Airbnb 面试题

2017 深秋版

一. Disclaim

这些题目是从地里, glassdoor, CSDN, GeekForGeeks等地收集来的。如果有任何建议或问题，请联系coolguy@一亩三分地。欢迎撒米。

参考链接：

- http://www.1point3acres.com/bbs/forum.php?mod=collection&action=view&ctid=277
- http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=282721
- http://community.bittiger.io/topic/626/airbnb%E9%9D%A2%E7%BB%8F-%E4%B8%8A%E7%AF%87
- http://community.bittiger.io/topic/627/airbnb%E9%9D%A2%E7%BB%8F-%E4%B8%8B%E7%AF%87
- https://github.com/jxr041100/system_design

二. Coding/Design Questions

一些问题的 source codes:

- 一些问题的 Python 解法: http://www.1point3acres.com/bbs/thread-220456-1-1.html
- 一些问题的 C++解法: http://zhangpin.info/2017/06/01/airbnb%E9%9D%A2%E8%AF%95%E9%A2%98%E6%B1%87%E6%80%BB/

问题的大致分类：

- 数学问题:
  - Collatz Conjecture
- 数据结构:
  - Design Queue with Limited Size of Array
  - List of List (2D List) Iterator
  - Display Page (Pagination)
  - Calculator
  - Travel Buddy (Buddy List)
  - Trie: File System
  - Trie: Palindrome Pairs
- Bit Operations
  - IP Range to CIDR
- Binary Search:
  - Find Median in Large File of Integers
- 状态机或矩阵变化
  - CSV Parser
  - Simulate Diplomacy
  - Water Land
  - Hilbert Curve
  - Text Justification
  - Regular Experssion

- Time Interval
  - Meeting Time
- Greedy
  - Round Numbers
  - A*: Sliding Game
- DP
  - Maximum Number a Night You Can Accommodate
- Permutation/Combination/Search(BFS/DFS)：
  - Find Case Combinations of a String
  - Menu Combination Sum
  - K Edit Distance
  - Boggle Game
  - Minimum Cost with At Most K Stops
  - String Pyramids Transition Matrix
- 图论
  - Flood Filling: Finding Ocean
  - Topological Sorting： Preference List
  - Minimum Verticaes to Traverse Directed Graph
  - 有向图最短路径: 10 Wizards
  - Union Find: Number of Intersected Rectangles
- Calling AirBnB Internal Services
  - echo TCP client
  - Guess Number

Coding Exercise：

https://www.hackerrank.com/tests/52d6d0953bddb/357c91cd6d50039a74f53a47501e23d7
https://www.hackerrank.com/tests/9e26m70rtfm/cfa8b776a12fb5661055772b4297e8ca

1. Collatz Conjecture

https://en.wikipedia.org/wiki/Collatz_conjecture

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=273149

题目是给你公式，比如偶数的话除 2，奇数的话就变成 3*n+1，对于任何一个正数，数学猜想是最终他会变成 1。每变一步步数加 1，给一个上限，让找出范围内最长步数。

比如 7，变换到 1 是如下顺序：7->22->11->34->17->52->26->13->40->20->10->5->16->8->4->2->1, 总共需要 17 步。

```
Map<Integer, Integer> map = new HashMap<>();

private int findSteps(int num) {
    if (num <= 1) return 1;
    if (map.containsKey(num)) return map.get(num);
    if (num % 2 == 0) return 1 + findSteps(num / 2);
    return 1 + findSteps(3 * num + 1);
```

```
}

public int findLongestSteps(int num) {
    if (num < 1) return 0;

    int res = 0;
    for (int i = 1; i <= num; i++) {
        int t = findSteps(i);
        map.put(i, t);
        res = Math.max(res, t);
    }

    return res;
}
```

Collatz Conjecture

2. Implement Queue with Limited Size of Arrays

https://www.cs.bu.edu/teaching/c/queue/array/types.html

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=279191

Design a Queue with arrayList, 但是有限制条件, arraylist 的长度最多为 10, queue 不限制长度。

简单点儿方法可以把每个 array 最后一个 item 存下一个 array 的指针，这样每个 size 为 10 的 array 可以存 9 个元素。在空间上没有很高效，但是实现起来容易一些。

pop, 先看外边的 list 中最后一个 list 的大小, 如果是 1, 删了当前的 list, 返回, 如果不是, 返回当前 list 最后一个元素
push, 和上边相反, 当前最后一个 list 的大小是 10, 就建个新的.

queue 啊 更简单了, pop 的时候用 iterator, 自动每次都走第一个
push 的时候没区别

楼主可以看下 circular buffer,感觉可以用双指针解决

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=215975

implement 一个 queue，但是要求用 int[]储存，并且一次生成的 int[]长度不可以超过 5。其实这是一个内存分配的模型，每个 int[]可以看成一个 block 的抽象，每次需要加更多元素的时候就要申请多分配 block，remove 的时候就要回收 block。标准做法是用 linkedlist，结果秀逗了用了 tree，面试官居然就让我这么写了，他多追问一下我就想出来了。不过还好过了。

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=294918

其实就是 linkedlist 来解。面试官的思路也是需要自己创建 listnode 里面包含 limied size 的 array。如果这样的话，就比较简单了，就用一长串 linked list 存储就好了，每个位置 node 放 limited size 元素。

```java
public class QueueWithFixedArray {
    private int fixedSize;

    private int count;
    private int head;
    private int tail;
    private List<Object> headList;
    private List<Object> tailList;

    public QueueWithFixedArray(int fixedSize) {
        this.fixedSize = fixedSize;
        this.count = 0;
        this.head = 0;
        this.tail = 0;
        this.headList = new ArrayList<>();
        this.tailList = this.headList;
    }

    public void offer(int num) {
        if (tail == fixedSize - 1) {
            List<Object> newList = new ArrayList<>();
            newList.add(num);
            tailList.add(newList);
            tailList = (List<Object>)tailList.get(tail);
            tail = 0;
        } else {
            tailList.add(num);
        }
        count++;
        tail++;
    }

    public Integer poll() {
        if (count == 0) {
            return null;
        }

        int num = (int)headList.get(head);
        head++;
        count--;

        if (head == fixedSize - 1) {
            List<Object> newList = (List<Object>)headList.get(head);
            headList.clear();
            headList = newList;
            head = 0;
        }

        return num;
    }

    public int size() {
        return count;
    }
}
```

Implement Queue with Fixed size of Arrays

3. List of List (2D List) Iterator

Leetcode 相似问题:

- Leetcode #251 Flatten 2D Vector
- Leetcode #341 Flatten Nested List Iterator

For example, Given 2d vector = [[1,2], [3], [4,5,6]]

By calling next repeatedly until hasNext returns false, the order of elements returned by next should be: [1,2,3,4,5,6].

- boolean hasNext() return true if there is another element in the set
- int next() return the value of the next element in the array
- void remove()
    - remove the last element returned by the iterator.
    - That is, remove the element that the previous next() returned.
    - This method can be called only once per call to next(), otherwise an exception will be thrown.

Removes from the underlying collection the last element returned by this iterator (optional operation). This method can be called only once per call to next(). The behavior of an iterator is unspecified if the underlying collection is modified while the iteration is in progress in any way other than by calling this method.

So the remove() method actually removes the element returned from the next().

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=167190

remove 是 remove 上一个 getNext() get 到的数。就是说不 remove 还没 getNext 的数的意思

举个例子：
如果函数被 call 的顺序是这样的：hasnext() getnext() hasnext() getnext() remove()
假设 data 是 ［［1, 2］ ［3］］
那么返回 true, 1, true, 2, void
data 变为 [[1], [3]]

```
public class Solution implements Iterator<Integer> {
    private Iterator<List<Integer>> rowIter;
    private Iterator<Integer> colIter;

    public Solution(List<List<Integer>> vec2d) {
        rowIter = vec2d.iterator();
        colIter = Collections.emptyIterator();
    }

    @Override
    public Integer next() {
        return colIter.next();
```

```
        }

        @Override
        public boolean hasNext() {
            while ((colIter == null || !colIter.hasNext()) && rowIter.hasNext())
                colIter = rowIter.next().iterator();
            return colIter != null && colIter.hasNext();
        }

        @Override
        public void remove() {
            while (colIter == null && rowIter.hasNext())
                colIter = rowIter.next().iterator();
            if (colIter != null)
                colIter.remove();
        }
    }
```

List of List Iterator

4. Display Page (Pagination)

This is well known as OA2.

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=278720

You're given an array of CSV strings representing search results. Results are sorted by a score initially. A given host may have several listings that show up in these results. Suppose we want to show 12 results per page, but we don't want the same host to dominate the results. Write a function that will reorder the list so that a host shows up at most once on a page if possible, but otherwise preserves the ordering. Your program should return the new array and print out the results in blocks representing the pages.

Input: An array of csv strings, with sort score number of results per page. example:

"host_id,listing_id,score,city"
"1,28,300.1,San Francisco"

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=299158

linkedlist 的 remove 是 O(1), remove 之后，只是改变一下指针就完事了，但是 arraylist 在 remove 之后，需要 copy 剩余的数值，全体移动，虽然这个操作没有体现在 arraylist 的 remove 中，但是 cost 要比 linkedlist 的修改指针慢很多，所以这道题要用 linkedlist 而不是 arraylist。

当你 traverse 这个 linkedlist 的时候，当你 traverse 到这个 entry 并且把他加到 page 里，这个时候完全可以马上删除掉这个 entry，没有必要再用一个 for loop 第二次访问它，所以不存在 remove() 大于 O(1)的可能性。帖子里有人说 remove 最后一个 node 的时间复杂度是 O(n)，这种说法是错误的，应该是 search，或者说 visit 最后一个 node 的时间复杂度是 O(n),但是 remove 永远是 O(1)。

这道题的 worst case 就是像楼主说的那样，所有的 entry 全是一模一样的，比如全是 1,1,1,1,1,1,1 这样假如你的 page size 是 2，时间复杂度就是 O(n*n),最理想的状态是 1,2,3,4,5,6 假如 page size 是 2，时间复杂度是 O(n)。

O(pn - p ^ 2 * m) define p = # of pages, m = page size

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=290914

5
13
1,28,310.6,SF
4,5,204.1,SF
20,7,203.2,Oakland
6,8,202.2,SF
6,10,199.1,SF
1,16,190.4,SF
6,29,185.2,SF
7,20,180.1,SF
6,21,162.1,SF
2,18,161.2,SF
2,30,149.1,SF
3,76,146.2,SF
2,14,141.1,San Jose

以上是一个 Sample 输入，和希望的输出，1,28,100.3,Paris 代表 Host ID, List ID, Points, City. 这是 Airbnb 根据用户搜索条件得出的一些 list，然后我们要分页，第一行的 5 代表每一页最多展示 5 个 list，13 应该是代表有 13 个 List.所以我们是要分成 3 页。规则是：每一页最多展示一个 host 的 list，但是如果再没有其他 host 的 list 可以展示了，就按照原有的顺序填补就可（根据 Points，也就是排名）。应得到的输出：
希望输出：
1,28,310.6,SF
4,5,204.1,SF
20,7,203.2,Oakland
6,8,202.2,SF
7,20,180.1,SF

6,10,199.1,SF
1,16,190.4,SF
2,18,161.2,SF
3,76,146.2,SF
6,29,185.2,SF -- 这时不得不重复了，从原有队列拉出第一个

6,21,162.1,SF
2,30,149.1,SF
2,14,141.1,San Jose

先找不同 host ID 的 post,填到一页里，如果找不到不同 host ID 的 post，就从开头顺序拉取，填满为止。就这样简单，不要想太复杂！

比较直接的做法可以把这些 List 全部放在一个 LinkedList 里，从头到尾扫 另一个 HashSet 记录每一页的重复值，重复就下一个，没重复就 Remove from the list. 如果实在找不到不重的，就从头拉一个即可。

runtime 的话是 O(pageNum * n), space 是 O(n)

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=298030

做题界面讲的非常详细，还举了几个例子，如果你面经读不懂题的话，做题的时候读也是肯定能读懂的。比如 1,2,3,1,1,2,2，每一页要填 5 个数，第一页不重复的只能填 1,2,3，没填满怎么办（怎么办），那就从剩余的数里面依次放进去两个补满，第一页就变成 1,2,3,1,1 就好了。

输入是 String[]，输出是 String[]，每一页中间用个空格 String 隔开。比如{"第一页", " ", "第二页"}

Iterator 的 remove 是完全可以用的。之前有个妹子的帖子里说 remove()不能用，搞得人心惶惶，我也一直担心不能 remove()。后来大概猜想了一下，妹子可能是直接 Arrays.asList(input)完就定义 iterator 了，这种情况下是肯定不能 remove 的（不知道自己表述清楚没有，也不知道自己猜的对不对 =。=）。Hackerrank 编译器本身没问题，不放心的自己先去 test 里面试试。

一共 11 个 test cases，全都公开可以看见。限时一共小时做完。HR 给了 7 天时间完成 OA，但是强调说大多数 candidates 都是 72 小时内做完的。之前准备另一家面试，所以 96 个小时才做，希望没影响。

刚刚接到 follow up 的电话，先让我讲一下思路，然后问了时间复杂度，答 O(numOfPage * n)面试官不满意，面试官说应该是 n + n - m + n - 2m...最后是 O（n^2）。然后问我当 worst case 的情况下，如何优化数据结构

http://www.1point3acres.com/bbs/thread-297778-1-1.html

在地里达成共识的是，在去重的同时保证排名的相对次序，每页用 hashset 记录已出现的，erase 已经列出在 page 上的 id,为了保证 erase 为 O(1),所以要用 linked list 来转换原始记录
当记录里面的 distinct id 已经用完的时候，就重新从记录的 begin()来 traverse，这时候就不得不使用重复的 id 了.

在通常的情况下，duplicate 的 id 不是很多，所以只会在最后一页中出现 duplicate 的 ID，那么 worstcase 呢？假如我们只有 2 个 distinct id,但是有 20 条记录，每页要显示 10 条呢？
正确应该显示的记录是 1212121212 1212121212
第一页，第一次 traverse，hashset 里面记录 12，填了 2 个 id 以后就到记录的 end 了，然后要回到开头，清空 hash set,然后重新开始。。。循环 5 次后，开始下一次

http://blog.csdn.net/whuwangyi/article/details/43600839

这题的思路不难，但是实现起来还是有点难度的。在遍历的时候需要维护一个 **LinkedHashMap** 作为 page 并且完成去重。用 LinkedHashMap 的好处是可以保证所有的 entry 是按插入的顺序排序的，所以仍然可以保证按 score 排序的性质。另外，一旦遇到相同的 host_id，则将其对应的行存到另一个 buffer 里。由于需要变遍历边增减容器里的数据，需要用 ListIterator，并调用 remove 和 add 方法。之前只用过 remove，从来没用过 add。

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=234655

给你一组数据，和每页的容量，要求每页上力求没有重复的 ID 例子:
input：1,2,3,4,1,5,1,2,3,1,3 ; page size : 5
output:
1,2,3,4,5 / 1,2,3,1,1 / 3

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=300128

OA2 就是老面经：display list page 的题目。我看主流的做法是现在流传的。但是之前看了很多人问了如何优化，因为大家都写的一样的所以要踢人。那问题来了怎么优化呢？和朋友讨论过后（朋友过了），得出结果。类似于 merge k sorted list 的写法。时间复杂度是 o(n) + nlogn。就是把相同编号的先排在一起，然后编号内部也是按照 score 从大到小排序的。我觉得我点的够透彻了。大家自行领悟。

```
public List<String> displayPages(List<String> input, int pageSize) {
    List<String> res = new ArrayList<>();
    if (input == null || input.size() == 0) {
        return res;
    }

    List<String> visited = new ArrayList<>();
    Iterator<String> iter = input.iterator();
    boolean reachEnd = false;
    while (iter.hasNext()) {
        String curr = iter.next();
        String hostId = curr.split(",")[0];
        if (!visited.contains(hostId) || reachEnd) {
            res.add(curr);
            visited.add(hostId);
            iter.remove();
        }

        if (visited.size() == pageSize) {
            visited.clear();
            reachEnd = false;
            if (!input.isEmpty()) {
                res.add(" ");
            }
            iter = input.iterator();
        }

        if (!iter.hasNext()) {
            iter = input.iterator();
            reachEnd = true;
        }
    }
```

```
    return res;
}
```

Display Page (解法一)

```
public List<String> displayPages(List<String> input, int pageSize) {
    List<String> res = new ArrayList<>();
    Iterator<String> iter = input.iterator();
    Set<String> set = new HashSet<>();
    boolean reachEnd = false;
    int counter = 0;
    while (iter.hasNext()) {
        String cur = iter.next();
        String id = (cur.split(","))[0];
        if (!set.contains(id) || reachEnd) {
            res.add(cur);
            set.add(id);
            iter.remove();
            counter++;
        }

        if (counter == pageSize) {
            if (!input.isEmpty())
                res.add(" ");
            set.clear();
            counter = 0;
            reachEnd = false;
            iter = input.iterator();
        }

        if (!iter.hasNext()) {
            reachEnd = true;
            iter = input.iterator();
        }
    }

    return res;
}
```

Display Page (解法二)

5. Calculator

Leetcode 相似问题:

- Leetcode #224 Basic Calculator
- Leetcode #227 Basic Calculator II

6. Travel Buddy (Buddy List)

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=214074

每个人都有一些想去的city，如果你想去的city和另一个人想去的city的相似度高于 50%的话你们就是travel buddy，叫你ouput一个list of travel buddy按相似度从高往低排序

我没有用倒排序哎，直接用 hashset 存自己的然后历遍一遍别人的就可以了吧。。面试官也认可的.

如果我的城市是 a1,a2 那和 jack 的相似度就是 2/3，超过了 50%，jack 就是我的 travel buddy。和 tom 的相似度是 1/3，没到 50%，就不是 travel buddy。城市顺序没有关系，最后 output 要相似度高的人排在前面。for 每一个朋友，相似度的定义就是这个朋友和我一样的城市／这个朋友所有的城市。

我的解法就是把自己的城市放在 hashset 里面，遍历一遍所有朋友，如果相似度大于 50%，记下这个朋友已经他的相似度，最后 sort according to 相似度。

followup 的话写了另一个 recommend 城市的 function，好像是你的 travel buddy 想去的城市但你不想去的按某个顺序 output 之类的，具体我忘了。

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=218938

followup是给了一个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buddy

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

就是推荐系统，cosine 求相似度吗？我的是{1 1 1 1 0 0 0}, buddy1 是 (1,1,0,0,1,1,0}, buddy2 是 (1,0,1,1,0,0,1)， you& buddy1 就是 2/4, you & buddy2 就是 3/4，结果等于 0 就是没交集的；然后再根据 cosine 排序就是，把我的放入 hash 中，遍历 buddys。

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=278915

每个伙伴有一列城市，给定一列城市，把伙伴按照 match 的城市排列。

followup: recommend at mos k cities. solution: iterate over the city list of the the ranked buddy list from the first question, output it if it is not already output

```
public class Solution {
    private List<Buddy> buddies;
```

```
    private Set<String> myWishList;

    public Solution(Set<String> myWishList, Map<String, Set<String>>
friendsWishList) {
        this.buddies = new ArrayList<>();
        this.myWishList = myWishList;
        for (String name : friendsWishList.keySet()) {
            Set<String> wishList = friendsWishList.get(name);
            Set<String> intersection = new HashSet<>(wishList);
            intersection.retainAll(myWishList);
            int similarity = intersection.size();
            if (similarity >= wishList.size() / 2) {
                buddies.add(new Buddy(name, similarity, wishList));
            }
        }
    }

    public List<Buddy> getSortedBuddies() {
        Collections.sort(buddies);
        List<Buddy> res = new ArrayList<>(buddies);
        return res;
    }

    public List<String> recommendCities(int k) {
        List<String> res = new ArrayList<>();
        List<Buddy> buddies = getSortedBuddies();

        int i = 0;
        while (k > 0 && i < buddies.size()) {
            Set<String> diff = new HashSet<>(buddies.get(i).wishList);
            diff.removeAll(myWishList);
            if (diff.size() <= k) {
                res.addAll(diff);
                k -= diff.size();
                i++;
            } else {
                Iterator<String> it = diff.iterator();
                while (k > 0) {
                    res.add(it.next());
                    k--;
                }
            }
        }

        return res;
    }

    class Buddy implements Comparable<Buddy> {
        String name;
        int similarity;
        Set<String> wishList;

        Buddy(String name, int similarity, Set<String> wishList) {
            this.name = name;
            this.similarity = similarity;
            this.wishList = wishList;
        }

        @Override
        public int compareTo(Buddy that) {
```

```
            return that.similarity - this.similarity;
        }
    }
}
```

Travel Buddy

7. File System

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=215056

开始的时候是写两个 function， create 和 get
create("/a",1)
get("/a") //得到 1
create("/a/b",2)
get("/a/b") //得到 2
create("/c/d",1) //Error，因为它的上一级“/c”并不存在
get("/c") //Error,因为“/c”不存在

follow up 是写一个 watch 函数，比如 watch("/a",new Runnable(){System.out.println("helloword");}) 后，每当 create("/a/b"，1) 等在/a 之下的目录不产生 error 的话，都会执行绑在“/a”上的 callback 函数

比如 watch("/a",System.out.println("yes"))
watch("/a/b",System.out.println("no"))

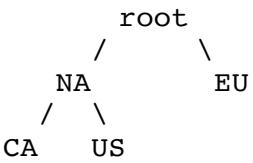
当 create("/a/b/c",1)时，两个 callback 函数都会被触发，会 output yes 和 no

我对 java 的 callback 并不是很熟悉，面试官小哥很好地给了 trigger callback 的接口，因为之前在地里看到说没必要建 trie，所以直接 hashmap 解决，但处理字符串找它的上一层目录处理“/”比较容易出 bug，要注意判断根目录的情况 .

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=215981

三个 method:create(path, value), set_value(path, value), get_value(path)

让你实现一个长成这样的tree:



其中root是没有name和value，剩下的每个点都有name和value.

- create(path，value):给你一个path，比如“NA/MX”，和value，比如“3”。那么你就在NA下面创建一个点叫MX，值是3。

- set_value(path, value): 给你一个path，找到path的叶子，然后set value，如果 叶子不存在，返回false;
- get_value(path): 给你一个path，返回叶子的值，没有叶子的话返回NULL。

```java
public class Solution {
    Map<String, Integer> pathMap;
    Map<String, Runnable> callbackMap;

    public Solution() {
        this.pathMap = new HashMap<>();
        this.callbackMap = new HashMap<>();
        this.pathMap.put("", 0);
    }

    public boolean create(String path, int value) {
        if (pathMap.containsKey(path)) {
            return false;
        }

        int lastSlashIndex = path.lastIndexOf("/");
        if (!pathMap.containsKey(path.substring(0, lastSlashIndex))) {
            return false;
        }

        pathMap.put(path, value);
        return true;
    }

    public boolean set(String path, int value) {
        if (!pathMap.containsKey(path)) {
            return false;
        }

        pathMap.put(path, value);

        // Trigger callbacks
        String curPath = path;
        while (curPath.length() > 0) {
            if (callbackMap.containsKey(curPath)) {
                callbackMap.get(curPath).run();
            }
            int lastSlashIndex = path.lastIndexOf("/");
            curPath = curPath.substring(0, lastSlashIndex);
        }

        return true;
    }

    public Integer get(String path) {
        return pathMap.get(path);
    }

    public boolean watch(String path, Runnable callback) {
        if (!pathMap.containsKey(path)) {
            return false;
        }
```

```
            callbackMap.put(path, callback);
            return true;
        }
    }
```

File System

8. Palindrome Pairs

Leetcode 相似问题: Leetcode #336 Palindrom Pairs

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=143760

就是给你一堆词，找出所有的 pair 拼起来能组成 palindrome，比如 ["aabc", "cbaa"] 这种。

正确的普通做法是，把每个词的倒序放进一个hashset，然后枚举每词word ：

- 枚举所有前缀，如果前缀在hashset里 AND 当前后缀是回文，那么拼起来(word是pair的第一个)
- 枚举所有后缀，如果后缀在hashset里 AND 当前前缀是回文，那么拼起来(word是pair的第二个)

更好的做法是用trie加速枚举，这样一旦trie里面找不到了就不用枚举更长的前/后缀了。但是时间复杂度是一样的，是O(NL^2)，L是平均长度.

Leetcode 上有很好的解法。

9. Find Median in Large File of Integers

Leetcode 相似问题 Leetcode #295 Find Median from Data Stream LC 这道题是 stream input,是没有办法做 binary search 的。

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=171676

看到题目有点懵。。。乍一看是 Divide Conquer。可是 merge 的时候不太容易。
想一想 LC 的 median from stream。还是需要 heaps 来储存。。。看来不行。
百般尝试之后，小哥开始 push。说可以看看 median 的 upper bound 和 lower bound。
这两个词一出不就是 binary search 了么。。。讲出了自己的想法，聊了一下复杂度。。。编了一会。。。编出了 bug。。。找到的 median 不在数组里面。。。最后发现了问题。。。想解决已经来不及了。。。

其实有很多对大 file 的处理方法。 在 interview 当中，华人哥哥给出的 hint 是用 binary search 首先， 我们知道对任何大的 file median of any int will between INT_MIN and INT_MAX。所以我们知道了 upper bound 和 lower bound。我们猜一下 median might be "guess = lower+(upper-lower)/2"。之后我们可以验证对不对。就是扫一遍这个 file，看看是不是有一半的 element 确实小于这个数字。如果是的话，这里注意一定要返回 smallest element in the file that is larger than the guess。如果有超过一半的数据小于这个 guess，可想而知用 binary search 的方法，下一步就是移动上线到 guess-1. 反之移动下线。对吧。那么这个算法最多需要 scan 32 次 fille 对不？这个数字当时我有点含糊。但是现在想想应该是对的。

谢谢楼主分享。 想请问一下楼主，等不能这样想。 固定一个 guess 之后扫数组时，记录两个值，smaller 是小于 guess 的数目，larger 是大于 guess 的数目。 如果 smaller 等于 larger， 那就直接返回 guess， 如果 smaller < larger， 那就 start = guess， 如果 smaller > larger,那就 end = mid。 这两种轻卡 ugndou 不排除 guess 依旧是 median 的可能性。 最后结束条件是 start + 1 < end. 然后再根据 start 的数目多还是 end 这个数的数目多进行判断。

最后的判断应该是如果小于等于 start 的数目和大于等于 end 的数目哪个多。 一样多的话返回他俩的平均值。

```
public class Solution {
    private long search(int[] nums, int k, long left, long right) {
        if (left >= right) {
            return left;
        }

        long res = left;
        long guess = left + (right - left) / 2;
        int count = 0;
        for (int num : nums) {
            if (num <= guess) {
                count++;
                res = Math.max(res, num);
            }
        }

        if (count == k) {
            return res;
        } else if (count < k) {
            return search(nums, k, Math.max(res + 1, guess), right);
        } else {
            return search(nums, k, left, res);
        }
    }

    public double findMedian(int[] nums) {
        int len = 0;
        for (int num : nums) {
            len++;
        }

        if (len % 2 == 1) {
            return (double) search(nums, len / 2 + 1, Integer.MIN_VALUE,
Integer.MAX_VALUE);
        } else {
            return (double) (search(nums, len / 2, Integer.MIN_VALUE,
```

```
    Integer.MAX_VALUE) +
                        search(nums, len / 2 + 1, Integer.MIN_VALUE,
    Integer.MAX_VALUE)) / 2;
            }
        }
    }
```

Find Median from Large File of Integers

10. IP Range to CIDR

这个是背景介绍： 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/questio ... nge-to-cidr-in-java, 叶泰航给的那个

表达是: ip地址/掩码位数 表示一个区间

比如 0.0.0.8 / 30 就是以0.0.0.8为准，前30位不能变--》0.0.0.(0000 10 00)-- 0.0.0.(0000 10 11)

然后给你一个起始ip，和数量。用最少的cidr表示这个区间 Examples:

Input: 1.1.1.0 4 Output: 1.1.1.0/30

Input: 1.1.1.1 4 Output: 1.1.1.1/32 1.1.1.2/31 1.1.1.4/32

Given a string ip and number n, print all cidr addresses that cover that range - This is one of the problems that is out of left field.

Given an IPv4 IP address p and an integer n, return a list of CIDR strings that most succinctly represents the range of IP addresses from p to (p + n).

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=282139

题目：给一个 IP 地址范围，比如"10.10.1.1" - "10.12.235.12"，转换成其 CIDR 表示。CIDR（classless inter-domain routing）的定义可以自行 google，这题的主要意思就是转换成一系列的 CIDR 地址，刚好覆盖给出的 IP 地址范围（不多不少）。

算法：
从一个简单的例子开始：
比如 IP 地址范围是 0.0.0.111 - 0.0.0.120，对应的二进制表示如下
1101111 - 111
1110000 - 112
...
1110111 - 119
1111000 - 120

显然 0.0.0.(11000000)/24 可以覆盖整个地址区间。但是这个 CIDR 不是一个“合格”的 CIDR，因为它的范围太大了，超出了输入的 IP 地址范围，所以我们要进一步缩小。

缩小的办法是把这个最大的 CIDR 对半分得到两个 CIDR：
- 0.0.0.(11000000)/24
- 0.0.0.(11100000)/25

我们可以把输入的 IP 地址范围分到这两个 CIDR 中去，然后再递归求解，直到 CIDR 恰好覆盖 IP 区间，则得到合格的 CIDR。

```
public class Solution {
    private long ipToLong(String strIP) {
        long[] ip = new long[4];
        String[] ipSec = strIP.split("\\.");
        for (int k = 0; k < 4; k++) {
            ip[k] = Long.valueOf(ipSec[k]);
        }

        return (ip[0] << 24) + (ip[1] << 16) + (ip[2] << 8) + ip[3];
    }

    private String longToIP(long longIP) {
        StringBuffer sb = new StringBuffer("");
        sb.append(String.valueOf(longIP >>> 24));
        sb.append(".");
        sb.append(String.valueOf((longIP & 0x00FFFFFF) >>> 16));
        sb.append(".");
        sb.append(String.valueOf((longIP & 0x0000FFFF) >>> 8));
        sb.append(".");
        sb.append(String.valueOf(longIP & 0x000000FF));

        return sb.toString();
    }

    public List<String> ipRange2Cidr(String startIp, int range) {
        // check parameters
        String a = "";
        long start = ipToLong(startIp);
        long end = start + range - 1;
        List<String> res = new ArrayList<>();
        while (start <= end) {
            // identify the location of first 1's from lower bit to higher
bit of start IP
            // e.g. 00000001.00000001.00000001.01101100, return 4 (100)
            long locOfFirstOne = start & (-start);
            int curMask = 32 - (int) (Math.log(locOfFirstOne) /
Math.log(2));

            // calculate how many IP addresses between the start and end
            // e.g. between 1.1.1.111 and 1.1.1.120, there are 10 IP address
            // 3 bits to represent 8 IPs, from 1.1.1.112 to 1.1.1.119 (119 -
112 + 1 = 8)
            double currRange = Math.log(end - start + 1) / Math.log(2);
            int currRangeMask = 32 - (int) Math.floor(currRange);

            // why max?
            // if the currRangeMask is larger than curMask
            // which means the numbers of IPs from start to end is smaller
```

```
than mask range
            // so we can't use as many as bits we want to mask the start IP
to avoid exceed the end IP
            // Otherwise, if currRangeMask is smaller than curMask, which
means number of IPs is larger than mask range
            // in this case we can use curMask to mask as many as IPs from
start we want.
            curMask = Math.max(currRangeMask, curMask);

            // Add to results
            String ip = longToIP(start);
            res.add(ip + "/" + curMask);
            // We have already included 2^(32 - curMask) numbers of IP into
result
            // So the next roundUp start must insert that number
            start += Math.pow(2, (32 - curMask));
        }
        return res;
    }
}
```

IP Range to CIDR

11. CSV parser

Parse an escaped string into csvformat

Input: csvformat

John,Smith,john.smith@gmail.com,Los Angeles,1

Jane,Roberts,janer@msn.com,"San Francisco, CA",0

"Alexandra ""Alex""",Menendez,alex.menendez@gmail.com,Miami,1 """Alexandra Alex"""

Output: escaped string

John|Smith|john.smith@gmail.com|Los Angeles|1

Jane|Roberts|janer@msn.com|San Francisco, CA|0

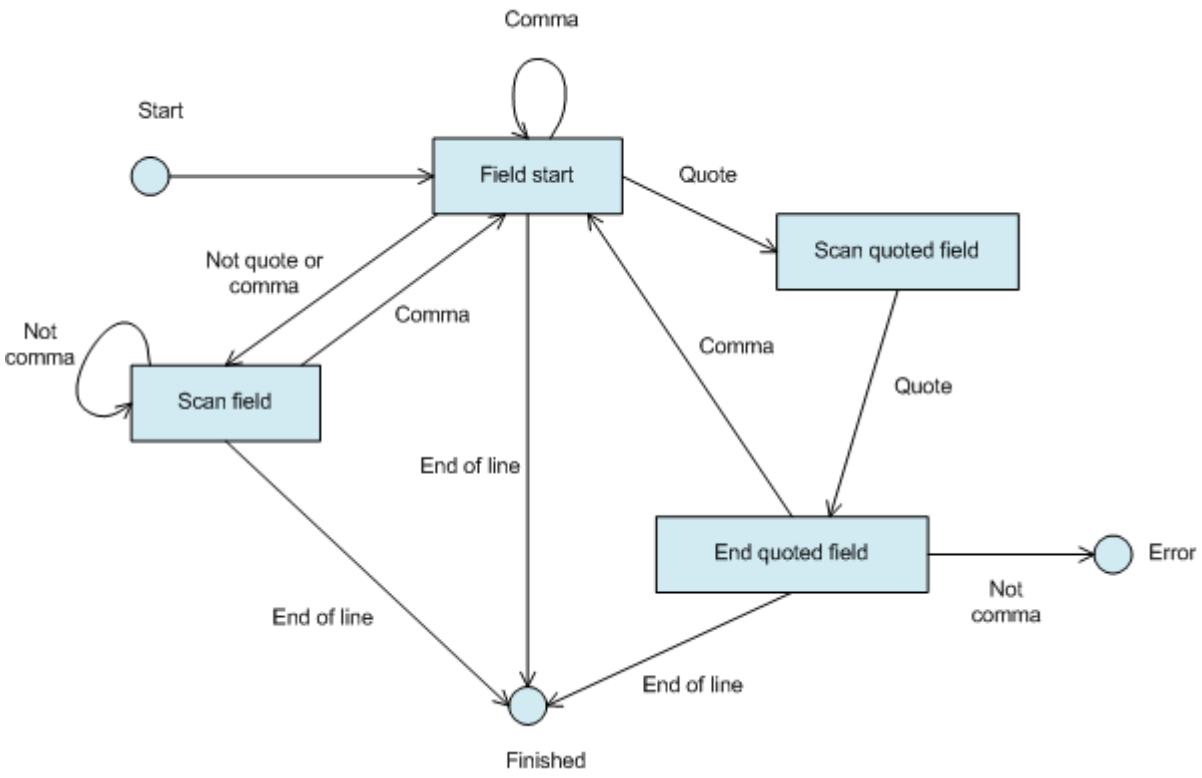
Alexandra "Alex"|Menendez|alex.menendez@gmail.com|Miami|1 "Alexandra Alex"

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=154363

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=131724

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=140445

可以考虑画一下状态图：




```java
public String parseCSV(String str) {
    List<String> res = new ArrayList<>();
    boolean inQuote = false;
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < str.length(); i++) {
        if (inQuote) {
            if (str.charAt(i) == '\"') {
                if (i < str.length() - 1 && str.charAt(i + 1) == '\"') {
                    sb.append("\"");
                    i++;
                } else {
                    inQuote = false;
                }
            } else {
                sb.append(str.charAt(i));
            }
        } else {
            if (str.charAt(i) == '\"') {
                inQuote = true;
            } else if (str.charAt(i) == ',') {
                res.add(sb.toString());
                sb.setLength(0);
            } else {
                sb.append(str.charAt(i));
            }
        }
    }

    if (sb.length() > 0) {
        res.add(sb.toString());
    }
```

```
        return String.join("|", res);
    }
```

CSV Parser

12. Text Justification

Leetcode 相似问题: Leetcode #68 Text Justification

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=299301

Given a JSON input with a list of strings, e.g.. more info on 1point3acres.com

```
  {
    'text': 'first word',
    'text': 'my second sentence',
    'text': 'now it's third',
  }.
```

要求 print 出来这样的形式（格式崩了，大家意会即可，左右间距以最长 str 为准）

```
+----------------------+
|first word            |
+----------------------+
|my second sentence    |
+----------------------+
|now it's third        |
+----------------------+
```

和小哥大致讨论了下，秒了。

然后小哥来了个 follow up, 说 input JSON 里再加一个 width, 要求左右间距以 width 为准，左缩进，print 出来大概这样：

```
+----------------+
|first word      |
+----------------+
|my second       |
|sentence        |
+----------------+
|now it's third  |
+----------------+
```

又秒了。

之后小哥说奥森！但还有个 follow up，这回你不用写 code 了，大致说下想法就好：input JSON 为 list of lists, 一级 list 表示一行，二级 list 表示每行中的一列，每列规定 width（相当于每个 string 都会有自己的 width）, print 出来大概这样：

```
+--------------+----------+
|first word    |my        |
|              |second    |
|              |sentence  |
+--------------+----------+
|now it's third           |
+-------------------------+
```

大致讲了下想法，小哥说破费*2，然后时间到了欢乐得 say 了 bye。

之后一个 followup 说的是如果一个单词一行装不下怎么办，面试官提示说可以用'-';比如[bbbbbb, bbb], n = 3;
bb-
bb-
bb-
bb-
b__
空格用‘_’代替；可以做成预处理[bb-, bb-, bb-, bb-, b]之后用原来的代码，我和面试官说了思路但是代码没有写完，面试官安慰说我知道你的意思了，followup 没关系。

Leetcode 上有很好的解法。

13. Regular Experssion

Leetcode 相似问题:

- Leetcode #44 Wild Card Matching
- Leetcode #10 Regular Experssion Matching

https://www.mitbbs.com/article_t/JobHunting/33067863.html

是leetcode上面wildcard和regex match的综合，加了+号，大家有啥好方法吗？

- *: 0个或多个之前字符
- .: 任何字符
- +:1个或多个之前字符

另外一点不太清楚是这个加号的作用，如果是 aa 和 +a 这种算是match到了吗？就是加号前面是空。

应该是 a+吧

https://www.glassdoor.com/Interview/1-find-all-the-combinations-of-a-string-in-lowercase-and-uppercase-For-example-string-ab-and-gt-ab-Ab-aB-AB-QTN_582954.htm

Implement a simple regex parser which, given a string and a pattern, returns a boolean indicating whether the input matches the pattern. By simple, we mean that the regex can only contain special character: * (star), . (dot), + (plus). The star means what you'd expect, that there will be zero or more of previous character in that place in the pattern. The dot means any character for that position. The plus

means one or more of previous character in that place in the pattern.

```
public boolean regMatch(String source, String pattern) {
    if (pattern.length() == 0) return source.length() == 0;
    if (pattern.length() == 1) {
        if (source.length() > 1 || source.length() == 0) return false;
        return source.charAt(0) == pattern.charAt(0);
    }

    if (source.length() != 0 && (pattern.charAt(0) == '.' || pattern.charAt(0)
== source.charAt(0))) {
        if (pattern.charAt(1) == '*') {
            return regMatch(source.substring(1), pattern) || regMatch(source,
pattern.substring(2));
        } else if (pattern.charAt(1) == '+') {
            return regMatch(source.substring(1), pattern.substring(2)) ||
regMatch(source.substring(1), pattern.substring(2));
        } else {
            return regMatch(source.substring(1), pattern.substring(1));
        }
    }
    return pattern.charAt(1) == '*' && regMatch(source, pattern.substring(2));
}
```

Regular Experssion

14. Water Drop/ Water Land

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=218383

往一个int array 代表海拔的格子里倒水，打印出倒水后的图， 输入：int[] 海拔， int 水数量， int 倒得位置。

Example:

int[] 海拔 {5,4,2,1,2,3,2,1,0,1,2,4}

```
+
++          +
++   +      ++
+++ +++     ++
++++++++ +++
++++++++++++
012345678901
```

水数量8， 倒在位置5 ->

```
+
```

```
++          +
++www+      ++
+++w+++www++
++++++++w+++
++++++++++++
012345678901
```

请问水滴下来。是忘左边走还是右边走呢。还是说我可以假设所有的水滴掉下来都优先往左边走。直到遇见了不能存储的情况才考虑右边？
都可以 我假设往左先面试官说可以的

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=279000

这个题目比较open－ended。关键是和interviewer讲清楚你的assumption。

我的assumption：

- handle the water drop by drop。 there are infinitely high walls on the left and right
- 水先向左走，走到不能走为止。call it leftMost
- 如果leftmost的水比开始点低，leftMost水+1，done
- 如果leftmost的水不比开始点低，水向右走，走到不能走为止。call it rightMost
- 如果rightmost的水比开始点低，rightMost水+1，done
- 如果rightmost的水不比开始点低，leftMost水+1，done

这道题是这样，很多东西都是很面试官确认出来，像我和他讨论出的结果就有：水滴优先往左流，没地流再往右流，也没地了就在当前位置涨；两边有无限高的墙挡着；水滴是一滴一滴的，不能分为小数，所以一滴水会一直往左走到尽头（其实是不符合物理规则的但理他呢。。）

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=229065

输入有一个int数组代表地表的高度，一个下雨的位置，水的总量，要求print出一 个图(+表示土地，W表示水)显示下雨后水的积存情况。

类似

```
+
+W+
+++
```

char[][] simulateWaterDrop(int[] input, int pos, int volume). 需要一滴一滴模拟，每滴水要找左右两边的peak，有几种case，分开考虑一下，完成一滴水之后，再更新高度。

```
public void pourWater(int[] heights, int water, int location) {
    int[] waters = new int[heights.length];
    int pourLocation;

    while (water > 0) {
        int left = location - 1;
        while (left >= 0) {
            if (heights[left] + waters[left] > heights[left + 1] + waters[left
+ 1]) {
                break;
            }
            left--;
        }
        if (heights[left + 1] + waters[left + 1] < heights[location] +
waters[location]) {
            pourLocation = left + 1;
            waters[pourLocation]++;
            water--;
            continue;
        }

        int right = location + 1;
        while (right < heights.length) {
            if (heights[right] + waters[right] > heights[right - 1] +
waters[right - 1]) {
                break;
            }
            right++;
        }
        if (heights[right - 1] + waters[right - 1] < heights[location] +
waters[location]) {
            pourLocation = right - 1;
            waters[pourLocation]++;
            water--;
            continue;
        }

        pourLocation = location;
        waters[pourLocation]++;
        water--;
    }

    print(heights, waters);
}

private void print(int[] heights, int[] waters) {
    int n = heights.length;

    int maxHeight = 0;
    for (int i = 0; i < n; i++) {
        maxHeight = Math.max(maxHeight, heights[i] + waters[i]);
    }

    for (int height = maxHeight; height >= 0; height--) {
        for (int i = 0; i < n; i++) {
            if (height <= heights[i]) {
```

```
                System.out.print("+");
            } else if (height > heights[i] && height <= heights[i] + waters[i])
{
                System.out.print("W");
            } else {
                System.out.print(" ");
            }
        }
        System.out.println();
    }
    System.out.println();
}
```

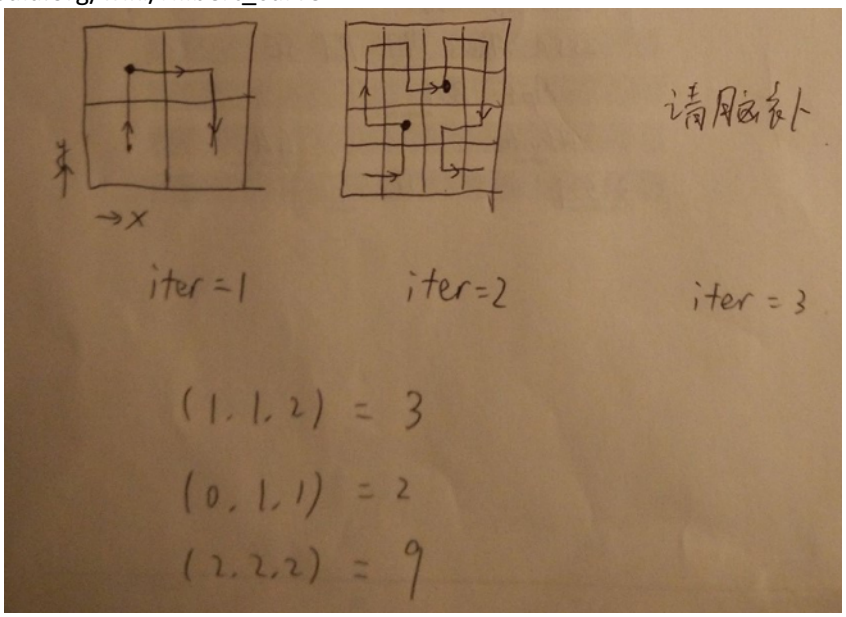
Water Drop (both sides have no limited walls)

15. Hilbert Curve

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=146537

每次给你一个(x, y, iter)，问你在 iter 这张图中在(x, y)坐标的点是第几个？
解法：从大往小左，逐渐细化。每次先算出当前点在当前 iter 是在第几象限，先加上前面那些跳过去的象限里的点。然后找到这个点在这个象限的相对坐标新(x,y)，但是还不够！对于三四象限的点，因为方向变了，需要做镜面映射，把(x,y)映射成(y,x) (第三象限) 或 (M-y, M-x) (第四象限)，M 是象限的长宽。

https://en.wikipedia.org/wiki/Hilbert_curve



http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=220456

Analyze! 发现每一个 level 的图会是 4 个 level-1 的变化得来的（一样，或者根据对角线对称）。

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=229065

输入是坐标 x，y，还是第几 iteration，输出是从原点走多少步达到这个点。
结合 code，先把 iteration 1，2，3 画出来，就知道是干什么的了，对了，如果可能，也可以试试反着来一遍。输入是第几 iteration，steps，返回坐标

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=218196

顺着那个递归的思路就可以做的。代码并不难。思路就是每个 generation 是由前一个 generation 组成的，有的象限会有翻转。每次进去判断在那个象限然后把相应的坐标翻转一下继续递归就行了

```
public int hilbertCurve(int x, int y, int iter) {
    if (iter == 0) return 1;
    int len = 1 << (iter - 1);
    int num = 1 << (2 * (iter - 1));

    if (x >= len && y >= len) {
        // 3 Shape is identical with previous iteration
        return 2 * num + hilbertCurve(x - len, y - len, iter - 1);
    } else if (x < len && y >= len) {
        // 2 Shape is identical with previous iteration
        return num + hilbertCurve(x, y - len, iter - 1);
    } else if (x < len && y < len) {
        // 1 Clock-wise rotate 90
        return hilbertCurve(y, x, iter - 1);
    } else {
        // 4 Anti-Clockwise rotate 90
        return 3 * num + hilbertCurve(len - y - 1, 2 * len - x - 1, iter - 1);
    }
}
```

Hibert Curve

16. Simulate Diplomacy

Also known as OA 4

Leetcode 相似问题: Leetcode #289 Game of Life

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=298214

外交游戏可以参考 http://www.backstabbr.com/rules 这个链接里的规则，当然，题目做了简化。

输入是 "军队名 地点 1 action (地点 2／军队名)",比如“A Paris move London“，action 有 move, support, hold 三种，move 是从地点 1 到地点 2， hold 是留在地点 1，support 也留在原地,support 的对象是另一支军队.
主要需要计算的是 move 引起的军队地点变化和属性变化。move 的目的地如果别的军队，就会发生战斗，战斗的结果取决于 strength 大小，如果 strength 相等，那么都挂，不等的时候 strength 大的胜，别的挂。
strength 默认初始都一样，可以通过别人的 support 提升，但是 support 的军队的驻地如果被 attack，support 是无效的。

总体理解题意以后算法是蛮直接的，没什么技巧，祝大家都好运！

所以这题两个 strength 相等是都挂，但原游戏其实是都撤退?

补充一下 followup: 白人小哥， walk through the solution, 分析优化，然后提问题，这题没太多可说的，很快问完，聊的很愉快

17. Meeting Time

Leetcode 相似问题: Leetcode 252/253 Meeting Rooms I/II

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=146537

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。
举例:
[[1, 3], [6, 7]], [[2, 4]], [[2, 3], [9, 12]]
返回
[[4, 6], [7, 9]]

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。

follow up: 求不少于 k 个员工空闲的时间段（改一下 check count 的条件就可以了）

```
public class Solution {
    class Point implements Comparable<Point> {
        int time;
        boolean isStart;

        Point(int time, boolean isStart) {
            this.time = time;
            this.isStart = isStart;
        }

        @Override
        public int compareTo(Point that) {
            if (this.time != that.time || this.isStart == that.isStart) {
                return this.time - that.time;
            } else {
                return this.isStart ? -1 : 1;
            }
        }
    }

    public List<Interval> getAvailableIntervals(List<List<Interval>>
intervals, int k) {
        List<Interval> res = new ArrayList<>();
        List<Point> points = new ArrayList<>();

        for (List<Interval> intervalList : intervals) {
            for (Interval interval : intervalList) {
```

```
                points.add(new Point(interval.start, true));
                points.add(new Point(interval.end, false));
            }
        }
        Collections.sort(points);

        int count = 0;
        Integer availableStart = null;
        for (int i = 0; i < points.size(); i++) {
            Point point = points.get(i);
            if (point.isStart) {
                count++;
                if (availableStart == null && i == 0 && count <=
intervals.size() - k) {
                    availableStart = point.time;
                } else if (availableStart != null && count ==
intervals.size() - k + 1) {
                    res.add(new Interval(availableStart, point.time));
                    availableStart = null;
                }
            } else {
                count--;
                if (count == intervals.size() - k && i < points.size() - 1)
{
                    availableStart = point.time;
                } else if (availableStart != null && i == points.size() - 1
&& count <= intervals.size() - k) {
                    res.add(new Interval(availableStart, point.time));
                    availableStart = null;
                }
            }
        }

        return res;
    }
}
```

Meeting Time

18. Round Prices

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=146539

公司 list 价格分成好几个部分，但是都是整数，如果在美金是整数，到了欧洲的网页显示汇率转换之后就变成了 floating point，然后要 round 成整数，但是全部加起来 round，和单独 round 再加起来，结果会不一样

base price 100 => 131.13 => 131
cleaning fee 20 => 26.23 => 26
service fee 10 => 13.54 => 14
tax 5 => 6.5 => 7
=> 177.4E => 178E
sum 135$ => 178.93E => 179E

那么问题就来了，给个 input list of floating points, 要求 output list of integers, 满足以下两个 constraint， 就是和跟 Round(x1+x2+... +xn)的结果一样，但是 minimize output 和 input 的绝对值差

之和
#Input: A = [x1, x2, ..., xn]
Sum T = Round(x1+x2+... +xn) ； 178.93E => 179-google 1point3acres
Output: B = [y1, y2, ..., yn]

Constraint #1: y1+y2+...+yn = T
Constraint #2: minimize sum(abs(diff(xi - yi)))

举例
A = [1.2, 2.3, 3.4]
Round(1.2 + 2.3 + 3.4) = 6.9 => 7
1 + 2 + 3 => 6
0.2 + 0.3 + 0.4 = 1.0

1 + 3 + 3 => 7
0.2 + 0.7 + 0.4 = 1.3

1 + 2 + 4 => 7
0.2 + 0.3 + 0.6 = 1.1
所以[1,2,4]比[1,3,3]要好

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。

做法是这样：比如[1.2, 2.5, 3.6, 4.0]
建个新数列是原来数字的 int, arr = [1, 2, 3, 4]
然后算需要补多少个数字才能到需要的 sum. round(1.2+2.5+3.6+4.0) - (1+2+3+4) = 1， 补 1 个数字就好
现在把原数组和 arr 的差值排列一下。 差值是[0.2, 0.5, 0.6, 0]. 我们要从差值最大的数补起，我把 index 排列了一下就是[2, 1, 0, 3]，按这个顺序补数字就好。
因为我们只需要补一个数字，先补 index = 2 的, 所以最后的就结果是 [1, 2, 4, 4]。

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=156061

Given an array of numbers A = [x1, x2, ..., xn] and T = Round(x1+x2+... +xn). We want to find a way to round each element in A such that after rounding we get a new array B = [y1, y2, ..., yn] such that y1+y2+...+yn = T where yi = Floor(xi) or Ceil(xi), ceiling or floor of xi.

We also want to minimize sum |x_i-y_i|

贪心应该就可以解决
先对每个 element 进行 round（向最小的那边进行 round ceil 或者 floor）
然后把值加起来 如果和 T 有个差值 比 T 大或者小 X

如果比 T 大（小同理），我们可以对那些取 ceil 的 element,

按照这个 element 和它 floor 的差值从小到大排个序
从小到大取 X 个 elements 取 floor 就可以得到 T。

先将所有的 x 取 floor， 然后 T - sum（floor（x）） 得到多少个 x 需要 ceil
按照小数部分将数组排序，从大到小 ceil，其他的 floor。 最后就是想要的结果。

如果用 selection sort 的方法 可以达到 O(n)

```
public class Solution {
    public int[] roundUp(double[] arr) {
        int n = arr.length;
        NumWithDiff[] arrWithDiff = new NumWithDiff[n];
        double sum = 0.0;
        int floorSum = 0;
        for (int i = 0; i < n; i++) {
            int floor = (int) arr[i];
            int ceil = floor;
            if (floor < arr[i]) ceil++;
            floorSum += floor;
            sum += arr[i];
            arrWithDiff[i] = new NumWithDiff(ceil, ceil - arr[i]);
        }

        int num = (int) Math.round(sum);
        int diff = num - floorSum;
        Arrays.sort(arrWithDiff, new Comparator<NumWithDiff>() {
            @Override
            public int compare(NumWithDiff n1, NumWithDiff n2) {
                if (n1.diffWithCeil <= n2.diffWithCeil) return -1;
                else return 1;
            }
        });
        // Arrays.sort(arrWithDiff, (a, b) ->
(Double.compare(b.diffWithCeil, a.diffWithCeil)));

        int[] res = new int[n];
        int i = 0;
        for (; i < diff; i++) {
            res[i] = arrWithDiff[i].num; // 这些放 ceil
        }
        for (; i < n; i++) {
            res[i] = arrWithDiff[i].num - 1; // 剩下的只放 floor
        }
        return res;
    }

    class NumWithDiff {
        int num;
        double diffWithCeil;

        public NumWithDiff(int n, double c) {
            this.num = n;
            this.diffWithCeil = c;
        }
    }
}
```

Round Prices

19. Sliding Game (8 Puzzles)

You're given a 3x3 board of a tile puzzle, with 8 tiles numbered 1 to 8, and an empty spot. You can move any tile adjacent to the empty spot, to the empty spot, creating an empty spot where the tile originally was. The goal is to find a series of moves that will solve the board, i.e. get [[1, 2, 3], [4, 5, 6], [7, 8, -]…

http://www.1point3acres.com/bbs/thread-203769-1-1.html

实现一个sliding game，就是以前小时候玩的那种九宫格，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.

https://en.wikipedia.org/wiki/15_puzzle

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=291630

8 puzzle，面试官也没要求最短路径，只问 Ture False。
直接用了暴力 dfs。。。没想到搜索空间巨大，stack overflow 了。。。。当前也没想到。。。
估计这题得 bfs + pq 。大家小心这题。
哎😔 伤心啊，感觉前面面的都还行，到这大意了.

基本的深搜和广搜的性能都非常差，一般使用 A* 或者 IDA*算法

```
public class Solution {
    private final int[][] dirs = {{1, 0}, {0, 1}, {-1, 0}, {0, -1}};
    private int[][] matrix;
    private int m;
    private int n;
    private int originX;
    private int originY;
    private String recovered;

    public Solution(int[][] matrix) {
        this.matrix = matrix;
        this.m = matrix.length;
        this.n = matrix[0].length;
        int[][] recoveredMatrix = new int[m][n];
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                if (matrix[i][j] == 0) {
                    this.originX = i;
                    this.originY = j;
                }

                recoveredMatrix[i][j] = i * n + j;
            }
        }
        this.recovered = getMatrixString(recoveredMatrix);
    }
```

```
    public boolean canSolve() {
        Queue<int[]> items = new LinkedList<>();
        Queue<String> matrix = new LinkedList<>();
        Set<String> visited = new HashSet<>();

        String stringMatrix = getMatrixString(this.matrix.clone());
        items.offer(new int[]{originX, originY});
        matrix.offer(stringMatrix);
        visited.add(stringMatrix);

        while (!items.isEmpty()) {
            int size = items.size();
            for (int i = 0; i < size; i++) {
                int[] curElement = items.poll();
                String curMatrixString = matrix.poll();
                int x = curElement[0];
                int y = curElement[1];

                if (curMatrixString.equals(recovered)) {
                    return true;
                }

                for (int k = 0; k < dirs.length; k++) {
                    int xx = x + dirs[k][0];
                    int yy = y + dirs[k][1];

                    if (xx < 0 || xx >= m || yy < 0 || yy >= n) {
                        continue;
                    }

                    int[][] newMatrix =
recoverMatrixString(curMatrixString);
                    int temp = newMatrix[x][y];
                    newMatrix[x][y] = newMatrix[xx][yy];
                    newMatrix[xx][yy] = temp;
                    String newMatrixString = getMatrixString(newMatrix);
                    if (visited.contains(newMatrixString)) {
                        continue;
                    }

                    items.offer(new int[]{xx, yy});
                    matrix.offer(newMatrixString);
                    visited.add(newMatrixString);
                }
            }
        }

        return false;
    }

    public List<String> getSolution() {
        String[] pathWord = {"Down", "Right", "Up", "Left"};

        Queue<int[]> items = new LinkedList<>();
        Queue<String> matrix = new LinkedList<>();
        Queue<List<String>> path = new LinkedList<>();
        Set<String> visited = new HashSet<>();

        String stringMatrix = getMatrixString(this.matrix.clone());
        items.offer(new int[]{originX, originY});
```

```
            matrix.offer(stringMatrix);
            path.offer(new ArrayList<>());
            visited.add(stringMatrix);

            while (!items.isEmpty()) {
                int size = items.size();
                for (int i = 0; i < size; i++) {
                    int[] curElement = items.poll();
                    String curMatrixString = matrix.poll();
                    List<String> curPath = path.poll();
                    int x = curElement[0];
                    int y = curElement[1];

                    if (curMatrixString.equals(recovered)) {
                        return curPath;
                    }

                    for (int k = 0; k < dirs.length; k++) {
                        int xx = x + dirs[k][0];
                        int yy = y + dirs[k][1];

                        if (xx < 0 || xx >= m || yy < 0 || yy >= n) {
                            continue;
                        }

                        int[][] newMatrix =
    recoverMatrixString(curMatrixString);
                        int temp = newMatrix[x][y];
                        newMatrix[x][y] = newMatrix[xx][yy];
                        newMatrix[xx][yy] = temp;
                        String newMatrixString = getMatrixString(newMatrix);
                        if (visited.contains(newMatrixString)) {
                            continue;
                        }

                        List<String> newPath = new ArrayList<>(curPath);
                        newPath.add(pathWord[k]);

                        items.offer(new int[]{xx, yy});
                        matrix.offer(newMatrixString);
                        path.offer(newPath);
                        visited.add(newMatrixString);
                    }
                }
            }

            return new ArrayList<>();
        }

        private String getMatrixString(int[][] matrix) {
            StringBuilder sb = new StringBuilder();
            for (int i = 0; i < m; i++) {
                for (int j = 0; j < n; j++) {
                    sb.append(matrix[i][j]).append(",");
                }
            }
            return sb.toString();
        }

        private int[][] recoverMatrixString(String str) {
```

```
            String[] parts = str.split(",");
            int[][] res = new int[m][n];
            for (int i = 0; i < m; i++) {
                for (int j = 0; j < n; j++) {
                    res[i][j] = Integer.parseInt(parts[i * n + j]);
                }
            }
            return res;
        }
    }
```

Sliding Game

20. Maximum Number a Night You Can Accommodate

Leetcode 相似问题: Leetcode 198/213/337 House Robber I/II/III

Provide a set of positive integers (an array of integers). Each integer represents number of nights user request on Airbnb.com. If you are a host, you need to design and implement an algorithm to find out the maximum number a night you can accommodate. The constrain is that you have to reserve at least one day between each request, so that you have time to clean the room.

Given a set of numbers in an array which represent number of consecutive days of AirBnB reservation requested, as a host, pick the sequence which maximizes the number of days of occupancy, at the same time, leaving at least 1 day gap in between bookings for cleaning. Problem reduces to finding max-sum of non-consecutive array elements.

// [5, 1, 1, 5] => 10

The above array would represent an example booking period as follows -

// Dec 1 – 5
// Dec 5 – 6
// Dec 6 – 7
// Dec 7 - 12

The answer would be to pick dec 1-5 (5 days) and then pick dec 7-12 for a total of 10 days of occupancy, at the same time, leaving at least 1 day gap for cleaning between reservations.

Similarly,

// [3, 6, 4] => 7
// [4, 10, 3, 1, 5] => 15

转移方程：

- f(0) = nums[0]
- f(1) = max(num[0], num[1])
- f(k) = max(f(k-2) + nums[k], f(k-1))

```
public int rob(int[] nums) {
  if (nums == null) return 0;
  int n = nums.length;
  if (n == 0) return 0;
  if (n == 1) return nums[0];

  int f1 = nums[0]; // exclude max
  int f2 = Math.max(nums[0], nums[1]); // max so far
  for (int i = 2; i < n; i++) {
    int f = Math.max(f1 + nums[i], f2);
    f1 = f2;
    f2 = f;
  }

  return f2;
}
```

Maximum Number a Night You Can Accommodate

21. Find Case Combinations of a String

Leetcode 相似问题

- Leetcode #46 Permutation and Leetcode #47 Permutation II.
- Leetcode #78 Subsets and Leetcode #90 Subsets II.

Method that gives all permutations of a string. Involving examination of subsets.

https://www.glassdoor.com/Interview/1-find-all-the-combinations-of-a-string-in-lowercase-and-uppercase-For-example-string-ab-and-gt-ab-Ab-aB-AB-QTN_582954.htm

Find all the combinations of a string in lowercase and uppercase. For example, string "ab" >>> "ab", "Ab", "aB", "AB". So, you will have 2^n (n = number of chars in the string) output strings. The goal is for you to test each of these strings and see if it matchs a hidden string.

```
private boolean isBitSet(int n, int offset) {
    return (n >> offset & 1) != 0;
}

public List<String> ermutateString(String text) {
    List<String> res = new ArrayList<>();
    if (text == null || text.length() == 0) {
        return res;
    }
    char[] chars = text.toCharArray();
    for (int i = 0, n = (int) Math.pow(2, chars.length); i < n; i++) {
        char[] curr = new char[chars.length];
```

```
            for (int j = 0; j < chars.length; j++) {
                curr[j] = (isBitSet(i, j)) ? Character.toUpperCase(chars[j]) :
chars[j];
            }
            res.add(new String(curr));
        }
        return res;
    }
```

Find All Upper, Lower and Mixed Case Combinations of a String

22. Menu Combination Sum

http://www.1point3acres.com/bbs/thread-289032-1-1.html

给你一个菜单，要你输出一个金额所有能点的不同组 合。要求用完所有钱。

Given a menu (list of items prices), find all possible combinations of items that sum a particular value K. (A variation of the typical 2sum/Nsum questions).

Return the coins combination with the minimum number of coins. Time complexity O(MN), where M is the target value and N is the number of distinct coins. Space complexity O(M).

不过这里着重强调这里有一个小坑！！！之前帖子的人没见人提过，是都没掉里面么？？？就是说 有一个 interface (vector prices, double total) 这里无论你是使用何种algo，都逃脱不了的是 累加之后并比较。这里因为是 double类型的累加， 所以在 register 处理的时候会因为为了保证对其，而对二进制进行变化（当然不是每个double都会出现这种情况）。 这也是double/float 类型在操作的过程中的问题。没办法保证精度（有特殊的方法进行处理，不过这里不需要考虑） 举个栗子：double a = 1.5, b = 1.1213; double c = 2.6213; 这个时候 a+b-c 是不等于0的 具体的理解请会议 computer arch。 哎。。。其实当时这里想到了这个原因，不过因为最近收到了一个感觉还不错的 offer 面试的时候就变的懒了 这里就没提问。 当然也没想到这个小坑竟然把我挂了。。。 正确的过程是，面对这种需要考虑精度的问题 在一开始就要问面试官，精度要求多少 然后在最后求解的时候 给出这样的判断方式 （abs(a+b-c) < 0.0000001） 之类的。 或是 干脆对每个数字都乘以一个大的数，比如都乘以10000。 不过注意不要溢出。。。

```
public class Solution {
    private void search(List<List<Double>> res, int[] centsPrices, int start,
int centsTarget,
                        List<Double> curCombo, double[] prices) {
        if (centsTarget == 0) {
            res.add(new ArrayList<>(curCombo));
            return;
        }

        for (int i = start; i < centsPrices.length; i++) {
            if (i > start && centsPrices[i] == centsPrices[i - 1]) {
                continue;
```

```
            }
            if (centsPrices[i] > centsTarget) {
                break;
            }
            curCombo.add(prices[i]);
            search(res, centsPrices, i + 1, centsTarget - centsPrices[i],
curCombo, prices);
            curCombo.remove(curCombo.size() - 1);
        }
    }

    public List<List<Double>> getCombos(double[] prices, double target) {
        List<List<Double>> res = new ArrayList<>();
        if (prices == null || prices.length == 0 || target <= 0) {
            return res;
        }

        int centsTarget = (int) Math.round(target * 100);
        Arrays.sort(prices);
        int[] centsPrices = new int[prices.length];
        for (int i = 0; i < prices.length; i++) {
            centsPrices[i] = (int) Math.round(prices[i] * 100);
        }

        search(res, centsPrices, 0, centsTarget, new ArrayList<>(), prices);
        return res;
    }
}
```

Menu Combination Sum

23. K Edit Distance

Leetcode 相似问题:

- leetcode #72 Edit Distance
- leetcode #161 One Edit Distance

Find all words from a dictionary that are k edit distance away.

```
public class Solution {
    private void search(String curr, String target, int k, TrieNode root,
                        int[] prevDist, List<String> result) {
        if (root.isLeaf) {
            if (prevDist[target.length()] <= k) {
                result.add(curr);
            } else {
                return;
            }
        }

        for (int i = 0; i < 26; i++) {
            if (root.children[i] == null) {
                continue;
            }
```

```
            int[] currDist = new int[target.length() + 1];
            currDist[0] = curr.length() + 1;
            for (int j = 1; j < prevDist.length; j++) {
                if (target.charAt(j - 1) == (char) (i + 'a')) {
                    currDist[j] = prevDist[j - 1];
                } else {
                    currDist[j] = Math.min(Math.min(prevDist[j - 1],
prevDist[j]), currDist[j - 1]) + 1;
                }
            }

            search(curr + (char) (i + 'a'), target, k, root.children[i],
currDist, result);
        }
    }

    public List<String> getKEditDistance(String[] words, String target, int k)
{
        List<String> res = new ArrayList<>();
        if (words == null || words.length == 0 || target == null ||
                target.length() == 0 || k < 0) {
            return res;
        }

        Trie trie = new Trie();
        for (String word : words) {
            trie.insert(word);
        }

        TrieNode root = trie.root;
        // The edit distance from curr to target
        int[] prev = new int[target.length() + 1];
        for (int i = 0; i < prev.length; i++) {
            prev[i] = i;
        }

        search("", target, k, root, prev, res);

        return res;
    }

    class TrieNode {
        TrieNode[] children;
        boolean isLeaf;

        public TrieNode() {
            children = new TrieNode[26];
        }
    }

    class Trie {
        TrieNode root;

        public Trie() {
            root = new TrieNode();
        }

        // Add a word into trie
        public void insert(String s) {
            if (s == null || s.length() == 0) {
```

```
                return;
            }

            TrieNode p = root;
            for (int i = 0; i < s.length(); i++) {
                char c = s.charAt(i);
                if (p.children[c - 'a'] == null) {
                    p.children[c - 'a'] = new TrieNode();
                }

                if (i == s.length() - 1) {
                    p.children[c - 'a'].isLeaf = true;
                }

                p = p.children[c - 'a'];
            }
        }
    }
}
```

K Edit Distance

24. Boggle Game

Leetcode 相似问题: Leecode #79/#212 Word Search I/II

http://www.1point3acres.com/bbs/thread-191416-1-1.html

比如: 给定一个 2d matrix of letters 和一个 dictionary，找出一条 path 包含最多的存在于字典的 word 个数

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=158462

给你一个 board, 一个 dict 让你计算最多能有多少个 valid 单词出现在这个 Board 上面

http://www.1point3acres.com/bbs/thread-220456-1-1.html

Trie + DFS for search
DFS for max num

这题真的好蛋疼啊，首先你要找出一个单词所有可能出现的位置序列，然后根据每个单词出现的序列在分别做 DFS，分别是不取这个单词，取第一个序列，取第二个序列，etc。

http://www.1point3acres.com/bbs/thread-158462-1-1.html

比如你现在走了一个词 apple, 那么 a, p, p, l, e 这几个 char 的位置不能继续用了. 于是给你一个 board, 一个 dict 让你计算最多能有多少个 valid 单词出现在这个 Board 上面

此题和 LeetCode #212 Word Search II 的区别在于找出一条 path 包含最多的存在于字典的 word 个数。 总结一下，有两种解法，都是基于 word search II 的：

- 用 trie+DFS 找到一个 word 之后，wordCounter 加 1，从这个 word 最后一个 character 的位置再用 trie+DFS 继续查找，知道相邻位置都用过了，或者没有在 trie 里找到 match。
- 用 trie+DFS 找到所有 words 可能出现的位置序列. 然后根据每个单词出现的序列在分别做 DFS，分别是不取这个单词，取第一个序列，取第二个序列，etc。

```
public class Solution {
    private final int[][] dirs = { { 0, 1 }, { 1, 0 }, { 0, -1 }, { -1, 0 } };

    private String path2Word(char[][] board, List<int[]> curPath) {
        StringBuilder sb = new StringBuilder();
        for (int[] coor : curPath) {
            sb.append(board[coor[0]][coor[1]]);
        }
        return sb.toString();
    }

    private void search(List<List<int[]>> paths, char[][] board, int x, int y,
Trie trie,
                        boolean[][] visited, List<int[]> curPath) {
        String curWord = path2Word(board, curPath);
        ReturnType flag = trie.search(curWord);
        if (!flag.hasPrefix) {
            return;
        }
        if (flag.hasWord) {
            paths.add(new ArrayList<>(curPath));
        }

        int m = board.length;
        int n = board[0].length;

        for (int[] dir : dirs) {
            int xx = x + dir[0];
            int yy = y + dir[1];

            if (xx < 0 || xx >= m || yy < 0 || yy >= n) {
                continue;
            }

            visited[xx][yy] = true;
            curPath.add(new int[] { xx, yy });
            search(paths, board, xx, yy, trie, visited, curPath);
            curPath.remove(curPath.size() - 1);
            visited[xx][yy] = false;
        }
    }

    private void searchWords(List<String> res, List<String> curWords,
List<List<int[]>> paths,
                             int start, boolean[][] visited, char[][] board) {
        if (start == paths.size()) {
            if (curWords.size() > res.size()) {
                res.clear();
                res.addAll(curWords);
            }
            return;
        }

        for (int i = start; i < paths.size(); i++) {
```

```
            boolean canUse = true;
            for (int[] coor : paths.get(i)) {
                if (visited[coor[0]][coor[1]]) {
                    canUse = false;
                    break;
                }
            }

            if (canUse) {
                for (int[] coor : paths.get(i)) {
                    visited[coor[0]][coor[1]] = true;
                }
                curWords.add(path2Word(board, paths.get(i)));
                searchWords(res, curWords, paths, i + 1, visited, board);
                curWords.remove(curWords.size() - 1);
                for (int[] coor : paths.get(i)) {
                    visited[coor[0]][coor[1]] = false;
                }
            }
        }
    }

    public List<String> findMostStr(char[][] board, Set<String> dict) {
        List<List<int[]>> paths = new ArrayList<>();

        Trie trie = new Trie();
        for (String word : dict) {
            trie.insert(word);
        }

        int m = board.length;
        int n = board[0].length;
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                boolean[][] visited = new boolean[m][n];
                visited[i][j] = true;
                List<int[]> curPath = new ArrayList<>();
                curPath.add(new int[] { i, j });
                search(paths, board, i, j, trie, visited, curPath);
            }
        }

        List<String> res = new ArrayList<>();
        searchWords(res, new ArrayList<>(), paths, 0, new boolean[m][n],
board);

        return res;
    }

    class ReturnType {
        boolean hasPrefix;
        boolean hasWord;
        ReturnType(boolean hasPrefix, boolean hasWord) {
            this.hasPrefix = hasPrefix;
            this.hasWord = hasWord;
        }
    }

    class TrieNode {
        char c;
```

```
        boolean isEnd;
        Map<Character, TrieNode> children;
        public TrieNode(char c, boolean isEnd) {
            this.c = c;
            this.isEnd = isEnd;
            this.children = new HashMap<>();
        }
    }

    class Trie {
        private TrieNode root;
        public Trie() {
            this.root = new TrieNode(' ', false);
        }

        public void insert(String word) {
            TrieNode cur = root;
            for (int i = 0; i < word.length(); i++) {
                char c = word.charAt(i);
                if (!cur.children.containsKey(c)) {
                    cur.children.put(c, new TrieNode(c, false));
                }
                cur = cur.children.get(c);
            }
            cur.isEnd = true;
        }

        public ReturnType search(String word) {
            TrieNode cur = root;
            for (int i = 0; i < word.length(); i++) {
                char c = word.charAt(i);
                if (!cur.children.containsKey(c)) {
                    return new ReturnType(false, false);
                }
                cur = cur.children.get(c);
            }
            return new ReturnType(true, cur.isEnd);
        }
    }
}
```

Boggle Game

25. Minimum Cost with At Most K Stops

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=215824

给定很多航班信息，至多k stop，找最便宜路线. 给很多 tuple <depart city, dest city, cost> 代表flight, 找出 给订 city A, city B, maxStops, 最小cost的path

Given a flight itinerary consisting of starting city, destination city, and ticket price (2d list) - find the optimal price flight path to get from start to destination. (A variation of Dynamic Programming Shortest Path)

Output list of strings representing a page of hostings given a list of CSV strings.

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=220456

本质就是 BFS 一层一层往外扫。
min cost of flight from start to end if allowed at most k connections. 比如：
A->B, 100,
B->C, 100,
A->C, 500.
如果 k 是 1 的话，起点终点是 A，C 的话，那 A->B->C 最好, 我只想到 BFS 的解法。

```
public int minCost(List<String> lines, String source, String target, int k) {
    if (lines.size() == 0 || k < 0) {
        return 0;
    }
    Map<String, Flight> nodes = new HashMap<>();
    for (String line : lines) {
        String[] s = line.trim().split(",");
        String[] t = s[0].trim().split("->");
        String from = t[0];
        String to = t[1];
        int cost = Integer.valueOf(s[1].trim());
        if (!nodes.containsKey(from)) nodes.put(from, new Flight(from));
        if (!nodes.containsKey(to)) nodes.put(to, new Flight(to));
        nodes.get(from).nextNodes.put(to, cost);
    }

    boolean find = false;
    Queue<String> q = new LinkedList<>();
    Queue<Integer> cost = new LinkedList<>();
    q.offer(source);
    cost.offer(0);
    int stops = -1;
    while (!q.isEmpty()) {
        stops++;
        if (stops > k + 1) {
            break;
        }
        int qSize = q.size();
        for (int i = 0; i < qSize; i++) {
            Flight curr = nodes.get(q.poll());
            int currCost = cost.poll();
            curr.minCost = Math.min(curr.minCost, currCost);

            if (curr.name.equals(target)) {
                find = true;
                continue;
            }

            for (String next : curr.nextNodes.keySet()) {
                int nextCost = currCost + curr.nextNodes.get(next);
                if (nextCost < nodes.get(next).minCost && (stops < k || stops
== k && next.equals(target))) {
                    q.offer(next);
                    cost.offer(nextCost);
                }
```

```
                }
            }
        }

        return find ? nodes.get(target).minCost : -1;
    }
```

Minimum Cost with At Most K Stops

26. String Pyramids Transition Matrix

Given a list of leaf nodes in a pyramid, and a map which indicates what's the possible parent node given a left and right node. Return true if the one of leaf node could turn into the root node, Otherwise, return false.

Example:

```
     root
     / \
    X   X
   /\  /\
   X  X  X
  / \/ \/ \
 A   B  C  D
```

Map:

```
        left: A |  B   |   C | D
right---------------------------------
A             B |A or C|   D | A
B             D |B or C|   A |
C                           B
D
```

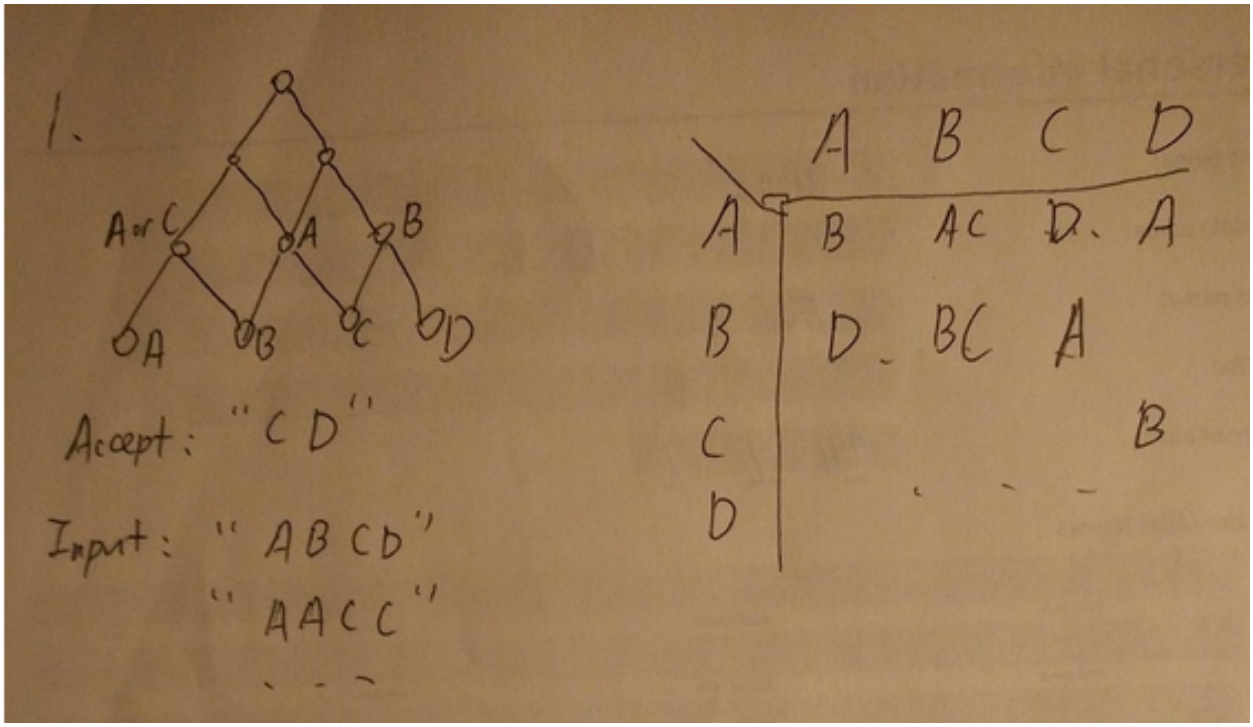
Note:1. If left child is B, right child is A, the parent node could be B or C

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=212660

给一个满二叉树的所有叶子，比如 A B C D E F， 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。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。

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=146537

见下图。给你一个字符对的转换 matrix，表示这个字符对会转化成一个字符(但是有的字符对可能有多个能够转化成的字符，原文是 nondeterministic)。以及若干个合法的终点(最顶上那一个点)状态，多次询问，每次一个字符串如果有一个方法能够走到合法状态就算是 YES，否则 NO
解法：记忆化搜索，记录所有中间状态。因为转化矩阵和合法终点都是固定的，某个字符串要不永远是 YES，要不永远是 NO。最好写个类因为他后来说的……



有了这个表，我们可以决定一个字符串如何往下推演直到只有一个字母。
比如给一个字符串"AC"，很明显，推演结果直接就是"A"。
比如字符串"ABCC"，第一轮我们每两个字母查表一次得出下一个字母。因为有的表中内容是有多个字母，也就是有多个可能，所以整个推演结构会有不同的结果。
拿 ABCC 举例，结果如下（从下往上看）：

```
  A
 A B
A C A
A B C C
```

或者

```
  C
 A B
C C A
A B C C
```

等等。

现在要求给一个这样的表，一个初始字符串和终止结果（字母），要求反悔有没有可能从初始字符串推到给定的终止结果。
比如 ABCC, A。则返回 true
AC, B。则返回 false。

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=293594

由于转化矩阵和合法终点都是固定的，那么设计一个类里面有个<string, bool> 的 map 确定当前字符串是 yes or no, 然后每一次 call check_string 的时候就会存储多条 true or false 的 string 到 map 里面，从而使得以后的 call check_string 如果能碰到一样的串就不用继续 recursion 了。

Nondeterministic Trellis Automata
(https://users.cs.duke.edu/~ola/courses/cps149/problems/week2/trellis.html)

```
public class Solution {
    Map<String, Set<Character>> map;
    private Map<String, Boolean> cache;
    final String SEP = "###";

    Solution(String[] line) {
        cache = new HashMap<>();
        map = new HashMap<>();
        for (String s : line) {
            String[] splitted = s.split(",");
            String key = splitted[0] + SEP + splitted[1];
            Set<Character> set = new HashSet<>();
            for (char c : splitted[2].toCharArray())
                set.add(c);
            map.put(key, set);
        }
    }

    private void getNextLevel(List<String> res, String curr, int start,
StringBuilder sb) {
        if (start == curr.length()-1) {
            res.add(new String(sb));
            return;
        }
        for (int i=start; i<curr.length()-1; i++) {
            String key = curr.charAt(i) + SEP + curr.charAt(i+1);
            for (char c : map.get(key)) {
                sb.append(c);
                getNextLevel(res, curr, start+1, sb);
                sb.setLength(sb.length() - 1);
            }
        }
    }

    private boolean search(String input, String current) {
        if (cache.containsKey(input)) return cache.get(input);
        if (current.length() == 1) {
            cache.put(current, input.contains(current));
            return cache.get(current);
        }

        List<String> cands = new ArrayList<>();
        getNextLevel(cands, current, 0, new StringBuilder());
        for (String cand : cands) {
            // System.out.println(cand);
            if (cache.containsKey(cand)) return cache.get(cand);
            boolean res = search(input, cand);
            if (res) {
                cache.put(cand, true);
                return true;
            }
```

```
            }

            return false;
        }

        public boolean check(String input) {
            cache.clear();
            return search(input, input);
        }
    }
```

String Pyramids

27. Finding Ocean

Leetcode 相似问题:

- Leetocde #200 Number of Islands
- Leetocde #305 Number of Islands II
- Leetocde #694 Number of Distinct Islands
- Leetocde #711 Number of Distinct Islands
- Leetocde #695 Max area of Islands
- Leetcode #463 Island Perimeter

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=141746

```
/**
* Given: An array of strings where L indicates land and W indicates water,
*   and a coordinate marking a starting point in the middle of the ocean.
*
* Challenge: Find and mark the ocean in the map by changing appropriate Ws to Os.
*   An ocean coordinate is defined to be the initial coordinate if a W, and
*   any coordinate directly adjacent to any other ocean coordinate.
*
* void findOcean(String[] map, int row, int column);
*
* String[] map = new String[]{
*  "WWWLLLW",
*  "WWLLLWW",
*  "WLLLLWW"
* };
* printMap(map);
*
* STDOUT:
* WWWLLLW
* WWLLLWW
* WLLLLWW
*
* findOcean(map, 0, 1);
*
* printMap(map);
```

*
* STDOUT:
* OOOLLLW
* OOLLLWW
* OLLLLWW
*/

```
public void floodFill(char[][] board, int i, int j, char oldColor, char
newColor) {
    if (board[i][j] != oldColor || board[i][j] == newColor) {
        return;
    }

    Queue<Integer> queue = new LinkedList<>();
    queue.add(i * board[0].length + j);
    board[i][j] = newColor;

    while (!queue.isEmpty()) {
        int pos = queue.poll();
        int m = pos / board[0].length;
        int n = pos % board[0].length;

        if (m + 1 < board.length && board[m + 1][n] == oldColor) {
            queue.add((m + 1) * board[0].length + n);
            board[m + 1][n] = newColor;
        }
        if (m - 1 >= 0 && board[m - 1][n] == oldColor) {
            queue.add((m - 1) * board[0].length + n);
            board[m - 1][n] = newColor;
        }
        if (n + 1 < board[0].length && board[m][n + 1] == oldColor) {
            queue.add(m * board[0].length + n + 1);
            board[m][n + 1] = newColor;
        }
        if (n - 1 >= 0 && board[m][n - 1] == oldColor) {
            queue.add(m * board[0].length + n - 1);
            board[m][n - 1] = newColor;
        }
    }
}
```

Finding Ocean

28. Preference List

Leetcode 相似问题: leetcode #269 Alien Dictionary

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=218938

也是地里出现过的了，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

拓扑排序解决你有list of list，这些叫preference list。

例如:

[[3, 5, 7, 9], [2, 3, 8], [5, 8]]

然后你要根据这个输入，输出一个总的preference list。 这这一题应该就是:

[2, 3, 5, 8, 7, 9]

因为这道题可能有多种符合要求的输出，如何 break tie by person 1，也就是说 bfs 的时候每次优先选择 person 1 list 里面的元素。

```java
public List<Integer> getPreference(List<List<Integer>> preferences) {
    Map<Integer, Integer> inDegree = new HashMap<>();
    Map<Integer, Set<Integer>> nodes = new HashMap<>();
    for (List<Integer> l : preferences) {
        for (int i = 0; i < l.size() - 1; i++) {
            int from = l.get(i);
            int to = l.get(i + 1);
            if (!nodes.containsKey(from)) {
                inDegree.put(from, 0);
                nodes.put(from, new HashSet<>());
            }
            if (!nodes.containsKey(to)) {
                inDegree.put(to, 0);
                nodes.put(to, new HashSet<>());
            }
            if (!nodes.get(from).contains(to)) {
                Set<Integer> s = nodes.get(from);
                s.add(to);
                nodes.put(from, s);
            }
            inDegree.put(to, inDegree.getOrDefault(to, 0) + 1);
        }
    }
    Queue<Integer> q = new LinkedList<>();
    for (int k : inDegree.keySet()) {
        if (inDegree.get(k) == 0) {
            q.offer(k);
        }
    }
    List<Integer> res = new ArrayList<>();
    while (!q.isEmpty()) {
        int id = q.poll();
        res.add(id);
        Set<Integer> neighbors = nodes.get(id);
        for (int next : neighbors) {
            int degree = inDegree.get(next) - 1;
            inDegree.put(next, degree);
            if (degree == 0) q.offer(next);
        }
    }
    return res;
}
```

Preference List

29. Minimum Vertices to Traverse Directed Graph

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=273389

流水题有向图什么的 我写出来的答案都非常短非常简单易懂 但是这都是我花了很久自己想出来的 绝对没有地里有些人说的那么难 什么 scc 的根本不需要 我也不认识

每选中一个点 则可从该点到达的所有点都算作被遍历了 求最少选中多少个点可以 遍历全图

不用 scc 那么复杂的东西 其实就是用一个 hashmap 记每个 node 的 source 最后算有几个 source 找 source 的时候先假设你现在看到的就是 source 然后一直往下找我用的是 dfs 注意处理环 你现在看到的如果已经在 hashmap 里了 那就是之前已经见过了 所以说 source 已经找到过了 就可以 skip 不知道我说的你能看懂么

不过好像觉得有 circle 的时候不太知道怎么处理，比如边是[[0, 1], [1, 0], [3, 1], [3, 2], [2, 1]] 的时候

我是每次 DFS 的时候另外用了个 unordered_set 记了一下都见过了什么 要是[[0, 1], [1, 0], [3, 1], [3, 2], [2, 1]]

假设从左到右 process 第一次 set 里就只有 0 跟 1 然后接着 process [1, 0]这个已经在 map 里了直接跳过 [3, 1]没在 map 里所以 DFS 最后 set 里有 3210，也就是说 DFS 里又见到[1, 0]是不跳过的

处理环主要三种情况： 一个环连 0 个头， 一个环连着一个头， 一个环连着两个头
分开讨论：
0 个头： toposort 是找不到这个的因为入度不为零。这个可以说大约当成多少个环就有多少个头（每个环只要一个点就能遍历）[1,2],[2,1]
1 个头：那就是一个入度为零的头可以遍历[1,2][2,3][3,2]
2 个头：两个入度为零的头可以遍历([1,2],[2,3],[3,2],[4,3])

思路：
1.处理（一个头，两个头）找所有入度为零的点，全加到 result 集，对每一个点 dfs 或者 bfs 皆可，然后用一个 set 存所有没访问过的点，遇到即从 set 中删除
2.剩下的就是无头的环，继续 dfs/bfs 剩下的 set 中的点，直到 set 为空，每次 dfs 前把开始的点加到 result 里(注意 set 的 concurrent modification error)

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=212885

```
  0  1  2  3  4  5  6  7  8  9
0[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
1[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
2[0, 0, 0 ,0, 0, 0, 0, 0, 0, 1],
3[0, 0, 0, 1, 0, 1, 0, 1, 0, 0],
4[0, 0, 0, 0, 0, 0 ,0, 0, 1, 0],
5[0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
6[0, 0, 0, 0, 0, 0, 1, 0, 0 ,0],
7[0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
8[0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
```

9[0, 0, 0, 1, 0, 0, 1, 0, 0, 0]

这是当时的一个 test case，找最少的点遍历所有点，结果是 0，1，2.

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=220456

所以首先做 SCC (Strongly Connected Components)，对，就是那个 Kosaraju's algorithm，参见 https://en.wikipedia.org/wiki/Kosaraju%27s_algorithm。然后取没有入度的 component。也是蛋疼。

应该是求 List of all vertices without in-degrees.

感觉这题这么处理太复杂了，直接扫所有有入度的点，从所有点的 set 中去掉，再返回 set.size() 即可

```
public class Solution {
    private void search(Set<Integer> res, Map<Integer, Set<Integer>> nodes,
int cur, int start,
                        Set<Integer> visited, Set<Integer> currVisited) {
        currVisited.add(cur);
        visited.add(cur);
        for (int next : nodes.get(cur)) {
            if (res.contains(next) && next != start) {
                res.remove(next);
            }
            if (!currVisited.contains(next)) {
                search(res, nodes, next, start, visited, currVisited);
            }
        }
    }

    public List<Integer> getMin(int[][] edges, int n) {
        Map<Integer, Set<Integer>> nodes = new HashMap<>();
        for (int i = 0; i < n; i++) {
            nodes.put(i, new HashSet<>());
        }
        for (int[] edge : edges) {
            nodes.get(edge[0]).add(edge[1]);
        }

        Set<Integer> visited = new HashSet<>();
        Set<Integer> res = new HashSet<>();
        for (int i = 0; i < n; i++) {
            if (!visited.contains(i)) {
                res.add(i);
                visited.add(i);
                search(res, nodes, i, i, visited, new HashSet<>());
            }
        }

        return new ArrayList<>(res);
    }
}
```

Minimum Vertices to Traverse Directed Graph

30. 10 Wizards

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=277494

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.

wizard[0] list: 1, 4, 5 wizard[4] list: 9 wizard 0 to 9 min distance is (0-4)^2+(4-9)^2=41 (wizard[0] ->wizard[4]->wizard[9])

巫师不同级别，算最小路径. 有向图里面找最短路径

抽象一下，典型的 djstra 问题
BFS 是可以做的，时间复杂度比 Dijkstra 还小
如果保证 wizard 认识关系双向的话，最优路径不会回头，一个简单 dp 就可以了，线性复杂度。
不保证的话，选一种最短路算法实现，顺手就行。 Dijkstra m*log(n).
楼上的 bfs 算法会退化，别的先不说，一个点在队列里出现两次是没有意义的。（补上了还是会退化。）

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=278915

想请教下魔法师这道题，我没太明白题目含义:
简化下比如说有五个魔法师，求 0 到 4 认识的最少距离
0 号认识 1 号和 2 号
1 号认识 3 号
2 号认识 3 号，4 号
3 号认识 4 号

那么 0 号到 4 号的三条路径有: 0->1->3->4, 0->2->3->4, 0->2->4
最小权重和的路径是 0->2->3->4 （3-2）^2 + (4-3)^2= 2
对吗？ 因为帖子里说：介绍人的收费是两个巫师级别差的平方，初始巫师本来就认识的那些巫师不需要收费

多问一句，这个是无向图还是有向图？我理解的认识的话，是两个人都互相认识，对吗？谢谢！

我面试时，面试官没有提到“初始巫师本来就认识的那些巫师不需要收费”，所以你的例子要加上（2-0）^2.但是我觉得这一点对解法没有影响。
我是根据无向来理解。

```
public class Solution {
    public List<Integer> getShortestPath(List<List<Integer>> wizards, int
source, int target) {
        if (wizards == null || wizards.size() == 0) return null;
        int n = wizards.size();
```

```
        int[] parent = new int[n];
        Map<Integer, Wizard> map = new HashMap<>();
        for (int i = 0; i < n; i++) {
            parent[i] = i;
            map.put(i, new Wizard(i));
        }

        map.get(source).dist = 0;
        Queue<Wizard> pq = new PriorityQueue<>(n);
        pq.offer(map.get(source));
        while (!pq.isEmpty()) {
            Wizard curr = pq.poll();
            List<Integer> neighbors = wizards.get(curr.id);
            for (int neighbor : neighbors) {
                Wizard next = map.get(neighbor);
                int weight = (int) Math.pow(next.id - curr.id, 2);
                if (curr.dist + weight < next.dist) {
                    parent[next.id] = curr.id;
                    pq.remove(next);
                    next.dist = curr.dist + weight;
                    pq.offer(next);
                }
            }
        }

        List<Integer> path = new ArrayList<>();
        int t = target;
        while (t != source) {
            path.add(t);
            t = parent[t];
        }
        path.add(source);
        Collections.reverse(path);
        return path;
    }
}
```

10 Wizards

31. Number of Intersected Rectangles

You have a plain with lots of rectangles on it, find out how many of them intersect.

本质是 rectangle intersection + union find

Leetcode 相似问题: Leetcode #323 Number of Connected Components in an Undirected Graph

```
public class Solution {
    private boolean intersect(int[][] r1, int[][] r2) {
        return r1[0][0] < r2[0][0] && r1[0][1] < r2[0][1] && r1[1][0] >
r2[0][0] && r1[1][1] > r2[0][1] ||
               r1[0][0] < r2[1][0] && r1[0][1] < r2[1][1] && r1[1][0] >
r2[1][0] && r1[1][1] > r2[1][1];
    }

    private int find(int val, int[] parents) {
        while (parents[val] != val) {
            val = parents[val];
        }
```

```
            return val;
        }

        public int countIntersection(int[][][] rectangles) {
            int[] parents = new int[rectangles.length];
            for (int i = 0; i < parents.length; i++) {
                parents[i] = i;
            }

            for (int i = 0; i < rectangles.length; i++) {
                for (int j = i + 1; j < rectangles.length; j++) {
                    if (intersect(rectangles[i], rectangles[j])) {
                        int root1 = find(i, parents);
                        int root2 = find(j, parents);

                        if (root1 != root2) {
                            parents[root1] = root2;
                        }
                    }
                }
            }

            Set<Integer> set = new HashSet<>();
            for (int i = 0; i < parents.length; i++) {
                set.add(find(i, parents));
            }

            return set.size();
        }
    }
```

Number of Intersected Rectangles

32. echo TCP client

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=165457

向面试官的server发请求， 读回数据。地里比较少人说这种， 我来详细说一下， 情境是这样的: 想象你开车， 踩下油门，车会加速，放开 油门， 车会减速。

client向server发的请求有以下2种:

- STATUS --表示查询现在车的速度和踩 下踏板的压力;
- THROTTLE 50.1 --- 这条指令是“THROTTLE” 加上一个数字， 表示我现在将踩油门的压力调为 50.1

EXAMPLE: 比如在telnet中

STATUS
0.0 0.0 (这行是server返回的， 第一个数字表示压力， 第二个数字表示速度)
THROTTLE 50.1 (这个指令发出 server没有返回)

STATUS

50.1 3.75

STATUS

50.1 15.98

对就是这样，像是简单物理实验。写完TCP client后，要求是写一个方法将速度控制到达一个target speed 我当时写的是直接最大值，等到了目标值之后再二分

最后一个问题是求这个 delta力 和 delta速度 之间的函数关系。。。。。。。。 醉了。我物理还给老师了。。。。。。时间不够了

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=148195

在一个 server 上有一个汽车的 simulator，要求楼主用一个 socket 连过去，然后可以 call 两个 API：getStatus()可以得到当前的速度和加速度，setStatus(加速度)可以修改汽车加速度。对于每一个加速度，这辆汽车会不断加速最终达到一个稳定速度。问题是要写一个 wrapper function 让汽车的速度稳定在 29mph。楼主花了十几分钟才理解清楚题目，又花了五分钟学习怎么连 socket。最后 code 是写完了，但是输出不正确。时间不够了就没有继续。跟面试官过了程序，他觉得算法没问题，也没找到不输出的原因。最后把程序 copy 走了了事。

加速度公式

Vt^2-Vo^2=2as
一个物体做匀加速运动经过一段距离 S。则末速度的平方减初速度的平方等于距离乘以加速度的 2 倍。

Va=(Vo+Vt)/2=V(t/2)
平均速度等于初速度与末速度的平均数，也等于 t/2 时刻的瞬时速度。

S2-S1=at^2
一个物体做匀加速直线运动，在两段连续且相等的时刻内通过的距离分别是 S1,S2,则两端距离的差等于 at^2

33. Guess Number

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=189016

猜数字

猜一个4位的数字，每位上数字从1到6.

提供一个API，输入一个你猜的数字，返回有几位是猜对的。

比如要猜的数字是1234，猜的数字是1111，则返回1

猜的数字是1212，则返回2

要求写个程序多次调用这个API以后，返回猜的数字结果是什么.

1111,2222,3333,4444,5555
5 次找出 4 个数字 6666 不用找，因为 guess(6666) = 4 - (guess(1111)+...+guess(5555))

然后试
a1a1a1a1, a1a2a2a2, a1a3a3a3 同理 a1a4a4a4 不用试
目的是找出后三个数的数字组合 （不是排列）
找出后 3 个数字 就确定了第一个数字，而且这一轮 a1a1a1a1 不用调 api，第一轮里有
第二轮 2 次： 固定第一个数字，找到后 3 个数字组合

同理第三轮 1 次：固定第二个数字，找到后 2 个数字组合

sorry 第四轮还有 1 次，后两个数字的排列测一个 剩下的就是
不过思考一下这一轮可能不需要.

更正一下：目前能想到最好 5+2+1 = 8 次

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=290126

要猜出一四位数字。

例子：

```
# correct code: 3264
# GUESS 1111 => 0 0 (no correct digits
# GUESS 1214 => 2 0 (digits 2 and 4 are correct and on correct position)
# GUESS 6111 => 0 1 (digit 6 is present, but on a different position)
# GUESS 6211 => 1 1 (digit 2 is not counted towards the second count!)
```

写的时候要连到一个 server 上。

```
# START\n
# GUESS 1111\n =>0 0
# etc.....
```

没看到过面经，几经提示答出来了。隔天说过了。

这个和 bull and cow 不一样。例子是 server 的回复。你的目标是要根据这个回复来猜出原数字是什么。你可以设计要怎么猜。

感觉更好的方法是初始 candidates = {0， 1， 2， 3， 4， ..., 9} (用 priorityqueue 可能比较方便写),
第一轮当然遍历 0000, 1111, 2222... 并且几下每组对应的返回值(a, b), 然后 cnt = a + b， 同时

candidates 删除(a = 0, b = 0)的元素， priorityqueue 以 cnt 从小到大排序。

猜的话初始 XXXX 表示四个位置全部 available， 可以用 candidates 外的元素作为初始值，以 cnt 最大的开始猜，比如 cnt = 2， 就猜 available 位置放 cnt 最大 digit 的元素，遍历下去， AAXX, AXAX, AXXA...这样下去 avaialble 的位置就减少到 4 - cnt, 依次进行， 但是每次只在 available 的位置遍历，减小搜索空间，方法适用于 n 长的 String.

```
public class Solution {
    String target;

    public Solution(String target) {
        this.target = target;
    }

    private int guessServer(String guess) {
        int res = 0;
        Map<Character, Integer> targetMap = new HashMap<>();
        for (char c : target.toCharArray()) targetMap.put(c,
targetMap.getOrDefault(c, 0) + 1);
        Map<Character, Integer> guessMap = new HashMap<>();
        for (char c : guess.toCharArray()) guessMap.put(c,
guessMap.getOrDefault(c, 0) + 1);
        for (char k : guessMap.keySet()) {
            if (targetMap.containsKey(k))
                res += Math.min(guessMap.get(k), targetMap.get(k));
        }
        return res;
    }

    private String genNumber(List<Integer> guessed, int c) {
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < guessed.size(); i++)
            sb.append(guessed.get(i));
        for (int i = guessed.size(); i < 4; i++)
            sb.append(c);
        return sb.toString();
    }

    private String genNumber(List<Integer> guessed) {
        if (guessed == null || guessed.size() == 0) return "";
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < guessed.size(); i++)
            sb.append(guessed.get(i));
        return sb.toString();
    }

    public String guess() {
        List<Integer> res = new ArrayList<>();
        List<Integer> cands = new ArrayList<Integer>(){{
            add(1);
            add(2);
            add(3);
            add(4);
            add(5);
            // insert(6);
        }};
        // System.out.println("\nstart to guess " + target + " ...");
        // System.out.println("res: " + res);
```

```
            // System.out.println("candList: " + candList);
            int counter = 0;
            Iterator<Integer> iter = cands.iterator();
            while (iter.hasNext() && res.size() < 4) {
                int cand = iter.next();
                counter++;
                int guessedCount = res.size();
                String guessCand = genNumber(res, cand);
                int guessRes = guessServer(guessCand);
                // System.out.println("cand: " + cand);
                // System.out.println("guessRes: " + guessRes);
                if (guessRes == guessedCount) {
                    iter.remove();
                } else if (guessRes > guessedCount) {
                    for (int i=guessedCount; i< guessRes; i++) {
                        res.add(cand);
                    }
                    iter.remove();
                } else {
                    // something wrong here
                    return genNumber(res);
                }
            }
            if (res.size() < 4) {
                for (int i=res.size(); i<4; i++)
                    res.add(6);
            }

            // System.out.println("guessed " + counter + " times");
            return genNumber(res);
        }
    }
```

Guess Number

34. Tagged as AirBnB at Leetcode

Leetcode 上分类为 AirBnB 的问题,但没有在前面出现。本人认为这些题目解题思路类似，但大部分不是原题：

- Leetcode #1 Two Sum
- Leetcode #2 Add Two Numbers (#415 Add Strings)
- Leetcode #20 Valid Parenthese
- Leetcode #23 Merge k Sorted List
- Leetcode #108 Convert Sorted Arrays to BST
- Leetcode #136/#137/#260 Single Numer I/II/III
- Leetcode #160 Intersection of Two Linked List
- Leetcode #190 Reverse Bits
- Leetcode #202 Happy Number
- Leetcode #217/#219/#220 Contains Duplicates
- Leetcode #221 Naximal Square (#85 Maximal Rectangle)
- Leetcode #385 Mini Parser

三. Design questions

1. RSS 订阅系统 / Feed System

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=295447

设计一个 RSS 订阅系统的数据库部分，并没有其他的。要的是设计这个 Tables，里头的 Column，我觉得这个部分并不是特别好答。

好多需求，会一直问一直 dig into.比如说，如果我要 search based on name 怎么办，如果说要显示最近的怎么办，如果要显示特定的 Rss provider 怎么办，如果你是 sort by time, 那能不能搞 2 个索引？一个 name 的索引，一个 time 的索引，我真不懂。我觉得没有 DB 设计经验的真的有些难吧。我也没有上过 DB 课程。只是有一点点 DB 基础罢了。

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=299066

设计一个 RSS feeder。有很多 user 和 publisher，user 可以 subscribe/unsubscribepublisher。每个用户登录要能看到一个网页列表，列表里是用户 subscribe 的 publisher 发布的网页，要求按 publish time 排序，最近 publish 的在列表的顶端。我猜可能是因为很多人会扯很多，比如 cache, sharding 之类的，面试官上来就直接告诉我，不要考虑这些，我们只关心一个 single DB instance 里面的 table 和 column 的设计。然后是如果一个文章某个用户读过了，就不要出现在这个列表里了，你怎么改设计。single index 查询太慢，提出 composite index。他貌似比较满意。还剩 10 分钟他就说他问的问题都问完了，不知道是真的问完了，还是我答得太烂了。不继续问了。10 分钟聊得我都没问题问了。

2. key value storage

高频设计题，反正按照一个常见的 NOSQL 说就好了，dynamo, reddis, cassadra。

3. Bank System

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=304012

design 一个 bank system, 跟地里之前的面经一样，要实现一个 deposite(id, timestamp, amount), withdraw(id, timestamp, amount)， check(id) --> return int; 另外还要实现一个 balance 的 function, 这个 function 跟地里的面经不太一样，要求在 logn 的时间复杂度内完成; 给的参数是 ID, startTime, endTime, 但是要注意 startTime 是不包含在内的。比如说给你一个 startTime 0, 这个时间点下一个时间是 10 的话，你算 balance 的时间段应该是从 10 开始，而不是从 0 开始。另外如果 startTime 是负数的话，那么 startTime 就从 0 开始算；这题之前在面经上看到过，看到的做法是用一个 map 来记录 timestamp 与 amount 的之间的对应关系，但是这样有一个问题便是 hashmap 中的元素是无序的，所以如果你直接用 hashmap 的话，你得事先排序，这样时间复杂度就不是 log 级别了。因为准备的时间有效，当时看面经的时候没有考虑到这点，其实这里应该另外用一个 Map<id, List<timestamp>>来存属于那个用户的时间线，这里 suppose 用户交易的时间是顺序的（跟面试官确认下），所以这样我们存的 list 就是有序的，就不需要额外的排序了，直接用 binary search 就好了；写到后面发现了这个问题，马上改 code，但还是没来得及，没写完。。。哎，还是准备不到位了

4. 设计翻译系统

面经也提到过,因为他们家支持 26+语言，每次在页面上更改信息或者添加新的页面，都需要生成不同语言的网页。需要设计一个 micro service，需求是要和 front end decouple，就是 front end engineer 在更改完页面之后，不需要自己操作，这个 service 就能自行的完成翻译。我就是这一点没有答好，我想的是，code change 之后总得需要 commit 什么的吧，在那个时候，send request 到这个 service。面试官表示不能认可。我就不知道如何 decouple from front end。而且最后 feedback 是 high level 都是可以的，但是具体细节不知道怎么实现的。。你们一个内部的 service，问我具体怎么实现的。。其他部分的设计可以参考这个 link http://nerds.airbnb.com/launching-airbnb-jp/.

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=229065

Design a internationalization service

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=290595

设计是地里的经典老题做翻译系统，要求之前地里同学也提过，就是有三个人，前端工程师，翻译官和用户的体验都要做好，他用了很久来解释这个题。这题我也是做了很久的准备，包括他们的 blog 和自己公司的经验。但这个题目和面试官完全无法交流的感觉。可能是因为我没有前端经验，他上来就问在这个 html 里怎么加一个 service call 然后说不用管 syntax，我想那这不就一行 code call 后端 API 不就行了么，然后给了点参数，他说要包含可翻译的和不可翻译的部分，那我就加上去了，然后就在 API 应该怎么写这讨论了五分钟，不是说系统怎么 design，就是这一行 code 要怎么写。有一种和小学生讨论的错觉，他问了很多“这还用说”这种类型的问题。做过 design 的人都知道，API 设计是一个整体，前中后都要考虑，光说前端应该要什么怎么能 design 好呢？面试官草草说赶快到下一个环节。下一个环节他说你设计一下 schema 好了。说有一个网站是显示需要翻译的东西的，问你怎么设计 table。那我就说他需要什么我就给什么就好了，直接一个表里需要的 column 都加上，加了 index 和 composit primary key。他也没问别的。最后就说怎么能让 user 看翻译的东西更快，假设 network 很慢怎么办。这个时候已经没什么时间了，我也没考虑太多，给 webserver 后面加了个 cache，然后让 internal network 可以去更新这个 cache 然后 external user 只看 cache 不看别的。现在想想都不知道应该怎么办，当时说 network 再慢也要 make call 吧，我们可以在浏览器 cache 一下翻译的东西能避免新的 call，其他的怎么样都要 call 一次吧。他还是没说什么，然后就结束了。

因为他们家支持 26+语言，每次在页面上更改信息或者添加新的页面，都需要生成不同语言的网页。需要设计一个 micro service，需求是要和 front end decouple，就是 front end engineer 在更改完页面之后，不需要自己操作，这个 service 就能自行的完成翻译。我就是这一点没有答好，我想的是，code change 之后总得需要 commit 什么的吧，在那个时候，send request 到这个 service。面试官表示不能认可。我就不知道如何 decouple from front end。而且最后 feedback 是 high level 都是可以的，但是具体细节不知道怎么实现的。。你们一个内部的 service，问我具体怎么实现的。。其他部分的设计可以参考这个 link http://nerds.airbnb.com/launching-airbnb-jp/

首先提了 2 个要求：前端程序员不受后端设计的影响，translator 有另一个 portal 输入翻译而且能马上看到效果，你把这两个都答了的话 就开始优化 perf 再来就是有时候 translator 翻译错了修改之后怎么样才能更快的让大家看见（如果你有 cache 的话） 我觉得我答得一般 所以没拿到 senior 所以就不拿出来说了

四. Cross Functional Questions

*有空可以多看看 **engineer blog** http://nerds.airbnb.com/*

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=165457

通过只有一个技巧， 他家很有前景，你很喜欢他家的服务，他家的服务能让人更好的感受local 的文化。
1) what bring you to airbnb?
2) what can you teach your co-woCrkers after you get in?
3) describe a person whom you admire most
4) describe your experience with airbnb
5) where have you been to?
6) what will you do if you win a lottery such as Powerball?
7) what is the biggest fear in your life?
8) how do describe Airbnb to a people back to 2003?
9) if you have a book that writes about your whole life, will you read it? why?
10) if you have a time machine, and you can either go back or go forth, will you choose to go back or to go forth?
11) among all the features of airbnb, what do you want to improve?
12) 描述一件你当时觉得非常risky 的事情，你是怎么做的，结果如何
13) 描述下你自己的品质和你最admire 的一个人

What can you teach me in a few minutes?
Tell me about why you want to work here.
describe you team
how to work with disagreements.
What's your role in a team
why you love traveling
where do you see our company in 10 years

问题有你觉得气床现在面临哪些挑战，怎样可以克服，你犯过什么错误给他人造成了不好的影响之类。

问了挺多问题，都是不同角度，很仔细的问了我对airbnb 文化的理解，我对自己的理解还有我的character 是怎么和airbnb 文化conform 的
中心思想就是我爱旅游， 我喜欢不同的文化， 世界上的恐惧都是不了解造成的。（我确实爱旅游， 我确实喜欢不同文化） 然后airbnb 的core value 也要看看， 把自己的经历往那上扯就好。
When working in teams, describe a time you made the biggest sacrifice.
Describe the best team you've worked with.
Describe a company that you think is doing really well and explain why.
Describe one of the creative things you have done recently.

问题有你觉得气床现在面临哪些挑战，怎样可以克服，你犯过什么错误给他人造成

了不好的影响之类

问题有你过去一年有没有很热情的帮助过别人，如果你下一年不愁钱了打算干啥之类，有没有人对你某个观念转变产生过很大影响。
你想给空气床加什么功能？如果能给空气床减功能，你减什么？你为啥要来空气床，给俩原因？
做过的最scary的事情？
host的经历？
你对airbnb公司五年内发展的预测？
还有什么其他公司也注重hospitality？
airbnb网站那个feature你觉得没必要？
如果是你你要加什么feature？
问了挺多问题，都是不同角度，很仔细的问了我对airbnb文化的理解，我对自己的理解还有我的character是怎么和airbnb文化conform的
中心思想就是我爱旅游， 我喜欢不同的文化， 世界上的恐惧都是不了解造成的。（我确实爱旅游， 我确实喜欢不同文化） 然后airbnb 的core value 也要看看， 把自己的经历往那上扯就好。

你怎么理解 airbnb 的 let anyone belong to anywhere 的 mission
你觉得很 challenging 但却一直在做的事，这里我还跟面试官开玩笑说比如这个面试？哈哈，最后她给我评价说我很有 sense of humor
你做过的一件 meaningful 的事
有什么其他人给你的忠告或者建议让你触动很大

the most hospitable person you met, the biggest trade off in your project.

回答的话基本就是围绕“我是 Airbnb 忠实用户”，“Airbnb 是个无敌的公司”，“我特别想来这里工作”，“我很喜欢旅游”，“我是超级大好人”这几个要点去回答。

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=174191

- What made you decide to apply for airbnb
- What do you suppose Airbnb to become in the future 5 years
- How do you think Airbnb has been working on that direction(指 2 里面回答的) in the past 5 years
- If Airbnb disappeare one day, what do you think should be written on Airbnb's tomb.
- What do you do outside of school and work.
- What will you suggest for Airbnb's development in China
- 有什么事你觉得你做不成，但最后做成功了的
- 你给过别人最好的礼物是什么

Talk about a failure
Talk about a time you made sacrifice for the team

describe you team
how to work with disagreements
what's your role in a team

why you love traveling
where do you see our company in 10 years