Given a csv file :

“some\_name|some\_address|some\_phone|some\_job”

Return a Json format :

Json format “{name:some\_name, address:some\_addres,phone:some\_phone, job:some\_job}”

**Give me an example of when you’ve been a good host**

- E4 X3 \! Y+ w3 B! q

One thing I would like to remove from the Airbnb experience.

**What can you teach me in a few minutes?**

1. regex match, slightly complicated version of<http://leetcode.com/2011/09/regular-expression-matching.html>

2. find maxium square inside a sqaure, similar to<http://stackoverflow.com/questions/1726632/dynamic-programming-largest-square-block>

\* B" M. h5 [. k8 q, c8 x

3. edit distance

前后去了两次他们公司,面了九轮...' `% X9 R4 d( \_  
  
第一轮,小印哥,TEXT JUSTIFICATION. BUG FREE,一次过.  
! L4 X$ ^3 P; O4 `" o" Y3 x  
第二轮,亚裔小哥,不是国人.感觉很傲娇,题目是三张图,楼主已经想不起来了,反正跟杨辉三角有关系.进来就说我至少知道四种做法等等...没有BUG FREE,不过都把BUG修好,然后跑过他给的TEST CASE了/ z$ M  `9 R  Y$ f8 S: a: R4 ]6 @  
  
国人哥哥陪吃饭.  
  
第三轮,SENIOR白人哥哥面设计题,设计一个系统,实现网站国际化.聊得很开心.

最好考虑不同地区的数据存储，各个地方都要有数据的备份，不同语言的ui要支持，这样很多其他的做翻译的只需要改些string 就可以，而不需要真正动里面的文字，运维各个方面，数据库的replica 实时和异步也是根据具体的要求来进行，如果是需要则是用好一点的实时更新，如果是不需要的就可以用差一点的进行异步更新。看具体的要求来进行  
  
第四轮,亚裔小哥,不是国人.PROJECT DIVE,你说一个你做的项目,他不停问问题.  
. \5 T- g4 Z" H0 \0 o7 m\* C5 y  
第五轮,白人美女,CULTRUE FITTING.+ F/ ^1 {8 A4 u' V" u, |  
  
第六轮,白人小哥,CULTURE FITTING.) z! E3 e9 [& B  
  
面完感觉还不错,两天后通知FEEDBACK是POS,不过需要加面三轮.3 ^) S& P) Q, ]: D) p  
  
第七轮,白人小哥,CULTURE FITTING., @% G1 m7 Z" R% C$ F( q  
' K5 I$ r, w6 W% h& j4 B  
第八轮,白人小哥,他在白板上画了三张图,然后给我讲一下十五分钟背景知识,需要实现坐标转化,原题已经记不清楚了.有BUG,改好,跑过他给的TEST CASE.  
6 P! V2 n" r4 Y9 ~0 L  
第九轮,亚裔小哥,不是国人.给你一个异国字典,然后求出这个国家语言的字母先后顺序.其实就是TOPOLOGY SORT典型应用.写完打错了两个字母改好就过了.

这个可以再细查一查

RSS 如果是需要一个整体的，可以看他的是一个整体的还是一个部分的，如果是整体的，可以是先整个存一个cache 然后每次更新的时候是看一个delta 然后再整体作为一个cache

如果是分开的可以让一个message queue 来进行，一切更新都在那个message queue 里面，之后需要的可以从这个queue 里面去拿，不同的地区可以在local 有储存，这样可以省下时间，然后可以做一个cache ，每次给top10，但是后台可以一次做top30，类似prefetch 然后用户需要得到后面的，可以做成page 或者是类似facebook app 的拖动载入，主要还是需要将未来的存在cache 中，如果用户刷新太快，只能让他来等否则可以是秒出结果

实现一个二维数组的iterator的三个函数9 R\* p9 P+ Y; f( U; T+ N( @  
hasNext()  
next()  
remove()  
! Z5 A; P, `1 p' B  
注意，remove要remove原数组的数字.

新人发面经攒积分http://www.meetqun.com/static/image/smiley/default/smile.gif  
两周前的面经了0 `# b2 x: v# }2 e8 W  
( ~- r+ L+ x% |" k7 [# ^  
两轮coding， 一轮design，一轮project deep dive， 两轮culture fit9 o$ K/ L/ D. F& a8 C8 [  
  
1. coding第一个是写一个TCP client，面试官有一个server（会告诉你ip和port），这个server会返回一个string。要求写一个client程序，拿到server的response，然后做一个加减计算。% F  G% G. x) k' e  
  
2 第二轮coding题是text justification。 基本上是这个题目，<https://leetcode.com/problems/text-justification/>， 不过输入一开始是一个长string，不是一个array of words" s3 \+ e  U% O9 a, G  
  
; p" Y) Y, b# ^' O/ F\* b  
project deep dive, 自己选一个project讲。 最后的followup是，如果有机会再重新做这个project，你会怎么improve， 会不会redesign什么的。  
  
我遇到的design 题目比较简单，是设计RSS feed的backend data storage，基本上就是问，最基本的功能需要存哪些数据， 慢慢加一些新的功能，又需要存什么数据。 数据多了，用哪个id去shard之类的。  
  
  
culture 面试就是很多behavioral问题，譬如问你有没有遇到过什么roadblcok啊，你最喜欢的tech innovation是什么啊。外加一些结合airbnb 的，譬如问‘如果你当一个airbnb host，你会怎么做’， ‘如果你加入公司但不是去engineering team，你会选哪个组’。+ c% r2 D+ M1 C+ P1 ?  
  
  
& {! V/ S& W% S1 ~) Z2 H  
他家coding是去电脑上直接做题，要编译／然后写一个main method之类的run 面试官给的test case。 我面试的房间没有钟，我对时间把握的很不好。譬如第一轮coding面试，后来我意识到，其实这个题目只给我大概10分钟，后来我没写完main 就被叫停，说是要再留5分钟让我问问题&#128563;。   
  
给后来人的tips就是，如果不确定还剩多少时间，就问一下面试官，大概需要计划多少时间完成这个coding part。  
3 y2 N- y0 t% s7 d8 o4 G# H  
  
**补充内容 (5-18-2015 11:18 AM):**  
三天了还没有审核，我等着积分看帖子，就把leetcode的链接删了重新发一次。  麻烦版主把这个帖子删了好嘛？  
/ e7 T7 k/ W2 P2 O8 M7 Q  
多谢啦！

Airbnb是当下全世界最火的starup之一，最新估值稳稳的迈过100亿美元大关。Airbnb模式已经开始颠覆”酒店/住宿“领域，简单说，尽管只是最近一两年才兴起，但在很多人看来（尤其是年轻人群），Airbnb将是他们将来最主要的旅行住宿方式：高性价比+原汁原味的当地人生活 > 传统的酒店体验。当然，最火的公司总是有难度的，所以小编抱着试一试的心态投了它家。回顾整个面试过程，由于当时手中握有其他几家的Offer，所以A家的流程走的相对比较顺畅，在此赞一个。

* 面试地点：加州三藩市
* 面试时间：2014年8月份

### Screen

网申一天后，小编收到HR电话，约定第一个电话面试时间。三天后，电面开始。面试官是一个美国人，很利索的自我介绍，然后马上进入Coding。题目是关于如何Parse csv file：举个例子，给定一个CSV文件，格式是 “some\_name|some\_address|some\_phone|some\_job”，要求输出Json format “{name:some\_name, address:some\_addres,phone:some\_phone, job:some\_job}”。输入内容中有些特殊符号要注意处理，这个面试官会提醒。小编的感觉，这道题目的重点是Bug free code，另外面试者需要在30分钟内完成代码，并且编译运行。最后面试官会用几个Test case来测试面试者的代码是否有问题。因此其实A家的电面难度并不低，应该用准备Onsite的标准对待。

### Onsite

由于小编在和HR聊天的时候告知自己的选择窗口比较紧，所以电面当天下午收到Onsite通知。简短截说：A家的Onsite一共有七轮，前五轮每轮四十五分钟，后两轮是三十分钟的culture fit.

第一轮遇到一位亚裔姑娘。主要就是聊天，考察以前做过的项目。面试官会慢慢深入，展开问项目细节和个人贡献。具体的项目回答心得在上篇博客中有[详细讨论](http://baozitraining.org/blog/2014-star-startup-interview-hortonworks/)。由于准备比较充分，所以这轮比较轻松。

第二轮是美国面试官，主要考察系统设计：设计一个distributed cache. 回答这类题目需要对分布式系统有一定了解，小编觉得以下几点是需要讲清楚的：

* Distribute策略：当Cache数据量巨大的时候，需要用多个Node来存储Cache。由于Cache本质上是Key-value-pair，那么可以通过对Key进行类似Hash的Sharding可以决定Value需要存储/读取的Node.
* Cache策略：基本上就是Pull (on-demand)/Push(regular update)的区别。具体情况要考虑需求的具体情况，比如对与Cache数据实时性的要求，对于Performance要求等，一定程度上也考察面试者的沟通能力。
* Failure case以及Scalability的讨论，主要是展示面试者思考问题的全面性。

从第三轮开始，连续三轮Coding interviews: 每轮除去了面试官讨论的时间，大概只有三十分钟做Coding，加之每个解法都需要编译出结果，所以感觉时间比较紧迫。

1. 基于数据结构的算法题：Encode an alien dictionary using a tree and then produce a dictionary using topological traversal. Topological sorting类型，Leetcode中等难度。
2. 算法题：实现Regular expression match，比Leetcode的原题多一两个通配符，但思路想通，做过Leetcode那道题的朋友，写这题难度不大。
3. DP题，具体题目实在无法记得，印象中是中等难度的Leetcode的DP题，sorry.

这三轮技术面试，总体感觉时间比较紧。小编建议最好与面试无关的问题留到最后，因为有些技术问题会有Follow-up，最好留出足够时间做完所有关卡。

最后的两轮面试比较轻松。和两位Non-technical面试官聊天，很多Behavior questions，主要考察面试者是否适合公司文化。比如，为什么选择Airbnb？ 如何改进Airbnb现有的产品？你最想做的事？给你三分钟，教他一件事？这种类型的问题。最好面试前看看网上关于Airbnb文化相关的讨论，答题尽量贴合的它家的做事风格，一般来说，过关问题不大。

Onsite三点半面完所有七轮，五点左右接到Offer，整个面试体验还是非常不错的。

### Reflection

小编感觉A家的面试主要有两个重点：Bug free code and culture fit. 三轮Coding越快写完越有利，很多面试官有Follow-up question，都做出来的话对拿Offer很有帮助。系统设计题，面试官问的非常详细，最好仔细看看类似题目的讨论（关注包子博客是必须的）。多插一句，有的面试官比较拽，态度不是很好，遇到这种情况，个人心态要处理好，不要着急，更不要get defensive，好好做题是关键！整体来看，A家办公条件非常Fancy，Work-life balance应该也不错，每个季度有500刀的Airbnb credit，鼓励员工使用Airbnb旅游。Offer package整体上中规中矩，如果认同A家的文化和愿景，这是一个很理想的可以改变世界的地方。

[**Hortonworks面经—明星Startup系列**](http://baozitraining.org/blog/2014-star-startup-interview-hortonworks/)

不同String之间的distance问题是面试常常考察的高频题目。所谓edit distance，通常指最小的edit distance，即从一个单词通过add，delete, replace变成另一个单词所需要的最小步骤数。实际上，找到一个字典中与当前输入string的edit distance小于k的词，常常用于文档中拼写的自动纠正当。本文主要讨论算法的难点，即如何通过使用DP尽量降低方法的复杂度。

**题目**

找到一个字典中与当前输入string的edit distance [1]，(edit distance通常指最小的edit distance，即从一个单词通过add，delete, replace变成另一个单词所需要的最小步骤数)，为1的词。

**分析思路**

最简单的方法就是把输入的string和字典里每个词比较edit distance，如果是一就返回 比较好的edit distance算法要求n^2时间复杂度 如果n是两个字符串的长度 这样假设字典有m个词，那总时间复杂度就是m\*n^2，非常慢。

我们通常想到的string matching against一个string set的方法是给string set建立trie。这道题不能直接用这种方法，因为我们要求edit distance为1。实际上，edit distance为1就是允许trie里的string有1个字符和输入字符不匹配。这种不匹配既可以是字典里的词多了一个letter，可以是输入的string多了一个letter，也可以是这两个词有一个letter不一样。对于这道题来说，依然为dict建立一个trie，依然去匹配输入的string，在匹配时（只）允许有一个字符不匹配，然后比较输入string和字典里的每一个词，这样在trie里就可以找到所有edit distance为1的词实现时我们借用通配符的概念，如果两个string已经有一个letter不一样，那就用掉了这个通配符，这时如果还有不匹配的letter，那就不用继续比较当前两个词了。

**题目**

找到一个字典中与当前输入string的edit distance小于k的词，通常用于文档中拼写的自动纠正当中。

**分析思路**

这道题看起来很像例题一，但是如果也用通配符的办法不会符合题意，因为edit distance一般指最小distance, 否则就没有意义了。如果这道题也用通配符的话，有可能重复操作一个字符(比如加上a，再减去a)。

Three ways to search for minimum edit distance in a dictionary:

1). Naive approach

The obvious way of doing this is to compute the edit distance from the query term to each dictionary term, before selecting the string(s) of minimum edit distance as spelling suggestion. This exhaustive search is inordinately expensive. Source: Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze: Introduction to Information Retrieval. The performance can be significantly improved by terminating the edit distance calculation as soon as a threshold of 2 or 3 has been reached.

2). Peter Norvig

Generate all possible terms with an edit distance. Generate all possible terms with an edit distance <=2 (deletes + transposes + replaces + inserts) from the query term and search them in the dictionary. For a word of length n, an alphabet size a, an edit distance d=1, there will be n deletions, n-1 transpositions, a\*n alterations, and a\*(n+1) insertions, for a total of 2n+2an+a-1 terms at search time.

This is much better than the naive approach, but still expensive at search time (114,324 terms for n=9, a=36, d=2) and language dependent (because the alphabet is used to generate the terms, which is different in many languages and huge in Chinese: a=70,000 Unicode Han characters)

3). Symmetric Delete Spelling Correction (FAROO)

Generate terms with an edit distance <=2 (deletes only) from each dictionary term and add them together with the original term to the dictionary. This has to be done only once during a pre-calculation step. Generate terms with an edit distance <=2 (deletes only) from the input term and search them in the dictionary. For a word of length n, an alphabet size of a, an edit distance of 1, there will be just n deletions, for a total of n terms at search time.

This is three orders of magnitude less expensive (36 terms for n=9 and d=2) and language independent (the alphabet is not required to generate deletes). The cost of this approach is the pre-calculation time and storage space of x deletes for every original dictionary entry, which is acceptable in most cases.

The number x of deletes for a single dictionary entry depends on the maximum edit distance: x=n for edit distance=1, x=n\*(n-1)/2 for edit distance=2, x=n!/d!/(n-d)! for edit distance=d (combinatorics: k out of n combinations without repetitions, and k=n-d). E.g. for a maximum edit distance of 2 and an average word length of 5 and 100,000 dictionary entries we need to additionally store 1,500,000 deletes.

Remark 1: During the precalculation, different words in the dictionary might lead to same delete term: delete(sun,1)==delete(sin,1)==sn. While we generate only one new dictionary entry (sn), inside we need to store both original terms as spelling correction suggestion (sun,sin).

Remark 2: There are four different comparison pair types:

* dictionary entry==input entry,
* delete(dictionary entry,p1)==input entry
* dictionary entry==delete(input entry,p2)
* delete(dictionary entry,p1)==delete(input entry,p2)

The last comparison type is required for replaces and transposes only. But we need to check whether the suggested dictionary term is really a replace or an adjacent transpose of the input term to prevent false positives of higher edit distance (bank==bnak and bank==bink, but bank!=kanb and bank!=xban and bank!=baxn).

Remark 3: Instead of a dedicated spelling dictionary we are using the search engine index itself. This has several benefits:

It is dynamically updated. Every newly indexed word, whose frequency is over a certain threshold, is automatically used for spelling correction as well.

As we need to search the index anyway the spelling correction comes at almost no extra cost.

When indexing misspelled terms (i.e. not marked as a correct in the index) we do a spelling correction on the fly and index the page for the correct term as well.

Remark 4: We have implemented query suggestions/completion in a similar fashion. This is a good way to prevent spelling errors in the first place. Every newly indexed word, whose frequency is over a certain threshold, is stored as a suggestion to all of its prefixes (they are created in the index if they do not yet exist). As we anyway provide an instant search feature the lookup for suggestions comes also at almost no extra cost. Multiple terms are sorted by the number of results stored in the index.

Reasoning

In our algorithm we are exploiting the fact that the edit distance between two terms is symmetrical:

* We can generate all terms with an edit distance <2 from the query term (trying to reverse the query term error) and checking them against all dictionary terms.
* We can generate all terms with an edit distance <2 from each dictionary term (trying to create the query term error) and check the query term against them.
* We can combine both and meet in the middle, by transforming the correct dictionary terms to erroneous strings, and transforming the erroneous input term to the correct strings.

Because adding a char on the dictionary is equivalent to removing a char from the input string and vice versa, we can on both sides restrict our transformation to deletes only. We are using variant 3, because the delete-only-transformation is language independent and three orders of magnitude less expensive.

Computational Complexity

Our algorithm is constant time ( O(1) time ), i.e. independent of the dictionary size (but depending on the average term length and maximum edit distance), whereas e.g. BK-Trees have a search time of O(log dictionary\_size).

**题目3**

找到一个字典中所有edit distance==k的词的pair。

**分析思路**

这道题可以利用例题2中的思路，先把每个词任意删除k/2个字符的所有可能记录下来，再看那些词的这些变化有重复。

http://baozitraining.org/blog/string-edit-distance/

正则表达式(regular expression)是字符串匹配的重要方式。本文讨论用多种不同的思路来实现类似正则表达式中的"." "\*" 来match多个字符串分析不同算法的性能和复杂度。

**题目**

* '.' Matches any single character.
* '\*' Matches zero or more of the preceding element.
* The matching should cover the entire input string (not partial).

The function prototype should be: bool isMatch(const char \*s, const char \*p)

Some examples:

* isMatch("aa","a") → false
* isMatch("aa","aa") → true
* isMatch("aaa","aa") → false
* isMatch("aa", "a\*") → true
* isMatch("aa", ".\*") → true
* isMatch("ab", ".\*") → true
* isMatch("aab", "c\*a\*b") → true

**思路分析**

这道题乍看很简单，就是把String和Pattern里的字符一个一个match就行了，当遇到.时就可以替换任何字符。但问题是当遇到\*时，比如a\*，应该替换String里多少a呢？尤其是Pattern里有这种组合时: .\* 情况就更复杂了很难马上知道应该替换多少字符。这样就可以使用我们刚才提到的recursion的方式，对于每一种可能的替换，我们都尝试匹配。

因为题目要求返回true or false来表示是否找到全string匹配，我们可以定义recursion function为boolean match(String s, int i, String p, int j)当遇到当S匹配到i，P匹配到j，并且p[j]==’.’ p[j+1]==’\*’时，我们尝试s里从i的任何匹配

for(int itr = i; itr < s.length(); itr++){

    if(match(s, itr, p, j+2)){

        return true;

    }

}

这里注意两点:

1. itr 从i 开始向下一层recursion传。因为\*可以代表0个前面字符，所以.\*可以不匹配任何字符;
2. itr < s.length()而不是(s.length() - (p.length() - j))，因为p剩下的字符依然可能是.\*组合，我们在的现在并不知道，所以依然要匹配到s.length();

**Code Sample**

public class Solution { public boolean isMatch(String s, String p) { assert(p!=null && (p.length()==0 || p.charAt(0)!='\*')); if(p.length()==0) return s.length()==0; if(p.length()==1 || p.charAt(1)!='\*'){ if(s.length()<1 || (p.charAt(0)!='.' && p.charAt(0)!=s.charAt(0))) return false; return isMatch(s.substring(1),p.substring(1)); }else{ int i=-1; while(i<s.length() && (i<0 || p.charAt(0)=='.' || p.charAt(0)==s.charAt(i))){ if(isMatch(s.substring(i+1),p.substring(2))) return true; i++; } return false; } } }

<http://baozitraining.org/blog/regular-expression-matching-problems-2/>

Airbnb: phone screen #1: 不相邻的range 求和最大 [check-in date, check-out date/ check-in date,  check-out date/ check-in date,....]求最多能租出去几天。& m% t) S) D; }. y9 f  G7 D2 c  
           phone screen #2: waived.  
Airbnb: onsite #1. Project Deep Diving# Z( R) m$ G/ Y5 c- \" w1 f+ ]  
           onsite #2. Behavior Cultural Fitting  
           onsite #3. Code: WordBreak 不许用brute force.  
           onsite #4. Code: Text Justification ) p: }  u% s' y, g8 G  
           onsite #5. Ping server: output the timestamp offset   
Dropbox:  **Given a pattern and a string input - find if** string follows the same pattern and return 0 or 1. Examples: 1) Pattern : "abba", input: "redblueredblue" should return 1. 2) Pattern: "aaaa", input: "asdasdasdasd" should return 1. 3) Pattern: "aabb", input: "xyzabcxzyabc" should return 0.   
           先给你有空格的版本：例如 pattern: "a a a a" input : "asd asd asd asd"           后给你没空格的版本：例如 Pattern: "aaaa", input: "asdasdasdasd"   
5 n+ A6 F  N" F

Onsite的一道题，要求实现一个TCP的client。能运行起来，API可以现场网上搜。  
  
蛮奇怪的啊，有朋友能贴段代码分享一下吗？

不同String之间的distance问题是面试常常考察的高频题目。所谓edit distance，通常指最小的edit distance，即从一个单词通过add，delete, replace变成另一个单词所需要的最小步骤数。实际上，找到一个字典中与当前输入string的edit distance小于k的词，常常用于文档中拼写的自动纠正当。本文主要讨论算法的难点，即如何通过使用DP尽量降低方法的复杂度。  
  
题目, z- M+ B7 N\* T. p  
找到一个字典中与当前输入string的edit distance [1]，(edit distance通常指最小的edit distance，即从一个单词通过add，delete, replace变成另一个单词所需要的最小步骤数)，为1的词。  R9 W6 j8 p) b- u% t  
  
分析思路9 |) [+ T( T1 F) g  B6 z2 e\* b3 A! |  
最简单的方法就是把输入的string和字典里每个词比较edit distance，如果是一就返回 比较好的edit distance算法要求n^2时间复杂度 如果n是两个字符串的长度 这样假设字典有m个词，那总时间复杂度就是m\*n^2，非常慢。  g6 C7 t: p5 }  
  
我们通常想到的string matching against一个string set的方法是给string set建立trie。这道题不能直接用这种方法，因为我们要求edit distance为1。实际上，edit distance为1就是允许trie里的string有1个字符和输入字符不匹配。这种不匹配既可以是字典里的词多了一个letter，可以是输入的string多了一个letter，也可以是这两个词有一个letter不一样。对于这道题来说，依然为dict建立一个trie，依然去匹配输入的string，在匹配时（只）允许有一个字符不匹配，然后比较输入string和字典里的每一个词，这样在trie里就可以找到所有edit distance为1的词实现时我们借用通配符的概念，如果两个string已经有一个letter不一样，那就用掉了这个通配符，这时如果还有不匹配的letter，那就不用继续比较当前两个词了。2 I& b, A5 k) r5 F\* L" W) w  
& V) }' O$ c, G1 x! u) O1 q+ P4 ]  
题目  
找到一个字典中与当前输入string的edit distance小于k的词，通常用于文档中拼写的自动纠正当中。! T" R" S: r! O2 o: I( Z  
  
分析思路& P! b) P5 P& o: L! o" ]  v  
这道题看起来很像例题一，但是如果也用通配符的办法不会符合题意，因为edit distance一般指最小distance, 否则就没有意义了。如果这道题也用通配符的话，有可能重复操作一个字符(比如加上a，再减去a)。  
  
Three ways to search for minimum edit distance in a dictionary:  v- F/ f8 T( I6 w  
  
1). Naive approach  
9 W- M- G0 t& \% T  
The obvious way of doing this is to compute the edit distance from the query term to each dictionary term, before selecting the string(s) of minimum edit distance as spelling suggestion. This exhaustive search is inordinately expensive. Source: Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze: Introduction to Information Retrieval. The performance can be significantly improved by terminating the edit distance calculation as soon as a threshold of 2 or 3 has been reached.  
( V) {4 b" E8 |; k8 P  
2). Peter Norvig: S& }) u& y% K5 ~1 ?- U8 f  
  
Generate all possible terms with an edit distance. Generate all possible terms with an edit distance <=2 (deletes + transposes + replaces + inserts) from the query term and search them in the dictionary. For a word of length n, an alphabet size a, an edit distance d=1, there will be n deletions, n-1 transpositions, a\*n alterations, and a\*(n+1) insertions, for a total of 2n+2an+a-1 terms at search time.2 R- h/ @( v3 g) w; l4 d  
$ F& V' x' k7 a. M  
This is much better than the naive approach, but still expensive at search time (114,324 terms for n=9, a=36, d=2) and language dependent (because the alphabet is used to generate the terms, which is different in many languages and huge in Chinese: a=70,000 Unicode Han characters)1 ~6 C5 v4 s! Q; t, F) B  
  
3). Symmetric Delete Spelling Correction (FAROO)  
/ @1 }' A' Y" F8 m  
Generate terms with an edit distance <=2 (deletes only) from each dictionary term and add them together with the original term to the dictionary. This has to be done only once during a pre-calculation step. Generate terms with an edit distance <=2 (deletes only) from the input term and search them in the dictionary. For a word of length n, an alphabet size of a, an edit distance of 1, there will be just n deletions, for a total of n terms at search time.  
This is three orders of magnitude less expensive (36 terms for n=9 and d=2) and language independent (the alphabet is not required to generate deletes). The cost of this approach is the pre-calculation time and storage space of x deletes for every original dictionary entry, which is acceptable in most cases.  
The number x of deletes for a single dictionary entry depends on the maximum edit distance: x=n for edit distance=1, x=n\*(n-1)/2 for edit distance=2, x=n!/d!/(n-d)! for edit distance=d (combinatorics: k out of n combinations without repetitions, and k=n-d). E.g. for a maximum edit distance of 2 and an average word length of 5 and 100,000 dictionary entries we need to additionally store 1,500,000 deletes.  
! K' L\* T- k" e  
Remark 1: During the precalculation, different words in the dictionary might lead to same delete term: delete(sun,1)==delete(sin,1)==sn. While we generate only one new dictionary entry (sn), inside we need to store both original terms as spelling correction suggestion (sun,sin).) ?# E& @6 a: K2 k  
  P% n4 w\* ^7 @  
Remark 2: There are four different comparison pair types:  
dictionary entry==input entry,  
delete(dictionary entry,p1)==input entry& \" x. e2 Z2 H; H% W  w6 P  
dictionary entry==delete(input entry,p2)  
delete(dictionary entry,p1)==delete(input entry,p2)  
The last comparison type is required for replaces and transposes only. But we need to check whether the suggested dictionary term is really a replace or an adjacent transpose of the input term to prevent false positives of higher edit distance (bank==bnak and bank==bink, but bank!=kanb and bank!=xban and bank!=baxn).4 v9 ]! [" W- {0 V4 R  
  
Remark 3: Instead of a dedicated spelling dictionary we are using the search engine index itself. This has several benefits:  
  
It is dynamically updated. Every newly indexed word, whose frequency is over a certain threshold, is automatically used for spelling correction as well.  
+ c5 C4 c1 J0 Y% `+ f- i  
As we need to search the index anyway the spelling correction comes at almost no extra cost.  
When indexing misspelled terms (i.e. not marked as a correct in the index) we do a spelling correction on the fly and index the page for the correct term as well.  
Remark 4: We have implemented query suggestions/completion in a similar fashion. This is a good way to prevent spelling errors in the first place. Every newly indexed word, whose frequency is over a certain threshold, is stored as a suggestion to all of its prefixes (they are created in the index if they do not yet exist). As we anyway provide an instant search feature the lookup for suggestions comes also at almost no extra cost. Multiple terms are sorted by the number of results stored in the index.  
  
Reasoning7 C) @. ]\* b/ K7 q- ?4 m3 }% r  
) k- v3 X- Q$ M  ?  
In our algorithm we are exploiting the fact that the edit distance between two terms is symmetrical:1 l$ ~5 S9 A( b6 W2 e& \_  
We can generate all terms with an edit distance <2 from the query term (trying to reverse the query term error) and checking them against all dictionary terms.  
We can generate all terms with an edit distance <2 from each dictionary term (trying to create the query term error) and check the query term against them.9 p5 g# b# |& y& l  
We can combine both and meet in the middle, by transforming the correct dictionary terms to erroneous strings, and transforming the erroneous input term to the correct strings.  
Because adding a char on the dictionary is equivalent to removing a char from the input string and vice versa, we can on both sides restrict our transformation to deletes only. We are using variant 3, because the delete-only-transformation is language independent and three orders of magnitude less expensive.  
Computational Complexity  
Our algorithm is constant time ( O(1) time ), i.e. independent of the dictionary size (but depending on the average term length and maximum edit distance), whereas e.g. BK-Trees have a search time of O(log dictionary\_size).  
题目3  
找到一个字典中所有edit distance==k的词的pair。  
  
分析思路$ Z7 D4 h  l! D9 A; X9 j' v\* W! \_- k  
这道题可以利用例题2中的思路，先把每个词任意删除k/2个字符的所有可能记录下来，再看那些词的这些变化有重复

题目是两个string integer 相加，返回string integer //输入string是valid  
  
    private static string Add(string s1, string s2){  
        //Precondition check  
        if(s1 == null || s2 == null || s1.Length == 0 || s2.Length == 0)  
            return null;  
          
        //length of s2 is longer or equal s1  
        if(s1.Length > s2.Length){  
            return Add(s2, s1);  
        }  
         
          
        int carry = 0;  
        string ret = "";  
          
        int i1 = s1.Length - 1, i2 = s2.Length - 1;  
          
        while(i1 >= 0){  
            int int1 = s1[i1] - '0';  
            int int2 = s2[i2] - '0';  
            int result = int1 + int2 + carry;  
              
            ret = (result % 10).ToString() + ret;  
            carry = result / 10;    
               
            i1--;  
            i2--;   
        }  
          
        while(i2 >= 0){  
            int result = carry + s2[i2] - '0';  
            ret = (result % 10).ToString() + ret;  
            carry = result / 10;  
            i2--;  
        }  
          
        if(carry > 0)  
            ret = carry.ToString() + ret;  
          
        return ret;  
          
    }

“Every traveller  
deserves a home”和“home everywhere”

我面的是intern的skype interview，不知道对楼主有没有帮助。  
  
implement一个regular expression。  
支持. \* +就好了。  
  
match('abba', 'abba') == true  
match('cda', 'adb') == false  
match('.\*abc', 'abc') == true  
match('.\*abc', 'ababc') == true  
match('.b\*a+', 'caaa') == true  
match('bb+a\*', 'baa') == false  
  
当时比较晕，答的很不好，也就没第二轮了。

//1' Edit Distance  
  
// cats //  
// carts add 26\*4 = 104  
// cat deletion 3  
// bats modification  78  
  
// dict = [ "cat", "bat", "batt", "beetle" ]  
  
// similar(string q) => words from dic that are <= 1 edit distance from q  
// similar("bat") => ["cat", "bat", "batt"]  
  
  
/\*  
Often, we want to encode raw IDs in our database by hiding them behind some   
2-way decodeable hash. So, a URL which would have at one time been:  
  
https://www.airbnb.com/rooms/848662  
  
becomes  
  
https://www.airbnb.com/rooms/kljJJ324hjkS\_  
  
We decode the ID kljJJ324hjkS\_ to 848662 on our backend and serve the   
relevant content. at some point, we start getting 404 errors from clients   
requesting a certain URL of the form  
  
https://www.airbnb.com/rooms/kljjj324hjks\_  
  
This can happen if certain clients, email services, or url shorteners "  
sanitize" the url. Unfortunately, this change breaks decoding and the   
resource cannot be found.  
To assess how big of a deal this is, we may want to recover the IDs of the   
targets that were 404ing.  
  
Your task:  
Given a method decode(testEncStr) which will return the decoded int id if   
testEncStr is decodeable or will throw an exception or return null (  
depending on the language) if not, implement a new method decodeFind(String   
badEncStr) which takes a string and returns the decoded int id.  
\*/

1. regex match, slightly complicated version of http://leetcode.com/2011/09/regular-expression-matching.html  
2. find maxium square inside a sqaure, similar to http://stackoverflow.com/questions/1726632/dynamic-programming-l  
argest-square-block  
  
3. edit distance  
  
算法不难，但是要在有压力的情况下写出高质量，能compile能run，结果对的code并不  
是太容易。估计我99%跪了...  
大家加油！

1.  text justification   
2.  比较两个document 是否相似  
3.  如何做国际化   
4.  Google reader design   
.鐣欏璁哄潧-涓€浜�-涓夊垎鍦�  
. 鍥磋鎴戜滑@1point 3 acres  
  
实现function:  
1. subscribe / unsubscribe  feeds  
2. mark 整个feed topic   as read / unreader  
  
table schema :   
user table:  
id name ... join\_date  
  
Feed table: . 鍥磋鎴戜滑@1point 3 acres  
id name created\_date 鏉ユ簮涓€浜�.涓夊垎鍦拌鍧�.   
  
Post table: ( 1 feed includ many posts)   
id feedId created\_date   
  
Post\_status table:  
id userId,   postId, status (read/unread)  created\_time  
  
================================================  
对于第二个 function, 除了用Post\_status挨个mark postId, 我们 可以design 一个 table 如下:   
  
Feed\_status table: -google 1point3acres  
id  userId feedId status (read/unread)  postTime  postIdRange   
  
如果 user mark 完了 ， 还有新的post 进来 ， 我们可以用Post Time 来标识，   
或者我们可以用个postId range 来表示 在user mark all as read/unread 时候的所有 postId， 我们认为 新的 postId 总是更大。  
  
还有更好的办法么？   不知道为啥被挂了

把CSV字符串转成规定的格式，规则大概是分隔符变成bar，双引号的内容不分隔，  
double双引号转成双引号  
比如：  
"Alexandra ""Alex""",Menendez,alex.menendez@gmail.com,Miami,1  
转成  
Alexandra "Alex"|Menendez|alex.menendez@gmail.com|Miami|1

今天面了Airbnb，网上他家的面经不多，发一个攒RP求onsite。

/\*\*

\* 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

\*/

先问了iteration的方法比recursion好在哪里，然后我用iteration写了一个DFS，当然还写好了main function，现场跑，有一点点小bug，马上就跑过了。然后面试官问了时间复杂度，不断问我如何优化来减少不必要的backtracking。对这道题而言，我说用BFS可能更快一点，每找到一个点就马上标记为‘O’。面试官还是不满意。又问在BFS的while loop里你还能做什么改进？你已经遍历过这个点了如何避免再次遍历它？额沉默。。。沉默了一会儿他说好了，就酱吧。

怒求RP~同时求问如何进一步优化。

关于下面贴的这道面试题  
当文件巨大，所有unique的单词不足以装到内存里面，  
如果分batch来处理，在merge的时候，内存也还是装不下，怎么办？  
如果要实现external merge sort, 感觉 复杂度就上来了  
请问还有什么更好的办法吗？  
  
====== 面试题 ======  
coding第一面让我实现一个词频统计，但是测试文件巨大，读啊读的不同的词就超了内  
存了。结果面试官提醒我要注意存中间结果，所以是读一批，统计一批，然后写回文件  
，然后反复这样做最终得到最后结果。

<http://www.mitbbs.com/article_t/JobHunting/33055313.html>

 leetcode上的flatten 2d vector那题

<http://www.1point3acres.com/bbs/thread-145784-1-1.html>

经典的CSV parser， 之前看了地里面的面经，但是对于双引号匹配的规则仍然有疑惑。写的时候还自己给自己找了一个corner case，并没有解决掉。。。最后跑的结果有一个bug，然后面试官一看有bug说了句ok就挂电话了。。。  
. 涓€浜�-涓夊垎-鍦帮紝鐙鍙戝竷  
  
等拒信中。。。。

<http://www.1point3acres.com/bbs/thread-145741-1-1.html>

a, b, c, d => a| b| c| d. 1point 3acres 璁哄潧  
"abc", a, b => abc| a| b  
"abc ""d""", a, b => abc "d"| a| b

这两天收到打车app，租房app，某all in one hr平台的口头offer，package细节下周  
能出来，这周末提前好好考虑下。  
另外手里还有个fb的offer保底。  
  
板上大牛们帮分析下，现在去哪家还能有点汤喝？  
  
---  
10/28 update  
准备卖身去打车公司当司机了，要去的组我非常喜欢，感觉能学到很多东西。  
其实租房网也很不错，我其实纠结了很久。他们家去了之后基本可以随便去喜欢的组，  
做喜欢的事情，这点很给力。至于车间的装修多么的好，我就不用说了，大家随便搜搜  
图片就可以看出来。  
但是综合来看，最后还是选择去当司机。主要是以下几点原因：  
  
1. 要去的组业务多，公司貌似很重视。我觉得很能锻炼人，这个对new grads来说比什  
么都重要。  
2. 要去的组同胞多，甚至基本都是同胞。我觉得在美国，真正最后能帮上忙称之为人  
脉的，永远只能是同胞。  
3. 未来的manager是标准的德艺双馨，我打听了很多人，都说口碑很好。我觉得找工作  
跟找phd有些许相似的地方，公司本身未必那么的重要，去哪个组以及你的老板怎么样  
，有时候能直接决定你的career path。  
4. Package不错，厂子我觉得有前景，应该能小赚点装修钱。即使争议不小也绝对值得  
一干。  
  
下面是承诺的面经和package。非常感谢板上大牛的帮忙内推！！  
  
==================  
  
Package (ms new grads)  
租房网：拾贰万基本劳动费, 壹万签字费，伍仟挪窝费，价值约贰拾万的臀票。  
人力资源平台：拾叁万基本劳动费, 壹万签字费，扣掉买入价后约价值拾万的赌票。  
打车网：就不说具体数字了，基本劳动飞比租房网和人力资源平台都低，没有签字费，  
臀票不错，业务表现奖金很给力。  
  
  
==================  
  
租房网面经  
  
电面：  
给一组不重复的字符串，找到连接后能回文的字符串pair。  
比如输入是["gab", "cat", "bag", "alpha", "race", "car"] 返回[["gab", "bag"],  
["bag", "gab"], ["race", "car"]]  
这题不难但是没做过的话不太容易一次性做对，多思考一些test case。  
  
onsite：  
1. Culture fit  
2. Culture fit  
  
3. Coding轮，这题之前没见过，描述比较长但是不难。  
给定一个固定的表比如：  
             (right)  
           \_\_A\_\_B\_\_\_C\_  
        A |  B  AC  A  
(left)  B |  C  A   C  
        C |  B  C   A  
  
这个表就意思是两个字母的组合的下一个结果。比如AC组合，结果查表可得出是A。  
如果是BC组合，查表是AB，则说明结果可以是A或者B。  
  
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。  
  
followup是如何优化。  
  
4. Text justification。输入是一长串字符串 需要自己split成单独的单词。  
follow up是修改现有的代码让多余的空格从右到左排。举例，max length是10：  
正常应该是"A  b cd ef"  
从右到左的话则应该是"A b cd  ef"  
  
5. 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。  
举例：  
[  
  [[1, 3], [6, 7]],  
  [[2, 4]],  
  [[2, 3], [9, 12]]  
]  
返回  
[[4, 6], [7, 9]]  
  
总结：租房网的题都不难，但是对做题要求比较高，需要在他们的mac上code。  
期望是尽可能快点bugfree写出来并且运行通过他们的test case。  
此外基本上每一题都需要给出详细的复杂度分析，尤其是递归题，当时推复杂度写了一  
黑板。  
  
----------------------------------------  
  
打车公司面经：  
准备签了就不说那么详细了。  
电面  
1. leetcode高频题  
2. leetcode高频题稍微一点点变种  
  
onsite  
1. leetcode高频题，有一点followup  
  
2.   
先面了20分钟简历，然后一道高频System design，但是内容很偏，完全没准备上，答  
得很烂很被动。感觉整个设计过程感觉不太关注分布式方面的，更多的是考察如何建模。  
  
3. leetcode高频题，各种优化相关的follow up  
  
4. Manager聊天  
  
总结：也是要带自己电脑，写代码要运行，但是对复杂度分析要求不高，我全程没有问  
这个。  
它们家design挺难，但是因为我是new grads可能要求不是太高。  
  
----------------------------------------  
人力资源平台面经：  
OA  
第三套OA，网上很容易搜到。一题类似word ladder，一题是n皇后变种  
  
电面  
1. Number to excel letter  
2. Median of two sorted array  
  
onsite  
1. OOD设计，如何设计一个parking system。  
要求有两种车位，一个是随来随走的public area，一个是需要预定的reversed only   
area。  
对于public车位，如果满了需要安排排队。  
对于reserved车位，如果满了则可以退化到public车位去。  
整轮都在白板画图，不需要写代码。  
  
2. leetcode Generate Parentheses变种。  
输入是一个>0的整数n，要求输出n个配对且合法且正确缩进的if endif组合  
  
比如n=1的时候，输出：  
if  
endif  
  
n=2的时候，输出：  
if  
endif  
if  
endif  
  
if  
  if  
  endif  
endif  
  
3. 给一组字符串代表公司的组织结构。  
每一行的字符串是三个，分别代表员工姓名，上司姓名，其他信息。对于ceo，他的上  
司则是他自己。  
比如:  
Jim, John, JimInfo  
John, Johm, JohnInfo  
Jack, Jim, JackInfo  
Steve, Jim, SteveInfo  
Tim, Jack, TimInfo  
保证没有环。  
  
输入是一个人的名字，要求打印出此人以下的组织架构图，上下级关系用锁进来表现。  
比如输入Jim，则这样打印：  
Jim, JimInfo  
  Jack, JackInfo  
    Tim, TimInfo  
  Steve, SteveInfo  
  
很简单的dfs就好。  
  
4. Manager聊天轮  
  
总结：它们家面经很难，但是我这次题都很简单，应该是运气比较好或者最近bar降低  
了。  
跟a和u一样，coding轮都需要带电脑运行通过。  
  
  
==================  
  
  
下面是几个最近被朋友们问到的比较多的问题。我把我的回答分享给大家。不一定正确  
，仅供参考，权当抛砖引玉了。  
  
\* Q1. 这些hot startup面试比较看重什么，是不是纯刷题就可以？  
  
A: 就我的感觉，startup非常看重文化是否match，而不仅仅是刷题考算法。尤其是  
airbnb，有两轮core values的面试，非常重要。我觉得面试之前之前，至少要做到深  
入了解公司产品，对公司文化精确把握。面试的时候一定要对公司前景有足够信心，言  
语之间充满激情。  
  
  
\* Q2. System design考的多吗？new grads没什么经验，有没有什么书可以推荐的？  
  
A: 因为我是new grads，实际上我没有考到太多design。airbnb完全没考design，uber  
考了一题design但是我答的很烂，zenefits考了oo design没考分布式system design。  
  
准备的话，基本上都在看这个github  
https://github.com/checkcheckzz/system-design-interview  
然后是sideshare里面不同系统的架构slides。  
最后读了两本书，可以推荐一下：  
  
1. big data principles and best practices of scalable realtime data systems  
twitter大神写的，深入浅出，没有相关背景也很好懂，看完对整个分布式架构都会有  
一个很完整的印象。整个书大概就是用一个完整的例子走一遍整个设计流程，理论和实  
现都有讲。值得一口气读完的好书，强推。  
这本书更多的是一个理论的讲解，或者说就是推销作者的lambda architecture。所以  
技术细节很多都是一带而过，所以作者也说了，这书不是technology survey。  
  
2. 大数据日知录  
这个是用代购平台从国内买的，算是国内技术书里面非常靠谱的了。跟上面那本书不太  
一样，这本书更多的是罗列了分布式系统中不同的技术和算法，然后对常见的都进行讲  
解，包括优缺点，对比，不同因素的权衡等。比如fb的haystack和tao，google的几大  
神器，linkedin的kafka等等等等，都有详细的描述。  
这本书就可以称之为technology suervey了，我是跟上面那本一起读的，优势互补。  
  
  
\* Q3. 既然Uber基本一定会考system design, 如何针对性准备，有没有可以分享的资  
源？  
  
A: Uber的几个高频design题最好针对性准备一下。如果有时间的话，尽可能深入：  
design uber, design netflix, design instagram, design spotify。这几题有很多  
是相同的部分，  
  
比如说design uber会有很多关于如何dispatch driver(重点)，如何处理data center  
的fail等等。  
design netflix可能会提到你说的cdn，还有upload，encode之类的问题。  
design spotify跟netflix类似，但是有可能会问如何存playlist。不同设备上  
playlist更新怎么merge，如何处理conflict等等。以及如何做歌曲的推荐。  
  
以下是我觉得挺有用的链接：  
  
1. design uber:   
这几个链接看完会对他们的架构有一个更深入的认识。  
<http://www.infoq.com/presentations/uber-market-platform>  
[http://highscalability.com/blog/2015/9/14/how-uber-scales-their](http://highscalability.com/blog/2015/9/14/how-uber-scales-their-real-time-market-platform.html)  
<http://www.infoq.com/news/2015/03/uber-realtime-market-platform>  
<http://www.open-open.com/lib/view/open1437574144178.html>

<http://www.open-open.com/news/view/1a9051e>

http://www.open-open.com/news/view/1ef97f0  
  
此外，Uber有一个application sharding组件，叫做ringpop。强烈建议把这个看一下。  
https://www.youtube.com/watch?v=OQyqJWQHp3g  
ringpop是基于SWIM gossip协议的一套consistant hashing的组件。也有大量概念和设  
计都是借鉴的dynamo。  
如果时间充足，强烈建议看一下Amazon的这篇业内非常有名的dynamo paper，ringpop  
基本上就是山寨的这个东西。  
[http://www.allthingsdistributed.com/files/amazon-dynamo-sosp200](http://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf)  
  
2. design netflix:  
<http://techblog.netflix.com/> 这里面很多有用的。  
注意netflix用了很多aws的东西，也是cassandra的重度用户。可以看一下cassandra的  
实现机制，以及与hdfs/dynamo的异同。

<http://www.infoq.com/cn/news/2015/02/netfix-data-process-architecture>

http://www.jiuzhang.com/qa/50/  
  
3. design spotify  
看完这几个链接我觉得就足够了  
[http://www.slideshare.net/protocol7/spotify-architecture-pressi](http://www.slideshare.net/protocol7/spotify-architecture-pressing-play?qid=aa0c970c-7dd8-4cf9-9e8c-643c068edb92&v=qf1&b=&from_search=1)  
[http://www.slideshare.net/davidpoblador/spotify-bcn2013slidesha](http://www.slideshare.net/davidpoblador/spotify-bcn2013slideshare?related=1)  
[http://www.slideshare.net/AdamKawa/big-data-at-spotify?related=](http://www.slideshare.net/AdamKawa/big-data-at-spotify?related=2)  
[http://www.slideshare.net/AdamKawa/hadoop-adventures-at-spotify](http://www.slideshare.net/AdamKawa/hadoop-adventures-at-spotify-strata-conference-hadoop-world-2013?related=3)  
<http://www.slideshare.net/dj4b1n/data-at-spotify?related=4>  
[http://www.slideshare.net/JimmyMrdell/playlists-at-spotify-cass](http://www.slideshare.net/JimmyMrdell/playlists-at-spotify-cassandra-summit-london-2013)  
最后一个链接挺重要，是讲他们如何设计和存储playlist的。有一个面经问到了如何对  
playlist做版本管理，这个slides有详细的解释。  
另外值得注意的是，除了cdn，spotify在不同user之间也有p2p的互相通信以降低服务  
器负载。这个上面的slides有讲到。  
  
===========  
  
希望这些东西能帮到大家。尤其是new grads们，今年形势不好，希望都能团结起来互  
相帮助。  
还有其他问题我能帮上忙的可以站内信给我。谢谢！

<http://www.mitbbs.com/article_t/JobHunting/33074761.html>

How fast can you parse strings?

有一个给一堆listing分页保证一个host的listing尽量在一页只出现一次 除了最后一页

大概就是给你几个小数 让你找到几个整数 使整数的和等于小数和的round

同时整数小数两两的差的和最小

小数那题需要考虑负数么？如果是都是正数的话，我先把他们都取floor 然后记下delta ，按照delta 排序，最后将原来小数的round 完 的和与floor 数组和多出来的按照delta 由大到小分，nlogn ，如果有负数的话这个方法好像就不行了~

刚随便一点开一个面经就看到重复的题了……血泪史告诉后人一定要记住这个题怎么做：  
  
第一个题很水忘了，第二题其实很简单，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。.鏈枃鍘熷垱鑷�1point3acres璁哄潧  
-google 1point3acres  
这个题忘了是在leetcode还是hackerrank做过，但是一时间竟然没想起来……  
  
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 ：  
1. 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)  
2. 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  
  
我这个题弄了大概半小时花了比较长的时间一点点想出算法，后来又各种编译不过，做oj习惯了main里面不大会写了……还老是忘记在main里调用自己的函数一直以为有死循环，囧死。比较紧张最后只写了1忘记了2的情况，不过面试官没说什么。估计是挂了吧，在这里提醒一下大家。面试官是中国人一直慢慢引导我…….鐣欏璁哄潧-涓€浜�-涓夊垎鍦�

<http://www.1point3acres.com/bbs/thread-143760-1-1.html>

a, b, c, d => a| b| c| d  
"abc", a, b => abc| a| b. 涓€浜�-涓夊垎-鍦帮紝鐙鍙戝竷  
"abc ""d""", a, b => abc "d"| a| b.鏈枃鍘熷垱鑷�1point3acres璁哄潧  
  
大概的意思是，如果字符串被左右引号包着，去掉那对左右引号，如果引号里面还有引号则要保留，但是如果有重复引号，就缩成一个。所以""d""会变成"d"

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=145741&extra=page%3D3%26filter%3Dsortid%26sortid%3D311%26sortid%3D311>

烙印大叔题目就讲了十几分钟吧 听完就懵了 完全不知道怎么做 发上来问问大家怎么做http://www.1point3acres.com/bbs/static/image/smiley/QQ/em03.gif  
. more info on 1point3acres.com  
他们公司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.1point3acres缃�  
# 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]. more info on 1point3acres.com  
# Sum T = Round(x1+x2+... +xn)  ;  178.93E => 179  
# Output: B = [y1, y2, ...., yn]  
.鐣欏璁哄潧-涓€浜�-涓夊垎鍦�  
# 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  
. 1point3acres.com/bbs  
# 1 + 3 + 3 => 7  
# 0.2 + 0.7 + 0.4 = 1.3  
  
# 1 + 2 + 4 => 7. From 1point 3acres bbs  
# 0.2 + 0.3 + 0.6 = 1.1  
所以[1,2,4]比[1,3,3]要好  
  
大家看看怎么做啊？求地里大神指导！http://www.1point3acres.com/bbs/static/image/smiley/QQ/em25.gif

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=146539&extra=page%3D1%26filter%3Dsortid%26sortid%3D311%26sortid%3D311>

public class Float\_Ceil\_or\_Floor {

// 思路就是先把所有floor加起来，然后看差多少，然后把多少个floor转成ceil 鏉ユ簮涓€浜�.涓夊垎鍦拌鍧�.

// 转的时候按照num本身与ceil的距离排序

public static int[] getNearlyArrayWithSameSum(double[] arr) {

NumWithDiff[] arrWithDiff = new NumWithDiff[arr.length];

double sum = 0.0;

int floorSum = 0;

for (int i = 0; i < arr.length; i++) {

int floor = (int)arr[i];

int ceil = floor;

if (floor < arr[i]) ceil++; // 查是不是4.0这种floor/ceil都是本身的. visit 1point3acres.com for more.

floorSum += floor;

sum += arr[i];

arrWithDiff[i] = new NumWithDiff(ceil, ceil - arr[i]); // 把ceil都放在数组里面进行比较

}

int num = (int) Math.round(sum);

int diff = num - floorSum;

Arrays.sort(arrWithDiff, new Comparator<NumWithDiff>() {.鐣欏璁哄潧-涓€浜�-涓夊垎鍦�

public int compare(NumWithDiff n1, NumWithDiff n2) {

if (n1.diffWithCeil <= n2.diffWithCeil) return -1;

else return 1;

}

});

int[] res = new int[arr.length];

int i = 0;

for (; i < diff; i++) {

res[i] = arrWithDiff[i].num; // 这些放ceil

}

for (; i < arr.length; i++) {.鐣欏璁哄潧-涓€浜�-涓夊垎鍦�

res[i] = arrWithDiff[i].num - 1; // 剩下的只放floor

}

return res;

}

public static void main(String[] args) {

double[] arr = { 1.2, 3.7, 2.3, 4.8 };

int[] res = getNearlyArrayWithSameSum(arr);

for (int i : res) System.out.print(i + " ");

}

}

class NumWithDiff {

-google 1point3acres int num;

double diffWithCeil;

public NumWithDiff(int n, double c) {

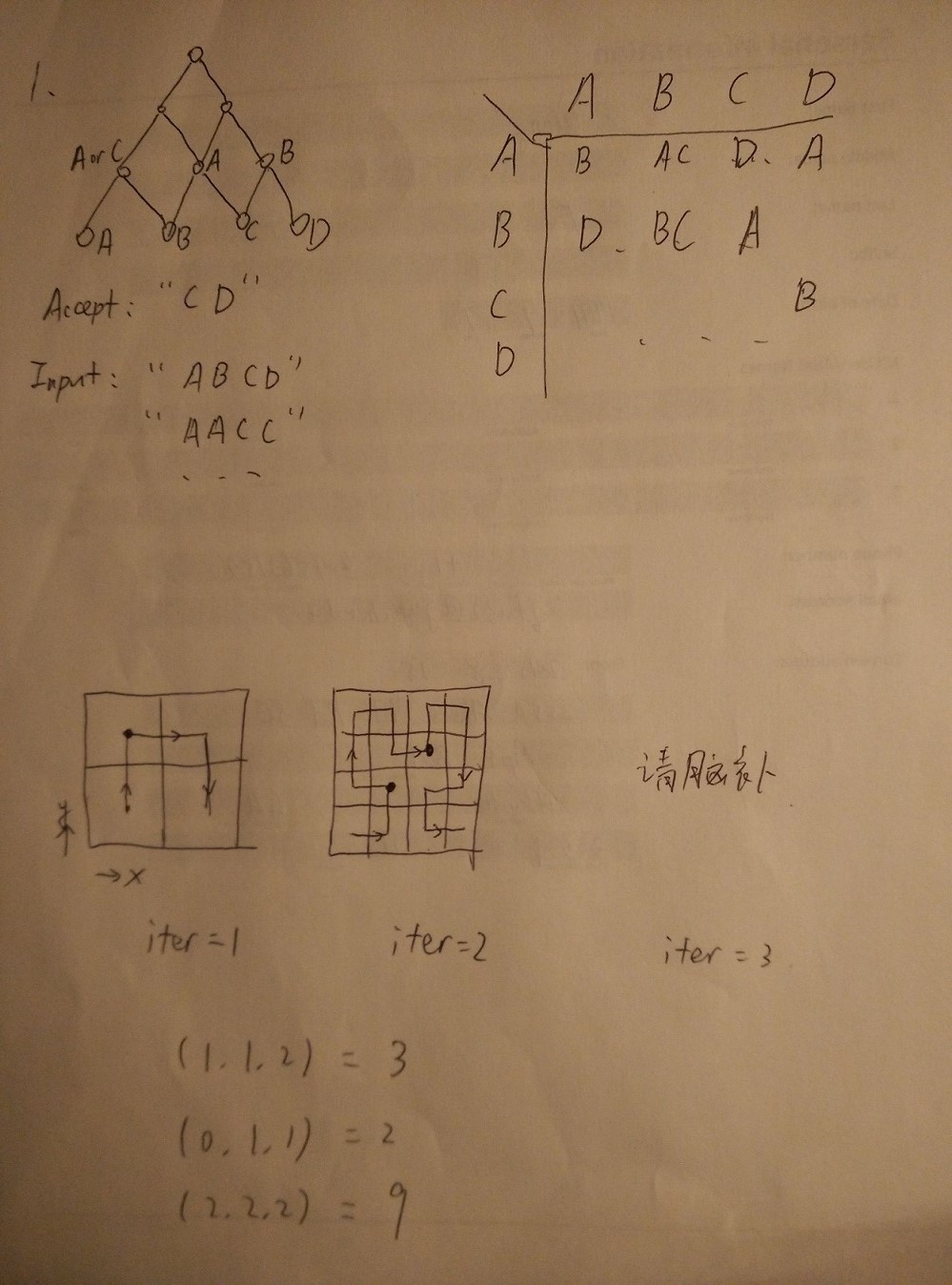
this.num = n;

this.diffWithCeil = c;

}. 鐗涗汉浜戦泦,涓€浜╀笁鍒嗗湴

}

上周面的Airbnb已挂，既然你不给我offer那我就把题说了……语文不好题难描述请见谅……  
  
三个coding面两个core value面。  
  
1. 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. more info on 1point3acres.com  
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……  
  
2. N个员工，每个员工有若干个interval表示在这段时间是忙碌的。求所有员工都不忙的intervals  
解法：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。. 1point 3acres 璁哄潧  
但是我当时也不知道是脑抽了还是脑子太好用了，弄了个堆的，堆里N个元素，存每个员工的下一个端点，如果堆顶是某个员工的左端点就busy\_employees加1，否则减1。这样好处是时间复杂度低了一点点，坏处是给面试官讲了半天而且代码不是太好写……不建议这么做  
-google 1point3acres  
  
3. 见图，忘了这东西叫什么名字了。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？-google 1point3acres  
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。  
  
core value面完全就是问些莫名其妙的东西，比如你想去哪旅游。当然有一点要注意，Airbnb的优点不少在于便宜！而是在于能够更好的了解local culture，不管你认不认同都要这么说……  
鏉ユ簮涓€浜�.涓夊垎鍦拌鍧�. 



<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=146537&extra=page%3D1%26filter%3Dsortid%26sortid%3D311%26sortid%3D311>

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | 68 | [Text Justification](https://leetcode.com/problems/text-justification/) | 15.1% | Hard |  |  |  |  |  | | --- | --- | --- | --- | | 253 | [Meeting Rooms II](https://leetcode.com/problems/meeting-rooms-ii/) | 29.9% | Medium |   http://buttercola.blogspot.com/2015/08/leetcode-meeting-rooms-ii.html  还有一个ip地址压缩的：用这个压缩一个ip地址区间.1point3acres缃� **Classless Inter-Domain Routing (CIDR) Chart** . 1point3acres.com/bbs 这是三轮技术面，会议室这个有个错没有当场编译出来，跪了，剩下俩个都一次通过，然并卵。  核心价值观面试两轮，问你天天玩什么，问我有过什么adventure，我说你看我这么胖，我从恶魔岛游回来了你怕不怕。  估计中午饭也是面试，我吃的太多了，不知道有没有关系。  转天早上hr发邮件说：你[太傻](http://www.1point3acres.com/%E5%A4%AA%E5%82%BB%E7%95%99%E5%AD%A6%E7%BD%91%E5%89%BD%E7%AA%83warald%E6%96%87%E7%AB%A0/)逼了，我们不要你，空气床大门向你打开，想再来找我。  然后没了 |

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=147112&extra=page%3D3%26filter%3Dsortid%26sortid%3D311%26sortid%3D311>

|  |  |
| --- | --- |
| |  | | --- | | 电面是新题，字符串两两组合成palidrome | |
|  |

最后一题：  
  
CIDR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

主要考点是将原来的ip 改成二进制，然后范围也改成二进制，最后用二进制加起来，然后求common prefix 然后最后转回10 （256） 进制

其实也不需要common prefix 因为二进制加法只看最后修改到第几位然后32-修改位数就是最后的结果

发现大家Airbnb电面都被问到palindrome，我来发道不一样的。  
  
给你一个list of posts，每个post对应一个host，这个list是已经排序好了的。  
因为同一个host可以发好几个post，用户不希望看到Airbnb给的推荐房源都是来自同一个户主。所以面试官希望对这个list调整一下排序，让每一页里的post不出现相同的host，otherwise preserve the ordering。. 鐗涗汉浜戦泦,涓€浜╀笁鍒嗗湴  
每一页中有12个post。

<http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=147508&extra=page%3D3%26filter%3Dsortid%26sortid%3D311%26sortid%3D311>

电面比较常规，写了两道题，word search II和不等长数组的iterator，要求实现next(), hasNext(), remove()。第二题出了n多bug，好在最后还是调出来了. 1point3acres.com/bbs  
  
onsite给安排了六轮，3轮技术，3轮非技术  
Round 1 project deepdive:  讲过去做过的项目，因为背景和面试官很相近，所以相谈甚欢  
Round 2 machine learning: 这轮按安排应该是技术面，但最后还是变成了project deepdive，只是侧重于过去项目里的ML内容，期间会让稍微解释下基本概念  
Round 3 coding 1: 碰到了一道非面经题。我发誓从来没有在网上见过任何类似的题。题目是这样的，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。希望不要给太负面的评价 http://www.1point3acres.com/bbs/static/image/smiley/QQ/em34.gif  
  
Round 4 coding 2: Leetcode Alien Dictionary，秒杀  
Round 5-6: culture fit, 天马行空谈笑风生。  
  
按照最近看到的面经，估计最后应该还是会挂，毕竟有一轮没给出正确答案。希望对大家有帮助

<http://www.1point3acres.com/bbs/thread-148195-1-1.html>