hulu面经

作者：隆兄

请大家不要把本文档流传到网上，不过私下传递还是可以的。。前面是实习面经，后面是全职面经（去掉和实习面经一样的部分）。请不要将本文档转给任何盈利组织，最终解释权。。。我什么都没干！

首先是我面跪的那题：

"""
Input:

Started name=dump_logs jobid=f863
Started name=dump_logs jobid=g301gas
...
Ended jobid=r0eas time=103
Ended jobid=f863 time=1021
Started name=grep_logs jobid=ac3de
Ended jobid=g301gas time=1343
Started name=read_logs jobid=r0eas
...
Started name=write_logs jobid=dg2dz
Ended jobid=ac3de time=52

Output:

Print out the names of the K distinct jobs (not job instance IDs, so de-dupe on job name) with the longest running individual job instances (no totaling). Note they can be out of order.

K=2

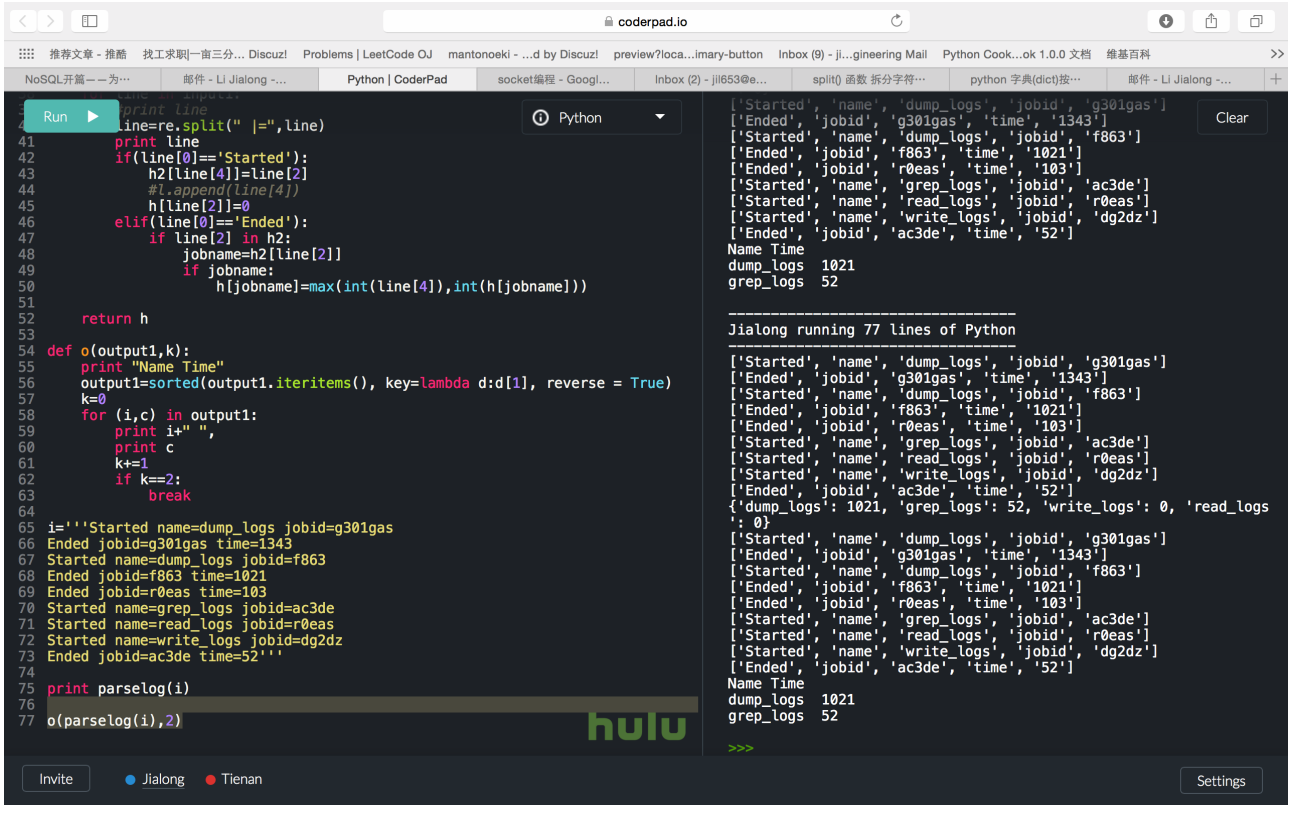
Name Time
dump_logs 1343
read_logs 103

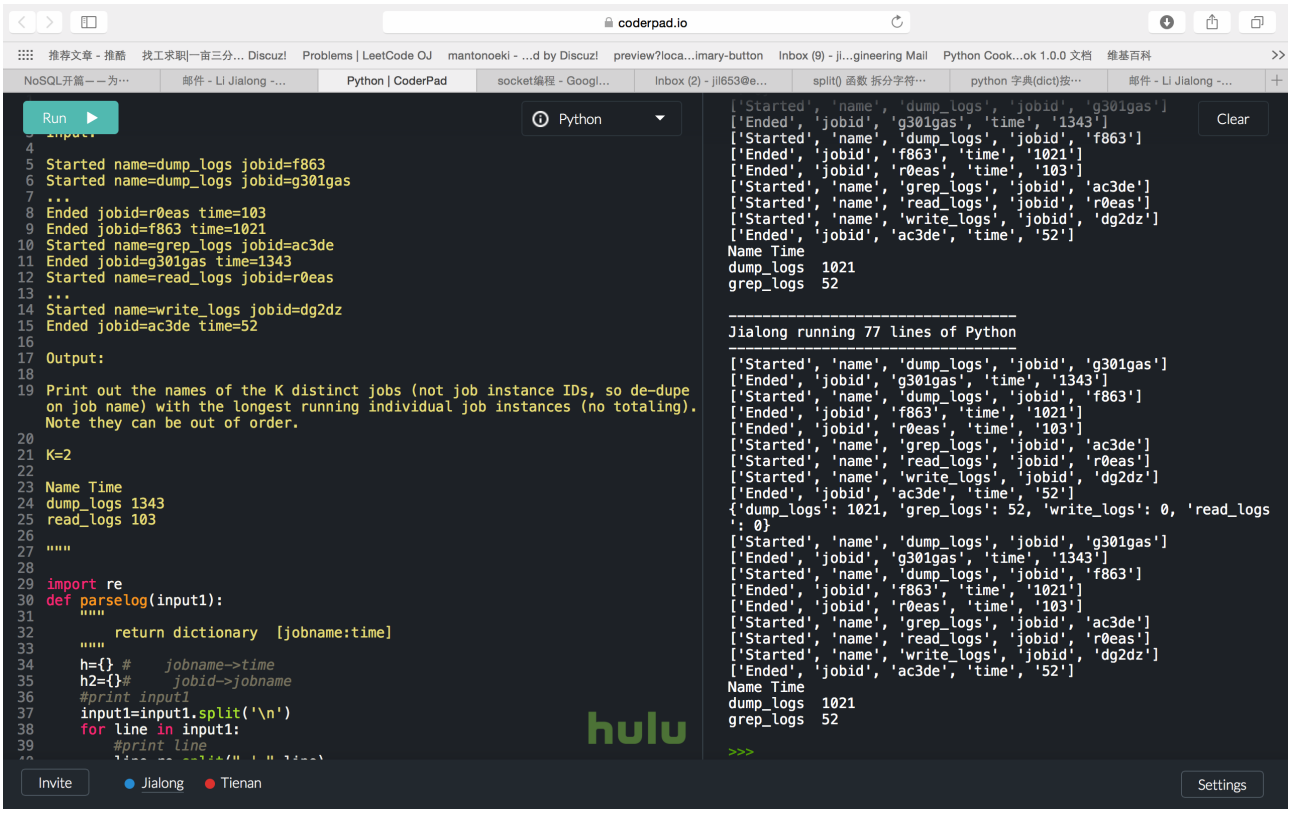
"""

```
import re
def parselog(input1):
    """
        return dictionary  [jobname:time]
    """
    h={} #    jobname->time
    h2={}#     jobid->jobname
    #print input1
    input1=input1.split('\n')
    for line in input1:
        #print line
        line=re.split(" I=",line)
        print line
        if(line[0]=='Started'):
            h2[line[4]]=line[2]
            #l.append(line[4])
            h[line[2]]=0 ********这个地方是有bug的。。。小哥指出来的。。。
        elif(line[0]=='Ended'):
            if line[2] in h2:
                jobname=h2[line[2]]
```

```
            if jobname:
                h[jobname]=max(int(line[4]),int(h[jobname]))

    return h

def o(output1,k):
    print "Name Time"
    output1=sorted(output1.iteritems(), key=lambda d:d[1], reverse = True)
    k=0
    for (i,c) in output1:
        print i+" ",
        print c
        k+=1
        if k==2:
            break

i='''Started name=dump_logs jobid=g301gas
Ended jobid=g301gas time=1343
Started name=dump_logs jobid=f863
Ended jobid=f863 time=1021
Ended jobid=r0eas time=103
Started name=grep_logs jobid=ac3de
Started name=read_logs jobid=r0eas
Started name=write_logs jobid=dg2dz
Ended jobid=ac3de time=52'''

print parselog(i)

o(parselog(i),2)
```

面试的界面是这样的，左边写代码 右边是控制台。希望大家早点做准备。






## 16 * 16的matrix，只有T和F，要求构件一棵树，如果全是T或F，节点就是T或F，并且不再继续遍历，否则就是M，并且继续遍历

建立一棵决策树，遇到块内T和F都有就标为M,然后继续建立决策树，如果块内只有T或F就停止并标记为T或F。

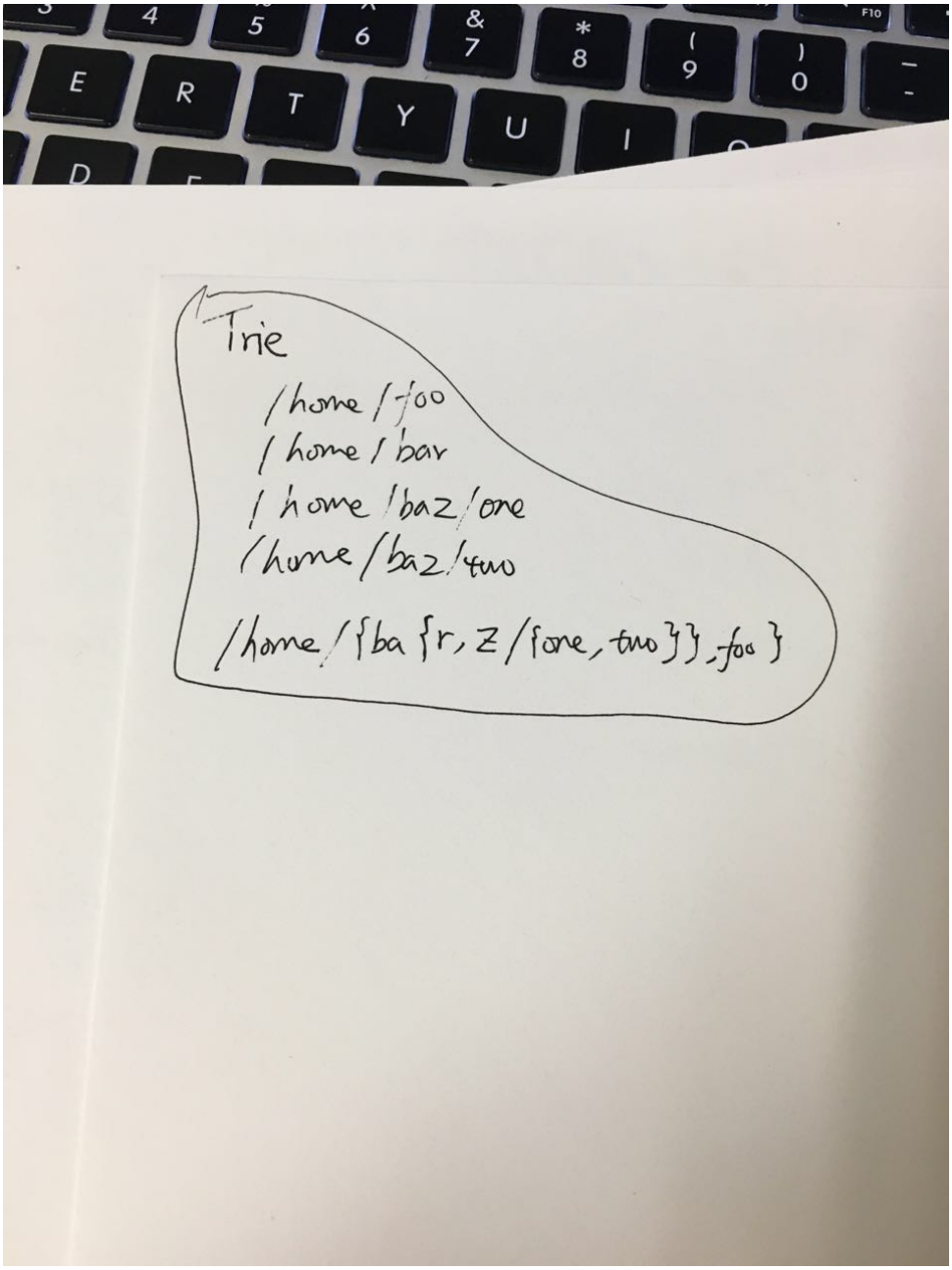
```
class Node(object):
        def __init__(self,value='M'):
                self.value=value
                self.children=[]

class Solution(object):
        def recurse(self,nums,i1,i2,j1,j2,node):
                #print str(i1)+"+"+str(i2)+"+"+str(j1)+"+"+str(j2)
                if i1+1==i2 and j1+1==j2:
                        node.value=nums[i1][j1]
                        return node
                p=nums[i1][j1]
                for i in range(i1,i2):
                        for j in range(j1,j2):
                                if not (nums[i][j]==p):
                                        p='M'
                                        node.value=p
                                        l1=Node()
                                        l2=Node()
                                        l3=Node()
```

```
                                        l4=Node()
                                        node.children.append(self.recurse(nums,i1,(i1+i2)/2,j1,(j1+j2)/
2,l1))
                                        node.children.append(self.recurse(nums,(i1+i2)/2,i2,j1,(j1+j2)/
2,l2))
                                        node.children.append(self.recurse(nums,i1,(i1+i2)/2,(j1+j2)/
2,j2,l3))
                                        node.children.append(self.recurse(nums,(i1+i2)/2,i2,(j1+j2)/
2,j2,l4))

                node.value=p
                return node
nums=[
        ['F','T','F','F','T','T','T','T'],
        ['T','F','F','T','F','T','F','T'],
        ['F','T','F','T','T','T','F','T'],
        ['T','T','F','T','F','F','T','T'],
        ['T','T','F','F','T','T','T','T'],
        ['F','F','T','T','T','T','F','T'],
        ['T','T','T','F','T','F','T','T'],
        ['T','T','T','T','T','T','T','T']
        ]
l=Node()
i=Solution()

#nums=(nums[0:4])[0:4]
print nums
#i.recurse(nums,0,4,0,4,l)
i.recurse(nums,0,8,0,8,l)

print l.children[3].children[2].value
```

一道老面经题，读取输入并预测其输出。其实就是trie。

```
class NoName:
    def __init__(self):
        self.children = {}
        self.name = ''

    def has_child(self, child):
        """
        :param child: a string
        :return: a boolean value
        """
        return child in self.children
    def add_child(self, child):
        """
        :param child: a string
        :return: a NoName instance
        """
        child_node = NoName()
        child_node.name = child
        self.children[child] = child_node
```

```
        return child_node

    def get_node(self, child):
        """
        :param child: a string
        :return: a NoName instance
        """
        if self.has_child(child):
            return self.children[child]
        else:
            return self.add_child(child)

    def add_list(self, input_string):
        """
        :param input_string: a string
        :return: void, no return value
        """
        current_node = self.get_node(input_string[0])
        input_string = input_string[1:] # take the substring starting at position 1
        if len(input_string) < 1:
            current_node.get_node('')
        else:
            current_node.add_list(input_string)

    def scan(self):
        """
        :return: a string
        """
        if len(self.children) == 0:
            return self.name
        if len(self.children) == 1:
            return self.name + self.children.values()[0].scan()

        temp = []  # vector
        for child in self.children.values():  # for child in values of self.children hashmap
            temp.append(child.scan())

        return self.name + '{' + ','.join(sorted(temp)) + '}'


x = NoName()
x.add_list('/home/foo')
x.add_list('/home/bar')
x.add_list('/home/baz/one')
x.add_list('/home/baz/two')

print x.scan()
```




simplify path

```
#def say_hello():
#    print('Hello, World')

#for i in range(5):
#    say_hello()

#./folder1/folder2/../folder3 -> /folder1/folder3

#input path -> simple path
def simplify(path):
#     try:

#     except:
    if not path or path is None:
#         print ...
```

```
        return ''
    v=path.split('/')
    strpath=''
    for i in range(len(v)):
        if v[i]=='.' or not v[i]:
            continue
        if v[i]=='..':
            for j in range(len(strpath))[::-1]:
                if strpath[j]== '/':
                    strpath=strpath[:j]
                    break
        else:
            strpath+='/'+v[i]
    if not strpath:
            return '/'
    return strpath

print(simplify('./'))
print(simplify('./folder1//folder2/../folder3'))
print(simplify('./folder1//./.././/../../...'))
print(simplify('//'))
#./folder1/../folder2/..
print(simplify(""))
print(simplify(None))
```

isanagram。

```
#"The day is sunny" "this is sun"
#"The day is suuny" "this sun"
#input whole sentence, generated sentence

def regression(input1,input2):
    alpha=[0 for i in range(26)]
    alpha_set=()
    for i in input2:
        if(0<=(ord(i)-ord('A'))<=26):
            i=chr(ord(i)-ord('A')+ord('a'))
        if(0<=(ord(i)-ord('a'))<=26):
            location= ord(i)-ord('a')
            alpha[location]+=1
            alpha_set.add(chr(location))
    for i in input1:
        if(0<=(ord(i)-ord('A'))<=26):
            i=chr(ord(i)-ord('A')+ord('a'))
        if(0<=(ord(i)-ord('a'))<=26):
            location= ord(i)-ord('a')
            if alpha[location]>0:
                alpha[location]-=1
    for i in alpha:
        if i>0:
            return False
    return True

print(regression("The day is sunny","this is sun"))
print(regression("The day is sunny","this sun"))
```

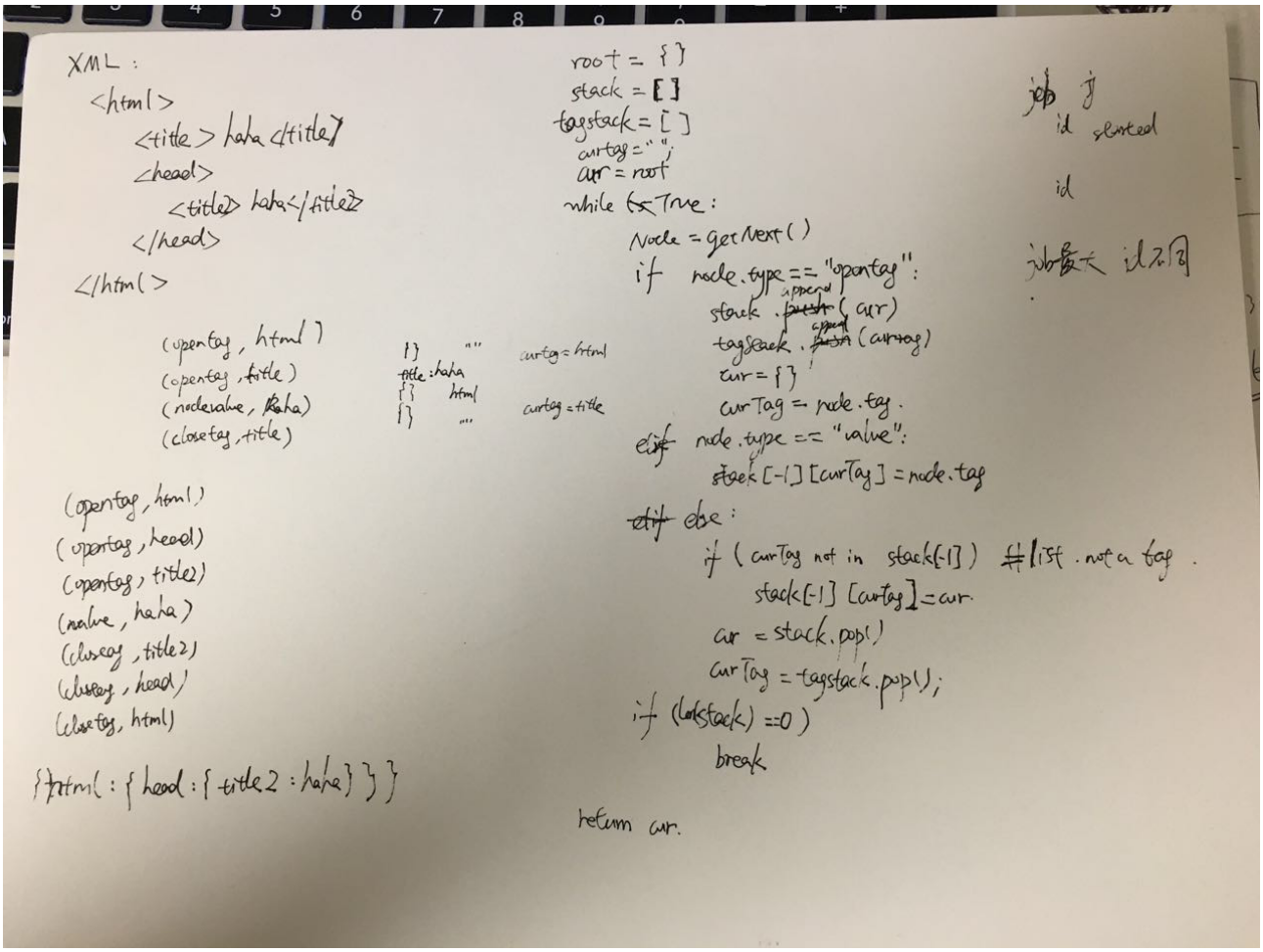
print(regression("The 123 day is sunny !!!", "this ssun"))

注意这个算法不是最优的。

下面是手写的面经题：

xml parser

给一个Iterator，每次next返回一个token，token中有tag名，类型，和value三个量。让根据这个iterator建一棵XML树




basic calculator I and II

Basic Calculator

```
def calculate(self, s):
    res, num, sign, stack = 0, 0, 1, []
    for i in s+"+":
        if i.isdigit():
            num = 10*num + int(i)
        elif i in "+-":
            res += num * sign * stack[-1]
            sign = 1 if i=='+' else -1
            num = 0
        elif i == "(":
            stack.append(sign * stack[-1])
            sign = 1
        elif i == ")":
            res += num * sign * stack[-1]
            num = 0
            stack.pop()
    return res
```

```
class Solution(object):
    def calculate(self, s):
        ops = []
        nums = []
        s = s.replace(" ", "")
        num = ""
        for i in s:
            if i.isdigit():
                num += i
            else:
                nums.append(int(num))
                num = ""
                ops.append(i)
            if len(num) > 0:
                nums.append(int(num))
        i = 0
        while (i < len(ops)):
            if ops[i] == "*":
                nums[i] = nums[i] * nums[i+1]
                ops.pop(i)
                nums.pop(i+1)
            elif ops[i] == "/":
                nums[i] = nums[i] / nums[i+1]
                ops.pop(i)
                nums.pop(i+1)
            else:
                i = i+1
        i = 0
        while (i < len(ops)):
            if op[i] == "+":
                nums[i] = nums[i] + nums[i+1]
                ops.pop(i)
                nums.pop(i+1)
            elif ops[i] == "-":
                nums[i] = nums[i] - nums[i+1]
                ops.pop(i)
        return nums[0]
```

LRU cache

```
def __init__(self, capacity):
    self.capacity = capacity
    self.h = collection.OrderedDict()

def get(self, key):
    if key not in self.h:
        return -1
    v = self.h.pop(key)
    self.h[key] = v

def set(self, key, value):
    if key in self.h:
        self.h.pop(key)
    elif self.capacity == len(self.h):
        self.h.popitem(last=False)
    self.h[key] = value
```

because we need to reach O(1) time for get. we need a hash map. in python, it's dictionary. To record the value, traditionally we need to use a linked list in C++, but here we can use OrderedDict directly.

LRU Cache

```
class Node(object):
    def __init__(self, key, value):
        self.key = key
        self.value = value
        self.prev = None
        self.next = None

class LRUCache(object):
    def __init__(self, capacity):
        self.capacity = capacity
        self.h = {}
        self.front = Node('f', 'f')
        self.tail = Node('t', 't')
        self.front.next = self.tail
        self.tail.prev = self.front

    def get(self, key):
        if key not in self.h:
            return -1
        self.insertFront(self.unlink(self.h[key]))
        return self.h[key].value

    def set(self, key, value):
        if key in self.h:
            self.insertFront(self.unlink(self.h[key]))
            self.h[key].value = value
        else:
            if self.capacity == len(self.h):
                del self.h[self.unlink(self.h[self.tail.prev.key]).key]
            self.h[key] = Node(key, value)
            self.insertFront(self.h[key])
```

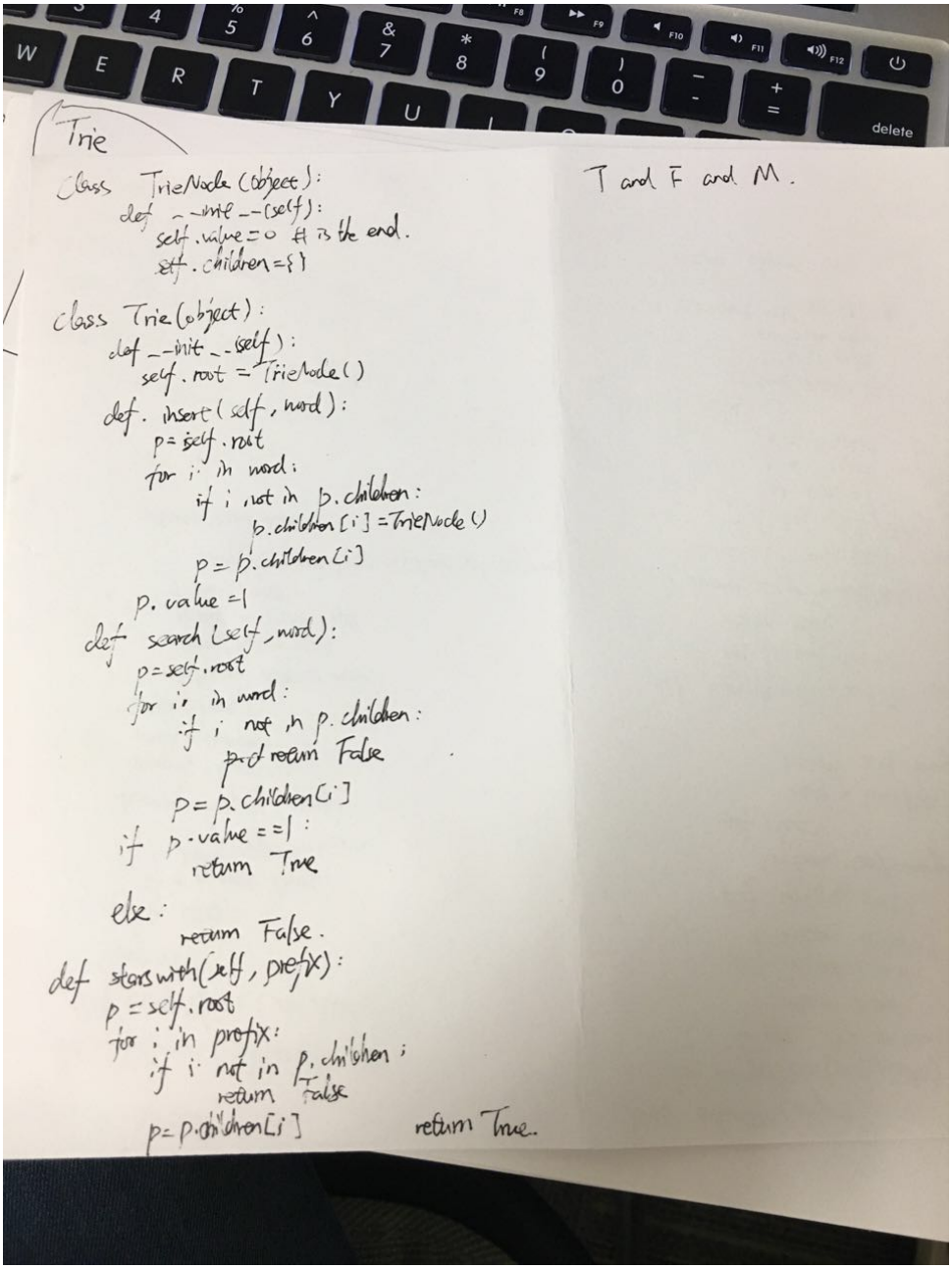
```
def insertFront(self, node):
    self.front.next.prev = node
    node.next = self.front.next
    self.front.next = node
    node.prev = self.front
```

```
def unlink(self, node):
    node.prev.next = node.next
    node.next.prev = node.prev
    node.prev = None
    node.next = None
    return node
```

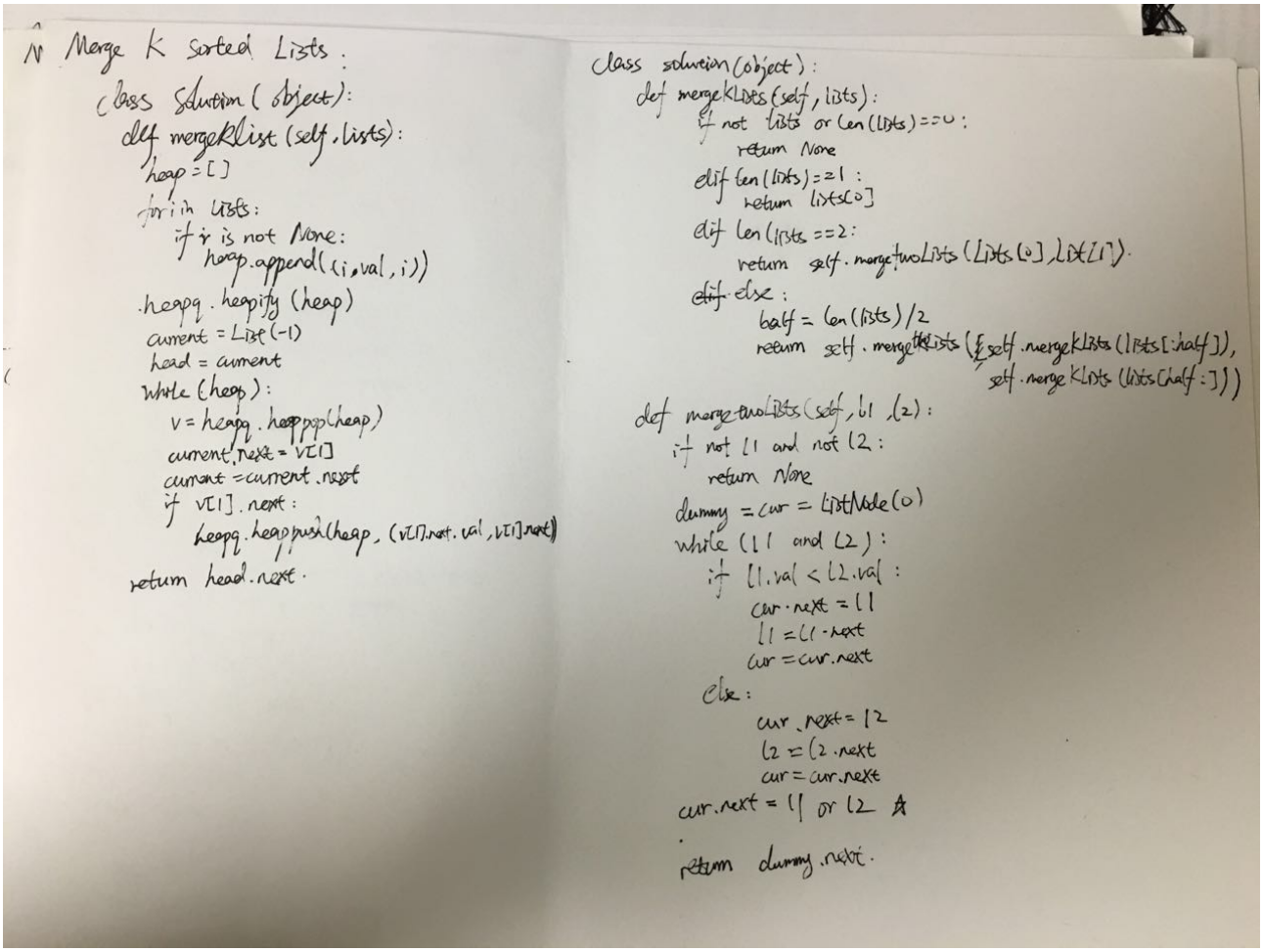
`
LRU cache的两种，注意不要写左边那种 会比较disappointing
implement Trie
可能会考变体：
Analyzing a trie structure and class that would form an autocomplete feature

```
class TrieNode(object):
    def __init__(self):
        """
        Initialize your data structure here.
        """
        self.value=0
        self.children={}


class Trie(object):

    def __init__(self):
        self.root = TrieNode()

    def insert(self, word):
        """
        Inserts a word into the trie.
        :type word: str
        :rtype: void
        """
        p=self.root
        for i in word:
            if i not in p.children:
                p.children[i]=TrieNode()
            p=p.children[i]
        p.value=1


    def search(self, word):
        """
        Returns if the word is in the trie.
        :type word: str
        :rtype: bool
        """
        p=self.root
        for i in word:
            if i not in p.children:
                return False
            p=p.children[i]
```

```
        if p.value ==1:
            return True
        else:
            return False

    def startsWith(self, prefix):
        """
        Returns if there is any word in the trie
        that starts with the given prefix.
        :type prefix: str
        :rtype: bool
        """
        p=self.root
        for i in prefix:
            if i not in p.children:
                return False
            p=p.children[i]
        return True
    def after(self, prefix):
        s=""
        p=self.root
        for i in prefix:
            print i
            if i not in p.children:
                return False
            p=p.children[i]
        while(True):
            if len((p.children))>1:
                return s
            elif len(p.children)==1:
                s+=str(p.children.keys()[0])
                p=p.children[p.children.keys()[0]]
            else:
                break
        return s

# Your Trie object will be instantiated and called as such:
trie = Trie()
trie.insert("somestring")
trie.insert("something")
trie.insert("somestrng")
print trie.search("key")
#print trie.after("some")
print trie.after("somet")
```




Merge K sorted lists 两种做法：



```
N Merge K Sorted Lists.

class Solution(object):
    def mergeKlist(self, lists):
        heap = []
        for i in lists:
            if i is not None:
                heap.append((i.val, i))
        heapq.heapify(heap)
        current = List(-1)
        head = current
        while (heap):
            v = heapq.heappop(heap)
            current.next = v[1]
            current = current.next
            if v[1].next:
                heapq.heappush(heap, (v[1].next.val, v[1].next))
        return head.next.
```

```
class solution(object):
    def mergeKLists(self, lists):
        if not lists or len(lists)==0:
            return None
        elif len(lists)==1:
            return lists[0]
        elif len(lists==2:
            return self.mergetwoLists(lists[0],lists[1])
        else:
            half = len(lists)/2
            return self.mergeKLists([self.mergeKLists(lists[:half]),
                                     self.mergeKLists(lists[half:])])

    def mergetwoLists(self,l1,l2):
        if not l1 and not l2:
            return None
        dummy = cur = ListNode(0)
        while (l1 and l2):
            if l1.val < l2.val:
                cur.next = l1
                l1 = l1.next
                cur = cur.next
            else:
                cur.next = l2
                l2 = l2.next
                cur = cur.next
        cur.next = l1 or l2
        return dummy.next.
```

Maximum subarray以及power(a,b)

Maximum Subarray:

```
def maxSubarray(self, nums):
    m = -sys.maxint - 1
    sum = 0
    for i in nums:
        if (num<0):
            sum = 0
        sum += i
        m = max(m, sum)
    return m
```

```
class Solution(object):
    def maxSubarray(self, nums):
        m = -sys.maxint - 1
        return self.divide(nums, 0, len(nums)-1, m)
    def divide(self, nums, left, right, m):
        mid = (left + right)/2
        if (left > right):
            return -sys.maxint - 1
        ml = self.divide(nums, left, mid-1, m)
        mr = self.divide(nums, mid+1, right, m)
        m = max(ml, mr, m)
        sum = 0
        mlmax = 0
        for i in range(left, mid)[::-1]:
            sum += nums[i]
            mlmax = max(mlmax, sum)
        sum = 0
        mrmax = 0
        for i in range(mid+1, right+1):
            sum += nums[i]
            mrmax = max(mrmax, sum)
        m = max(m, mlmax+mrmax+nums[mid])
        return m
```

Power(a, b)

```
def myPow(self, x, n):
    if n<0:
        return 1.0/self.myPow(x,-n)
    if n==0:
        return 1.0
    if n==1:
        return x
    if n%2:
        return self.myPow(x*x, n/2)*x
    else
        return self.myPow(x*x, n/2)
```

追加几个面经：hello->h3o 就是Unique Word Abbreviation的一部分。

```
def __init__(self, dictionary):
        """
        initialize your data structure here.
        :type dictionary: List[str]
        """
        self.d={}
        for i in dictionary:
            s=''
            if not i:
                continue
            s+=i[0]
            if(len(i)>2):
                s+=str(len(i)-2)
            s+=i[-1]
            if(s not in self.d.keys()):
                self.d[s]=i
            else:
                self.d[s]=''
            del s
```

系统设计：
Add feature to one of your favorite products.
shorten urls

以下是full time面经：
给出一个迭代器，迭代器里存了很多个类R的实例, 类R里只有两个参数，都是String类型，分别代表Parent 和 Child，child只有一个Parent，Parent 可以有任意数量child
按所给示例输出所有的层级关系

eg：
A
  B1

B2
C1
C2
B3
D1
D2
这个题我面HULU的时候也碰到了，不过是manager和employee的层级。 当时建了树外加哈希表

葫芦
做好对应的准备很重要，比如看看他们的engineer blog
面我的全是数据架构组的人，除了常规算法题，还根据经历（因为lz有contribution to Apache开源社区）问了些hdfs，hbase架构/逻辑，这个比算法更有意思点

一轮技术电面：
1. Generate Parentheses. 原题
2. OOD 设计一个poker game。

onsite在LA 不同于前面几个大公司 是关于system 包括了网络负载均衡 分布式等很多方面 而且followup的时候问的非常细 不懂的话基本不可能蒙混过关 会问到一致性模型之类的很具体的东西
coding题目虽然不难 不过有一面是dp题目

两轮四道题
3. 以"／"做分割把路径字符串分割出来
4. 在tree里有 * ，遍历树并打印所有的*

Hulu
第一题是copy一个linkedlist

第二题是给你一个公司员工的信息列表，标明每个人的上司是谁，让你整理出整个公司的结构然后打印出来。
接着onsite，
第一轮先问了一个有关拓扑排序的题目，是将英文单词进行拓扑排序然后输出
第二轮第一个问题是给你两个链表，每个链表里面的元素是线段的起点和终点，让你合并链表，同时合并线段。第二个是个动态规划，比较变态没做出来，题目也记不清了.....
第三轮是设计问题，问一个网站如何设计来针对两个不同国家。
第四轮第一题是实现一个ring buffer，然后第二问记不太清了.....

75分钟，只有一道题，输入一个字符串表示employee信息，用"--"分割每个employee信息，例如
Fred,Karl,Technician,2010--Karl,Cathy,VP,2009--Cathy,NULL,CEO,2007
所以拆开来有三个employee的数据，其中每个数据都是一个四元组，用","分开，分别是name, managere, title, year
Fred,Karl,Technician,2010 (表示employee name是Fred，他的manager是Karl，他的title是Technician，2010入职)
Karl,Cathy,VP,2009
Cathy,NULL,CEO,2007 (NULL说明这个是大boss了，他头上没有别人了)

要求输出公司的结构表，如下
Case #1
Cathy (CEO) 2007
-Karl (VP) 2009
--Fred (Technician) 2010

"-"表示对应的level

解法：处理数据比较麻烦，处理完数据用hashmap之类的建一个graph，用dfs跑一下整个graph就得出结果

实习OA
OA
给一个文件，里面每个entry中是一个员工，有员工姓名，老板姓名，职位。让根据这个文件建一个组织关系图。人的名字都是唯一的。

Onsite
1. 美国人
Stock I, Stock III, Ads Fatigue（给n个slots，让插入一个广告x，保证x之间的gap大于一个值k，用dp秒之，和house robber有点像）

2. 香港人
吃完做题，描述了很多没用的，其实就是飞行棋，掷色子，有的格子可以飞到较远的格子，有的格子会掉回之前的格子，求最短到达终点的步数（原题是地图，梯子，蛇等一堆信息，我觉得用飞行棋概括解释大家都能听懂）。抽象出来就是图的BFS，注意剪枝。

4. 中国人
确定一个字符串里的字符都是另一个字符串里拿出来的。比如aab是aabbcc中的，但aab不是abc中的。

给一个N位数字，让你拿出K个数字，保证得到数字最大，每一位的相对顺序不能变。比如12443，K=3，则结果是123，K=2, 则结果是12. 因为没见过这题，当时没想到贪心法，给了个耐心排序的次优解，要差一些。实际上就是从N位数字中删除N-K个数字，Google删数问题就行了。

**补充内容 (2015-10-21 09:18):**
Ad Fatigue是插入广告x，可以0个或多个，求问一共有多少种插入方法

**补充内容 (2015-10-22 00:38):**
最后一轮国人还问了如果你让你重新设计之前project的架构如何设计，响应式设计的历史由来，为什么要用响应式设计，如何处理不同终端需要不同Js，而Js又很大的情况，最后一问不太懂。

magict42 发表于 2015-11-11 13:47
楼主可以详细说说 Ads Fatigue吗？还是有点迷糊，谢啦！祝面试其他公司顺利！

就是比如3个slot _ _ _
gap是1
你可以这么放
x _ x
x _ _
_ x _
_ _ x
_ _ _
一共5种

王可雪 发表于 2014-11-17 14:47
"求string当中包含dictionary里所有单词的最短substring"這個題dict里是word還是char？要是word感覺挺複雜 ...

The interviewer asked me to find all the "words" but I single-sidedly changed it to char to simplify things a bit.. lol

第二题是求string当中包含dictionary里所有单词的最短substring

**补充内容 (2014-11-13 13:35):**
那道很恶心的print tree的题目要求print成一下那个样子，感觉这题不适合做onsite的interview题，所以感觉那人在黑我

node1
|-node1.1
|    |-node1.1.1
|    |_node1.1.2
|-node 1.2
|    |_node1.2.1
|
|_node...

## Hulu
的程序，写得很绕而且性能很差。面试官先问我这段代码的用途，然后问有什么方法优化，并要求我把代码写在 titanpad 上。接着他问了我第二段代码是做什么的。第二段代码也写得有点复杂，不过可以看出是一个检查有向图里面是否存在环的程序。

面完后还有点时间，我们就聊了聊。面试官是在 Hulu 做支付的，主要用的语言是 Scala（Hulu 里面各个团队用的技术都很自由，可以选择自己喜欢的语言开发，Python Ruby C# Java 都有）。我当时正好在字节社做 iOS 上的应用内购验证，就问了他有没有处理黑卡坏账方面的经验。可惜 Hulu 在 iOS 上只有订阅方式的支付途径，没有 non-consumable product 的相关经验。

coding 题很简单，就是在一个数组里找出两个数，使得他们和为给定的数。写完这个最基本的版本后还有些别的变化，比如如果所有的数都是一定范围内的正整数，这时可以用一个数组统计每个数字的出现频率。

前三轮由来自内容团队、API 团队和架构团队的工程师面试，最后一轮的面试是 CTO 亲自面的。题目也是涵盖了算法、设计和实际的编码，有一轮的问题从设计一个分布式系统开始，讨论了这个系统的身份验证、数据分片、原子性、容错等问题的设计，考察了很多细节的地方，最后还让我写了 SQL，以及如何优化这些查询指令。不过我对 SQL 很不熟悉，每次用都要查手册，当时用了 ActiveRecord 的查询接口代替。

CTO 面完后 HR 进来问了我有没有其他公司的 offer，接着带我逛了一圈 Hulu 的工作场所。Hulu 总部只有 40 位左右的工程师，大家相互之间都很熟，气氛非常好，也比较自由。逛完后 CTO 带我出去买了杯咖啡，在回来的路上给我发了 offer。Hulu 的 package 和 Facebook 的在数额上差不多，但是因为公司性质的问题没法发股票给我，只能用相近的奖金代替。

电话面试：先问了Docker相对于TYPE1 virtualization的优势，然后问了”如何检测分割电视剧片头和实际内容“， 我简单说了HLS标准中的discontuinity tag和DASH中的 manifest 特点，又扯了点场景识别的科研方面的东西。然后coding：“given an array of integers, find the K largest elements"

onsite： 四轮。 第一轮：system design。 设计一个机制，让video streaming的系统能够实时监测问题。问题包括：客户端播放器各种问题，用户家里网络问题，CDN问题，origin问题，streaming server或者transcoding server的问题。。。。。我跟他扯了45分钟客户端report模块的设计，然后还没说完就到时间了。。。。。  这个问题就是面试官自己的工作内容，他应该是完全抱着"讨论研究"的心态面试我的。怎么可能45分钟设计出一个完整的QA机制呢。。。。。

第二轮：面试前先吃饭，边吃边吹牛，面试官似乎对我的经历背景很感兴趣，吃了有将近一个小时。。。。。从为什么申请HULU到中国的高考，到荣光的三国志， Civilization V游戏。。。。就差聊聊HALO里的哪个Cortana最漂亮了。。。。  我觉得我是吹牛时impress他了。system design。设计一个API。允许用户上传和获取文件（GET和POST）。要求系统尽可能redundant. 不允许使用AWS等cloud service。我提出LB + Nginx group + SAN的方法，他和我详细讨论了Nginx的配置文件内容（主要是consistent hashing和proxy），然后跳到了SAN的同步问题（SAN需要时间做duplicate），我的设计中有可能出现文件上传后GET返回404的情况，后来我就懵了

第三轮：longest increasing subsequence 的变体，很简单。不过要先给面试官演示一遍算法，然后再写代码。35分钟左右刚刚够，问了两个问题就结束了。

第四轮：manager面，他先介绍了HULU总部的几个team，各自的工作。然后问了一个问题：given an array of integers, find the Kth largest element. time complexity should be O(n).

patition划分求the K largest elements，平均复杂度是O(N)，最坏时间复杂度仍为O(N^2)

**glassdoor**：

permutations, snake and ladder game, handle race conditions and designing a whole system.

A system design question about how to deal with server failure.

Given code sample, please state what it does. A recursive function calling a utility function. State what the utility and recursive function do. How would you make it better. . Give the Big O for both functions.

Given a string, parse it and return a string array. It's like a tokenizer, but the rules are too...

For exmple, string="abc(edf)hij{klmn}opq[rst]uvw"

The delimitors are (), {}, []. They are in pair. So output array:

["abc", "edf", "hij", "klmn", "opq", "rst", "uvw"]

That's the rule 1. The rule 2 is, if any two consecutive "(" means escaping, that is "((" is actually output char "(". It's not part of the delimitor. Similar to ")", "{", "}", "[", "]".

abc(e))df) => ["abc", "e)df"], since the "))" outpus ")".

Rule 3: if "{" is inside a delimitor pair (), then "{" isn't part of the delimitor. Output it as is.

abc(e{df}}g) => ["abc", "e{df}}g"]

So, parse the given string and assume the given string is always valid and parsable.

I think state machine is a good direction. But I didn't finish it.

two string comparison, hybrid data structure, number base

1. a toy program. complexity. make it better. ( I made a mistake right here. i will keep making mistakes..)

*Given an array of integer, find the smallest k that through +/- at most k for each element , the array can be changed to a strictly increasing array.*