;
                            break;
                        }
                }
            }
        }
        for (int i = 0; i < id; ++i) {
            if (in[i] == 0) {
                ans.push_back(*components[i].begin());
            }
        }
        return ans;
    }
```

c++(scc + union search)

```
typedef pair<int, int> pr;
typedef unordered_map<int, unordered_set<int> > connectInfo;
void fromDFS(int node, connectInfo from, vector<int>& order, unordered_set<int>& visited) {
    if(!visited.count(node)) {
        visited.insert(node);
        for(auto n : from[node])
            fromDFS(n, from, order, visited);
        order.insert(order.begin(), node);
    }
}

void toDFS(int node, int root, connectInfo to, unordered_map<int, int>& components) {
    if(!components.count(node)) {
        components[node] = root;
        for(auto n : to[node])
            toDFS(n, root, to, components);
    }
}

unordered_map<int, int> kosasrajus(connectInfo from, connectInfo to, const unordered_set<in
    unordered_set<int> visited;
    unordered_map<int, int> components;
    vector<int> order;
    for(auto node : nodes)
        fromDFS(node, from, order, visited);
    for(auto node : order)
        toDFS(node, node, to, components);
    return components;
}

vector<int> least_nodes(const vector<pr>& edges) {
    connectInfo from, to;
    unordered_set<int> nodes;
    vector<int> ans;
    for(auto edge : edges) {
        from[edge.first].insert(edge.second);
        to[edge.second].insert(edge.first);
        nodes.insert(edge.first);
        nodes.insert(edge.second);
    }
    unordered_map<int, int> components = kosasrajus(from, to, nodes);
    unordered_set<int> fromComponents, toComponents;
    for(auto edge : edges) {
        fromComponents.insert(components[edge.first]);
```

```
        if(components[edge.first] != components[edge.second]) {
            toComponents.insert(components[edge.second]);
        }
    }
    for(auto node : fromComponents)
        if(!toComponents.count(node))
            ans.push_back(node);
    return ans;
}
```

python:

```
def least_nodes(edges):
    pred = collections.defaultdict(set)
    succ = collections.defaultdict(set)

    for start, end in edges:
        pred[end].add(start)
        succ[start].add(end)

    components = kosarajus(pred, succ)
    component_pred = collections.defaultdict(set)
    component_succ = collections.defaultdict(set)
    for start, end in edges:
        if components[start] != components[end]:
            component_start = components[start]
            component_end = components[end]

            component_pred[component_end].add(component_start)
            component_succ[component_start].add(component_end)

    return set(component_succ.keys()) - set(component_pred.keys())

def kosarajus(pred, succ):
    all_nodes = set(pred.keys()) | set(succ.keys())

    order = []
    visited = set()
    def visit(node):
        if node not in visited:
            visited.add(node)
            for out_neighbor in succ[node]:
                visit(out_neighbor)
            order.insert(0, node)
    for node in all_nodes:
        visit(node)

    components = {}
    def assign(node, root):
        if node not in components:
            components[node] = root
            for in_neighbor in pred[node]:
                assign(in_neighbor, root)

    for node in order:
        assign(node, node)

    return components
```

meeting room

给一组meetings（每个meeting由start和end时间组成）。求出在所有输入meeting时间段内没有会议，也就是空闲的时间段。每个subarray都已经sort好。N个员工，每个员工有若干个interval表示在这段时间是忙碌的。求所有员工都不忙的intervals。

循环merge，然后遍历空闲区间（ps：另一种解法很简单，参考：这题最简单的方法就是把所有区间都拆成两个点，然后排序，然后扫描，每次碰到一个点如果是左端点就把busy_employees加1，否则减1，等到每次busy_employees为0时就是一个新的区间。这样复杂度O(MlogM)，M是总共区间数。）

```
//merge and search
vector<pr> merge(vector<pr> ft, vector<pr> sd) {//c++, merge
    if(ft.empty()) return sd;
    if(sd.empty()) return ft;
    vector<pr> ans;
    const int m = ft.size(), n = sd.size();
    int i = 0, j = 0;
    pr tmp(1, 1);
    while(i < m || j < n) {
        if ((i == m || tmp.second < ft[i].first) && (j == n || tmp.second < sd[j].first)) {
            ans.push_back(tmp);
            if(i == m) tmp = sd[j];
            else if(j == n) tmp = ft[i];
            else {
                tmp.first = min(ft[i].first, sd[j].first);
                tmp.second = min(ft[i].second, sd[j].second);
            }
        }
        if(i < m && ft[i].first <= tmp.second)
            tmp.second = max(tmp.second, ft[i++].second);
        if(j < n && sd[j].first <= tmp.second)
            tmp.second = max(tmp.second, sd[j++].second);
    }
    ans.push_back(tmp);
    return ans;
}
vector<pr> meetingRoom(vector<vector<pr> > meetings) {
    vector<pr> ans, tmp;
    const int n = meetings.size();
    if(n == 0) return ans;
    tmp = meetings[0];
    for(int i = 1; i < n; ++i) {
        tmp = merge(tmp, meetings[i]);
    }
    if(tmp[0].first > 1)
        ans.push_back(make_pair(1, tmp[0].first));
    for(int i = 0; i < tmp.size() - 1; ++i)
        ans.push_back(make_pair(tmp[i].second, tmp[i + 1].first));
    return ans;
}
```

```
//break and sort: C++
typedef pair<int, int> pr;
typedef pair<int, bool> timePoint;
vector<pr> meetingRoom(vector<vector<pr> > meetings) {
    vector<timePoint> times;
    vector<pr> ans;
    if ( meetings.empty()) return ans;
    for (auto meeting : meetings) {
        for (auto interval : meeting) {
            times.push_back(make_pair(interval.first, true));
            times.push_back(make_pair(interval.second, false));
        }
    }
    sort(times.begin(), times.end());
    int startCnt = 0, preTime = times[0].first;
    for(auto time : times) {
        bool starting = time.second;
        if (starting) {
            if (startCnt == 0 && time.first > preTime) {
                ans.push_back(make_pair(preTime, time.first));
            }
            ++startCnt;
```

```
        } else {
            if (startCnt == 1) preTime = max(preTime, time.first);
            --startCnt;
        }
    }
    return ans;
}

//break and sort: python
def find_free_time(schedules):
moment_status = []

for person_schedule in schedules:
    for interval in person_schedule:
        moment_status.append((interval[0], True))
        moment_status.append((interval[1], False))
moment_status.sort()
free_start = moment_status[0][0]
busy_count = 0
available_intervals = []
for moment, become_busy in moment_status:
    if become_busy:
        if busy_count == 0:
            if moment > free_start:
                available_intervals.append((free_start, moment))
        busy_count += 1
    else:
        if busy_count == 1:
            free_start = moment
        busy_count -= 1
return available_intervals
```

preference list

每个人都有一个preference的排序，在不违反每个人的preference的情况下得到总体的preference的排序 拓扑排序解决(https://instant.1point3acres.com/thread/207601)

```
vector<int> preferenceList(vector<vector<int> > &preList) {
    unordered_map<int, unordered_set<int> > mp;
    unordered_map<int, int> in;
    vector<int> ans;
    for(auto lt : preList) {
        for(int i = 1; i < lt.size(); ++i)
            mp[lt[i - 1]].insert(lt[i]);
    }
    for(auto m : mp)
        for(auto s : m.second)
            in[s]++;
    queue<int> q;
    for(int i = 0; i < preList.size(); ++i)
        if(!in.count(i)) {
            q.push(i);
            ans.push_back(i);
        }
    while(!q.empty()) {
        int c = q.front();
        q.pop();
        auto next = mp[c];
        for(auto s : next) {
            if(--in[s] == 0) {
                q.push(s);
                ans.push_back(s);
            }
        }
    }
    return ans;
}
```

buddy list

你和你的兄弟都有一个wishlist，找出和你相似度最高的。follow up是给出一个max值，找出你的buddy的wishlist里不在你的wishlist里的最多max个城市，根据buddy和你的重合程度来排序

例如：
你的wishlist是 a,b,c,d

buddy1 的wishlist 是 a,b,e,f, 有两个和你的一样，所以是你的buddy

buddy2 的wishlist 是 a,c,d,g, 有三个和你的一样，也是你的budy

问题是输出一个size最多为max的推荐城市列表。当size为10时，buddy1和buddy2的wishlist中不在你的wishlist中的城市都可以加入推荐中，因为buddy2的重合度更高，所以先输出buddy2中的，所以推荐为 g,e,f 当size为2时，推荐是g,e 或 g,f

代码我只写了重合度排名，推荐的话可以按照相似度从高到低遍历，找出不在你的wishlist中的输出，输出过程中可以标记是否已经输出

```
def find(self, nums, nums2):
    return sorted([(sum([num in set(nums) for num in nums2[i]]) / float(len(nums2[i])), i)
                   for i in xrange(len(nums2))])
```

flight ticket list

每一项包括departure, arrival, cost，然后给一个整数k, 表示最多允许k次中转。给定起始地点A，到达地点B, 要求输出从A到B的最小花费，最多k次中转。BFS一层一层扫。

```
def min_cost(flights, start, end, k):
    info = collections.defaultdict(set)
    for tour, cost in flights:
        st, ed = tour.split("->")
        info[st].add((ed, cost))

    cur_level = {start: 0}
    ans = 0x7FFFFFF
    for _ in xrange(k + 1):
        next_level = {}
        for port, cur_cost in cur_level.iteritems():
            for nx, cost in info[port]:
                if nx == end:
                    ans = min(ans, cost + cur_cost)
                else:
                    if nx not in next_level:
                        next_level[nx] = cost + cur_cost
                    else:
                        next_level[nx] = min(next_level[nx], cost + cur_cost)
        cur_level = next_level
    return ans

///C++: 太丑
typedef pair<string, int> costInfo;
int minCostFlight(const vector<string>& flights, string start, string end, int k) {
    unordered_map<string, set<costInfo> > costMap;
    unordered_map<string, int> reached[2];
    int ans = INT_MAX;
    for (auto flight : flights)
        auto nx = flight.find("->", 0);
        auto comma = flight.find(',', nx + 2);
        string st = flight.substr(0, nx);
        string ed = flight.substr(nx + 2, comma - nx - 2);
        int cost = atoi(flight.substr(comma + 1).c_str());
        costMap[st].insert(make_pair(ed, cost));
```

```
        }
        reached[0][start] = 0;
        for (int i = 0, j = 0; i <= k; ++i) {
            int nxIdx = (j + 1) % 2;
            reached[nxIdx].clear();
            for (auto st : reached[j]) {
                for(auto ed : costMap[st.first]) {
                    if (ed.first == end) {
                        ans = min(ans, ed.second + st.second);
                    } else {
                        if (!reached[nxIdx].count(ed.first))
                            reached[nxIdx][ed.first] = ed.second + st.second;
                        else
                            reached[nxIdx][ed.first] = min(reached[nxIdx][ed.first], ed.second
                    }
                }
            }
            j = nxIdx;
        }
        return ans;
    }
```

URL Shortener

（https://instant.1point3acres.com/thread/196339）

看描述好像是url里的id如果有某些位置大小写换了会导致原来的url decode有问题，需要重写encode方法，回溯改某些位的大小写判断

```
class decodeURL {
public:
    int decode(string url) {
        string dUrl = "kljJJ324hijkS_";
        if (url == dUrl) return 848662;
        return -1;
    }

    int decodeFind(string url) {
        return helper(url, 0);
    }

private:
    int helper(string s, int idx) {
        if (idx == s.length())
            return decode(s);
        if (isalpha(s[idx])) {
            int uid = helper(s.substr(0, idx) + char(tolower(s[idx])) + s.substr(idx + 1),
            int lid = helper(s.substr(0, idx) + char(toupper(s[idx])) + s.substr(idx + 1),
            if (uid != -1 || lid != -1)
                return uid != -1 ? uid : lid;
            return -1;
        } else {
            return helper(s, idx + 1);
        }
    }
};
```

wizards

There are 10 **wizards**, 0-9, you are given a list that each entry is a list of wizards known by wizard. Define the cost between wizards and wizard as square of different of i and j. To find the min cost between 0 and 9.

说白了，就是带权重的最短距离，最优解是Dijkstra algorithm。似乎面试官说，只要普通的BFS能得到解也是可以的，Dijkstra我最近正好写过，所以也写出来了。（https://instant.1point3acres.com/thread/218032）

```
def min_distance(wizards, start=0, end=9):
     # info = collections.defaultdict(set)
    # for idx, wizard in enumerate(wizards):
    #     info[idx] = set(wizard)
    cur_level = {start: 0}
    ans = 0x7FFFFFFF
    for _ in xrange(10):
        next_level = {}
        for idx, cur_cost in cur_level.iteritems():
            if idx >= len(wizards):
                continue
            for nx in wizards[idx]:
                cost = cur_cost + (nx - idx) ** 2
                if nx == end:
                    ans = min(ans, cost)
                else:
                    if nx not in next_level:
                        next_level[nx] = cost
                    else:
                        next_level[nx] = min(next_level[nx], cost)
        cur_level = next_level
    return ans
```

```
unordered_map<int, int> bfs(const unordered_map<int, int>& preLevel, unordered_map<int, uno
    unordered_map<int, int> nextLevel;
    for ( auto pre : preLevel) {
        for (auto nx : next[pre.first]) {
            int dis = (nx - pre.first) * (nx - pre.first) + pre.second;
            if (nx == end) {
                ans = min(ans, dis);
            } else if(!nextLevel.count(nx)) {
                nextLevel[nx] = dis;
            } else
                nextLevel[nx] = min(nextLevel[nx], dis);
        }
    }
    return k == 1 ? nextLevel : bfs(nextLevel, next, ans, k - 1, end);
}

int wizards(const vector<vector<int> > &wizards, int start, int end) {
    unordered_map<int, unordered_set<int> > next;
    unordered_map<int, int> curLevel;
    int ans = INT_MAX;
    curLevel[start] = 0;
    for (int i = 0; i < wizards.size(); ++i) {
        for (auto nx : wizards[i]) {
            next[i].insert(nx);
        }
    }
    bfs(curLevel, next, ans, end, end);
    return ans;
}
```

模拟倒水

water land。 比如terrian是[3,2,1,2] print出来就是

```
*


* *


* * * *


* * * *
```

然后给你一个dumpPoint，一个waterAmount，比如dumpPoint 1, waterAmount 2，因为有重力，所以是从index 2开始加水

*

**w*

****

****

terrian两边是最高，模拟，先向左找到非递增的最低点，如果该点和dumpPoint一样高，往右继续找非递增的最低点，如果一样高就放到dumpPoint，不一样的话放置在非递增的最低点

```
void getWaterLevel(vector<int> &height, int position, int count) {
    if(height.empty()) return;
    const int n = height.size();
    vector<int> water(n, 0);
    while(count--) {
        int putLocation = position;
        int left = position, right = position;
        while(left >= 1) {
            if(height[left - 1] + water[left - 1] > height[left] + water[left]) break;
            --left;
        }
        if(height[left] + water[left] < height[position] + water[position])
            putLocation = left;
        else {
            while(right < n - 1) {
                if(height[right + 1] + water[right + 1] > height[right] + water[right]) bre
                ++right;
            }
            if(height[right] + water[right] < height[position] + water[position])
                putLocation = right;
        }
        water[putLocation]++;
    }

    int highest = 0;
    for(int i = 0; i < n; ++i)
        if(height[i] + water[i] > highest)
            highest = height[i] + water[i];
    for(int h = highest; h >= 1; --h) {
            for(int i = 0; i < n; ++i) {
                if(height[i] + water[i] < h) cout<<" ";
                else if(height[i] < h) cout<<"w";
                else cout<<"*";
            }
            cout<<endl;
        }
    }
```

text justification

leetcode 68(https://leetcode.com/problems/text-justification/?tab=Description)

```
vector<string> justify(vector<string> &words, int L) {
    vector<string> ans;
    const int n = words.size();
    for (int i = 0; i < n;) {
        int num = 0, len = 0;
        while (i + num < n && words[i + num].size() + len <= L - num) {
            len += words[i + num].size();
            ++num;
```

```
        }
        string tmp = words[i];
        for (int j = 1; j < num; ++j) {
            if (i + num >= n) tmp += " ";
            else tmp += string((L - len) / (num - 1) + (j  <= (L - len) % (num - 1)), ' ');
            tmp += words[i + j];
        }
        tmp += string(L - tmp.size(), ' ');
        ans.push_back(tmp);
        i += num;
    }
    return ans;
}
```

string pyramids transition matrix

给一个满二叉树的所有叶子，比如ABCDEF，然后给一个map，记录了左右孩子分别给了的时候，父亲节点可能的值。例如 左A 右B =》 AC，意味着倒数第二层第一个节点可以是A或者是C。然后要求是给几个字母，问这个树的root节点是否可能是这几个字母之一。follow up是加速，记忆化搜索（不是很好写）。

```
def generate_status(all_status, matrix):
    if len(all_status) == 1:
        return all_status[0]

    next_all_status = []
    for i in xrange(len(all_status) - 1):
        cur_status = set()
        for first in all_status[i]:
            for second in all_status[i + 1]:
                cur_status |= set(list(matrix[first][second]))
        next_all_status.append(cur_status)

    return generate_status(next_all_status, matrix)


def is_legal_status(nodes, status, matrix):
    all_status = [set(node) for node in nodes]
    return status in generate_status(all_status, matrix)

nodes = "ABCD"
matrix = collections.defaultdict(lambda: collections.defaultdict(list))
matrix['A']['A'] = ['B']
matrix['A']['B'] = ['A', 'C']
matrix['A']['C'] = ['D']
matrix['A']['D'] = ['A']
matrix['B']['A'] = ['D']
matrix['B']['B'] = ['B', 'C']
matrix['B']['C'] = ['A']
matrix['C']['D'] = ['B']
print is_legal_status(nodes, 'D', matrix)
```

```
typedef unordered_map<char, unordered_map<char, unordered_set<char> > > matrixInfo;
void generateStatus(vector<unordered_set<char> >& allStatus, matrixInfo& matrix) {
    if (allStatus.size() == 1) return;
    const int n = allStatus.size();
    for (int i = 0; i < n - 1; ++i) {
        unordered_set<char> st;
        for (auto first : allStatus[i]) {
            for (auto second : allStatus[i + 1]) {
                st.insert(matrix[first][second].begin(), matrix[first][second].end());
            }
        }
        allStatus[i] = st;
    }
    allStatus.pop_back();
    generateStatus(allStatus, matrix);
}
```

```
bool checkStatus(matrixInfo& matrix, char result, const string status) {
    vector<unordered_set<char> > allStatus;
    for (auto c : status) {
        unordered_set<char> tmp;
        tmp.insert(c);
        allStatus.push_back(tmp);
    }
    generateStatus(allStatus, matrix);
    return allStatus[0].count(result) != 0;
}

int main() {
    matrixInfo mi;
    mi['A']['A'].insert('B');
    mi['A']['B'].insert('A');
    mi['A']['B'].insert('C');
    mi['A']['C'].insert('D');
    mi['A']['D'].insert('A');
    mi['B']['A'].insert('D');
    mi['B']['B'].insert('B');
    mi['B']['B'].insert('C');
    mi['B']['C'].insert('A');
    mi['C']['D'].insert('B');
    cout<<checkStatus(mi, 'A', "ABCD")<<endl;
}
```

sliding game

九宫格，一共8个方块，从1-8，一个方块空出来，然后打乱之后通过SLIDE还原，这个题要推广到N宫格，先实现这个游戏，然后对于一个任意的BOARD，要你把他解出来

```
def dis(x, y):  # A* evaluation func
    return (x - 2) ** 2 + (y - 2) ** 2


def play(board):
    m, n = len(board), len(board[0])
    x, y = 0, 0
    for i in xrange(m):
        for j in xrange(n):
            if board[i][j] == '0':
                x, y = i, j
    board_key = ''.join(''.join(row) for row in board)
    heap = [(dis(x, y), x, y, board_key)]
    visited = set()
    while heap:
        _, x, y, cur = heapq.heappop(heap)
        if cur in visited:
            continue
        visited.add(cur)
        if cur == "123456780":
            return True
        for dx, dy in zip((1, -1, 0, 0), (0, 0, 1, -1)):
            new_x, new_y = x + dx, y + dy
            if 0 <= new_x < m and 0 <= new_y < n:
                pos1, pos2 = x * m + y, new_x * m + new_y
                new_board = list(cur)
                new_board[pos1], new_board[pos2] = new_board[pos2], new_board[pos1]
                heapq.heappush(heap, (dis(new_x, new_y), new_x, new_y, ''.join(new_board)))

    return False

typedef tuple<int, int, int, string> boardInfo;
bool validSlidingGame(vector<vector<int> >& board) {
    const int m = board.size(), n = board[0].size();
```

```
    auto dis = [](int x, int y, int z, int p) {return (x - z) * (x - z) + (y - p) * (y - p)
    int x = 0, y = 0;
    string key = "";
    const int dir[4][2] = {{1, 0}, {-1, 0}, {0, 1}, {0, -1}};
    priority_queue<boardInfo> pq;
    unordered_set<string> visited;
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j) {
            if (board[i][j] == 0) {
                x = i, y = j;
            }
            key += to_string(board[i][j]);
        }
    }
    pq.push(make_tuple(-dis(x, y, m - 1, n - 1), x, y, key));
    visited.insert(key);
    while (!pq.empty()) {
        auto tp = pq.top();
        pq.pop();
        string curKey;
        tie(ignore, x, y, curKey) = tp;
//        x = get<1>(tp), y = get<2>(tp);
//        auto curKey = get<3>(tp);
        if (curKey == "123456780") return true;
        for (int i = 0; i < 4; ++i) {
            int newX = x + dir[i][0];
            int newY = y + dir[i][1];
            if (newX >= 0 && newX < m && newY >= 0 && newY < n) {
                swap(curKey[x * m + y], curKey[newX * m + newY]);
                if (!visited.count(curKey)) {
                    pq.push(make_tuple(-dis(newX, newY, m - 1, n - 1), newX, newY, curKey))
                    visited.insert(curKey);
                }
            }
        }
    }
    return false;
}
```

find median from large file of integers

https://instant.1point3acres.com/thread/159344

二分查找

思路就是：先找在INT_MIN和INT_MAX的median（0？），然后读large file of integers，找出比这个数小的个数是否有一半，然后调整二分的边界

```
double findNth(int N, int left, int right) {
    while(left <= right){
        int guess = (left + right) / 2;
        int x, cnt = 0, next = right;
        while(x = readFile()) {
            if(x < guess) ++cnt;
            else next = min(next, x);
        }
        if(cnt == N - 1)
            return next;
        if(cnt < N - 1)
            left = guess;
        else
            right = guess - 1;
    }
    return 0.0;
}

double findMedian() {
    int len = 0;
```

```
    while(readFile()) ++len;
    if(len & 0x1) return findNth(len >> 1, INT_MIN, INT_MAX);
    int x = findNth(len >> 1, INT_MIN, INT_MAX);
    int y = findNth(1 + (len >> 1), x, INT_MAX);
    return double(x + y) / 2;
}
```

Multiply Strings

但是string里面的数可以为负

leetcode 43题，多了个负数的情况（https://leetcode.com/problems/multiply-strings/?tab=Description）

```
#include <iterator>
#include <algorithm>
#include <vector>

typedef vector<int> bigInt;
bigInt make_bigInt(string num) {
    bigInt tmp;
    transform(num.rbegin(), num.rend(), back_inserter(tmp), [](char c) { return c - '0';});
    return tmp;
}
string make_string(bigInt num) {
    string tmp;
    transform(find_if(num.rbegin(), prev(num.rend()), [](int c){ return c != 0;}), num.rend
              back_inserter(tmp), [](int c) { return c + '0';});
    return tmp;
}
bigInt multiply(bigInt const& n1, bigInt const& n2) {
    bigInt n(n1.size() + n2.size());
    for (int i = 0; i < n1.size(); ++i)
        for (int j = 0; j < n2.size(); ++j) {
            n[i + j] += n1[i] * n2[j];
            n[i + j + 1] += n[i + j] / 10;
            n[i + j] %= 10;
        }
    return n;
}

string multiply(string num1, string num2) {
    if(num1.empty() || num2.empty()) return "0";
    bool sign = true;
    if (num1[0] == '-' || num1[0] == '+') {
        if (num1[0] == '-') sign = !sign;
        num1.sustr(1);
    }
    if (num2[0] == '-' || num2[0] == '+') {
         if (num2[0] == '-') sign = !sign;
        num2.sustr(1);
    }
    string ans = make_string(multiply(make_bigInt(num1), make_bigInt(num2)));
    if (ans == "0") return ans;
    return (sign ? "" : "-") + ans;
}
```

CSV PARSER

举个例子:

给定一个CSV文件，格式是 “some_name|some_address|some_phone|some_job”

要求输出Json format “{name:some_name, address:some_addres,phone:some_phone, job:some_job}”

输入内容中有些特殊符号要注意处理

```
vector<string> parseCSV(string s) {
    vector<string> ans;
```

```
bool inQuote = false;
string tmp = "";
for(int i = 0; i < s.length(); ++i) {
    if(inQuote) {
        if(s[i] == '"') {
            if(i == s.length() - 1) {
                ans.push_back(tmp);
                return ans;
            } else if(s[i + 1] == '"') {
                tmp += '"';
                ++i;
            } else {
                ans.push_back(tmp);
                tmp = "";
                inQuote = false;
                i++;
            }
        } else tmp += s[i];
    } else {
        if(s[i] == '"')
            inQuote = true;
        else if(s[i] == ',') {
                ans.push_back(tmp);
                tmp = "";
            } else tmp += s[i];
        }
    }
    if(!tmp.empty()) ans.push_back(tmp);
    return ans;
}

int main() {
    string ss[] = {"John,Smith,john.smith@gmail.com,Los Angeles,1", "\"Alexandra \"\"Alex\"
    for(auto s : ss) {
        auto parsed = parseCSV(s);
        for (int i = 0; i < parsed.size() - 1; ++i)
            cout << parsed[i] << "|";
        cout<<parsed[parsed.size() - 1]<<endl;
    }
    return 0;
}
```

boggle game

（https://instant.1point3acres.com/thread/201695）

这题面试来搞也太变态了，先用trie找出所有单词出现的路径map，然后dfs找出map里不重复的最大的。代码未验证~

```
struct Node {
    bool isWord = false;
    string word;
    Node *next[26];
    Node() {}
};

struct Trie {
    Node *root;
    Trie() {
        root = new Node();
    }

    void buildTrie(const vector<string>& &words) {
        for(auto word : words) {
            Node *cur = root;
            for(auto c : word) {
```

```
                int idx = c - 'a';
                if(!cur->next[idx])
                    cur->next[idx] = new Node();
                cur = cur->next[idx];
            }
            cur->isWord = true;
            cur->word = word;
        }
    }
};

const int dir[4][2] = {{1, 0}, {-1, 0}, {0, 1}, {0, -1}};
void searchBoard(const vector<vector<char> >& board, vector<vector<bool> >& visited, int x,
                 vector<pr>& path, unordered_map<string, vector<vector<pr>> >& mp) {
    if(x < 0 || x > board.size() || y < 0 || y > board[0].size() || visited[x][y])
        return;
    int idx = board[x][y] - 'a';
    if(!root->next[idx]) return;
    root = root->next[idx];
    path.push_back(make_pair(x, y));
    visited[x][y] = true;
    if(root->isWord) mp[root->word].push_back(path);
    for(int i = 0;i < 4; ++i)
        searchBoard(board, visited, x + dir[i][0], y + dir[i][1], root, path, mp);
    visited[x][y] = false;
    path.pop_back();
}
void dfs(int& ans, int cur, const vector<string>& words, vector<vector<bool> >& visited,
         unordered_map<string, vector<vector<pr>> >& mp, int idx) {
    if(idx == words.size()) {
        ans = max(ans, cur);
        return;
    }
    if(ans >= cur + words.size() - idx) return;
    string word = words[idx];
    if(mp.count(word)) {
        for (auto pts : mp[word]) {
            int cnt = 0;
            for (auto pt : pts) {
                if (visited[pt.first][pt.second]) break;
                cnt++;
            }
            if (cnt == word.size()) {
                for (auto pt : pts) visited[pt.first][pt.second] = true;
                dfs(ans, cur + 1, words, visited, mp, idx + 1);
                for (auto pt : pts) visited[pt.first][pt.second] = false;
            }
        }
    }
    dfs(ans, cur, words, visited, mp, idx + 1);
}

int findWords(const vector<vector<char> >& board, const vector<string>& words) {
    if(board.empty() || words.empty()) return 0;
    Trie trie;
    trie.buildTrie(words);
    const auto m = board.size(), n = board[0].size();
    unordered_map<string, vector<vector<pr>> > mp;
    for(int i = 0; i < m; ++i)
        for(int j = 0; j < n; ++j) {
            vector<vector<bool> > visited(m, vector<bool>(n, false));
            vector<pr> path;
            searchBoard(board, visited, i, j, trie.root, path, mp);
        }
    vector<vector<bool> > visited(m, vector<bool>(n, false));
    int ans = 0;
    dfs(ans, 0, words, visited, mp, 0);
```

```
    return ans;
}
```


算法 # 面试

‹ Docker+Ubuntu+jpeg-turbo编译opencv3

记一次mysql update bug排查 ›