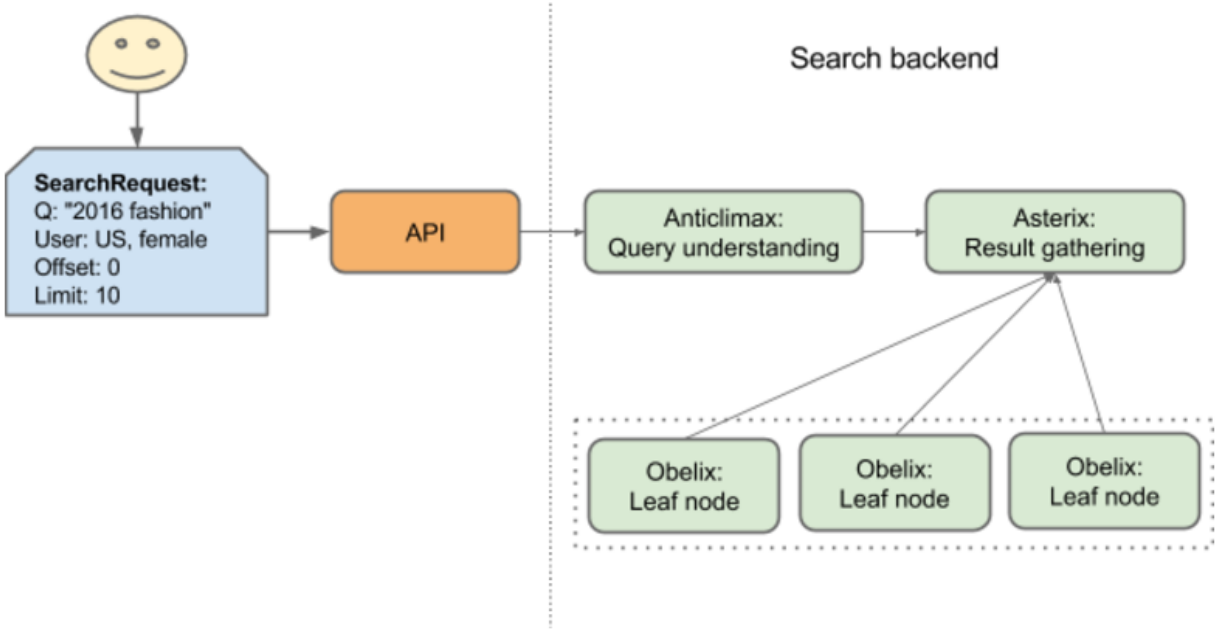
Machine Learning Engineer职位。5论
1论ML design
怎么去Rank Pins (pinterest 的 Feeds)，这个东西非常open ended。发散思维，多想就好。
https://medium.com/the-graph/search-serving-and-ranking-at-pinterest-224707599c92
2论coding
1论 主要问分布式算法的问题
1论午饭



Inside Asterix, there are three major components: cluster client, rerankers and blenders.
The cluster client is a scatter/gather service that distributes search requests to Obelix nodes
a machine learned reranker generates a new ranking score for each Pin based on context features, while a local reranker boosts Pins in a Pinner’s language.
The search results from different clusters are blended using both simple and complex blending logic.

第一轮：http://www.1point3acres.com/bbs/thread-300818-1-1.html
now we have data with 3 properties
user_id, timestamp, action
100, 1000, A
200, 1003, A
300, 1009, B
100, 1026, B
100, 1030, C
200, 1109, B
200, 1503, A
We want to output a graph to visualize it
Graph from input:
|---A (2)
| |---B (2)
| | |---C (1)
| | |---A (1)
|---B (1)
aggregate by user_id:
100 : A->B->C
200:   A->B->A
300: B
1. define data structure and classes
2. construct the graph from logfile
3. print out the graph similar to above
第二题问用一个字符代表pinterest的一个网页， 有一连串的sequence来代表浏览顺序，问用什么data structure来implement。
简单的trieNode

```
def build_log_system(logs):
    user_id, time_stamp, action = zip(*logs)
    user_id = sorted(list(set(user_id)))

    # construct a trie
    root = {'#':'#'}
    for uid in user_id:
        t = root
        for log in logs:
            if log[0] == uid:
                if log[2] not in t:
                    t[log[2]] = {'counter':1}
                else:
                    t[log[2]]['counter'] += 1
                t = t[log[2]]
        t['#']='#'

    # traversal
    def dfs(node, level_so_far=''):
       if node == '#':
            return
        for next_action in node:
           if next_action not in ['counter', '#']:
                print '{}|---{}({})'.format(level_so_far, next_action, node[next_action]['counter'])
                dfs(node[next_action], level_so_far+'|\t')
    dfs(root)
```

第二轮：Deisgn search autocomplete system，问了怎么shard，怎么返回top 10 most frequently searched keywords，怎么update index in realtime，caching。
https://www.youtube.com/watch?v=us0qySiUsGU
第三轮：Path Sum iii (LeetCode 437) 改成n-ary tree。
第四轮：Hit Counter 利口原题
第五轮：Manager聊

```
class nArryTreeNode(object):
    def __init__(self, x):
        self.val = x
        self.neighbor = []

def solution(root, target):
    if not root:
        return 0

    def dfs(root, target, so_far, dp):
        global result
        if root:
            complement = so_far + root.val - target
            if complement in dp:
                result += dp[complement]
            dp.setdefault(so_far + root.val, 0)
            dp[so_far+root.val] += 1
            for node in root.neighbor:
                dfs(node, target, so_far+root.val, dp)
            dp[so_far+root.val] -= 1
        return

    result = 0
    dfs(root, target, 0, {0:1})
    return result

from collections import deque
class HitCounter(object):
    windowLen = 300
    def __init__(self):
        self.hitQueue = deque() # each item is [timeStamp,hitCount]
        self.hitCountInWindow = 0

    def _removeOldHits(self, timestamp):
        while self.hitQueue and self.hitQueue[0][0] <= timestamp - self.windowLen:
            self.hitCountInWindow -= self.hitQueue.popleft()[1]

    def hit(self, timestamp):
        if not ( self.hitQueue and self.hitQueue[-1][0]==timestamp ):
            self.hitQueue.append( [timestamp,0] )

        self.hitQueue[-1][1] += 1
        self.hitCountInWindow += 1

    def getHits(self, timestamp):
        self._removeOldHits(timestamp)
        return self.hitCountInWindow
```

1. 数据结构和算法
给一堆公司门禁的安全日志，内容有时间，门编号，员工名称，是否有权限。根据是否不同的人在相同时间地点的出入记录推测哪些人是朋友，输出top10的朋友
followup：
如果公司很大，跨国公司，不同的工作地点（LA,NY,Beijing,blablabla)的日志，都存在一起，怎么海量日志找出top100

2. Problem Solving 行为树

3. System Design 设计一个滑动拼图小游戏，客户端和服务端的设计

4. Problem Solving (priority queue)
面经题：输入是一堆pin和一个col数，将这些pin放入col个list中返回，填充页面，规则是每次填一个pin的时候找当前像素数最少的那一列填

5. Manager
半小时，前面一半behavior question
Why Pinterest? Why not Amazon or google or facebook?

```
import heapq

class PriortyQueue:
    def __init__(self):
        self._queue = []

    def push(self, item, lst, priority):
        heapq.heappush(self._queue, (priority, lst, item))

    def pop(self):
        return heapq.heappop(self._queue)

def arrage_pins(pins, n_col):
    position = PriortyQueue()
    for i in range(n_col):
        position.push('Col_{}'.format(i), [], 0)
    for pin, name in pins:
        height, lst, name_col = position.pop()
        position.push(name_col, lst+[name], height + pin)
    return position._queue
```

1. coding 45min
2. manager聊天 30min
3. OOD 45分: 设计停车场 github code YouTube
4.coding 45分: 给你一个graph，问他能不能成为一个tree

```
from collections import defaultdict
class Graph():
    def __init__(self, V):
        self.V = V
        self.graph  = defaultdict(list)

    def addEdge(self, v, w):
        self.graph[v].append(w)
        self.graph[w].append(v)

    def isCyclicUtil(self, v, visited, parent):
        visited[v] = True
        for i in self.graph[v]:
            if visited[i] == False:
                if self.isCyclicUtil(i, visited, v) == True:
                    return True
            elif i != parent:
                return True
        return False

    def isTree(self):
        visited = [False] * self.V

        if self.isCyclicUtil(0, visited, -1) == True:
            return False

        for i in range(self.V):
            if visited[i] == False:
                return False

        return True

# Driver program to test above functions
g1 = Graph(5)
g1.addEdge(1, 0)
g1.addEdge(0, 2)
g1.addEdge(0, 3)
g1.addEdge(3, 4)
print "Graph is a Tree" if g1.isTree() == True else "Graph is a not a Tree"
```

第一轮
有一个app，要实现两个方法：updateRating() 和 getTopKRating()，总数N和要求的K都已知，要求getTopK()的时间复杂度O(1)。用一个list和一个priority queue分别存k和n－k个rating，然后不断update。
第二轮
countEvent, 就是问一个event在N秒内被call了多少次。给两个method signature：eventOccured(), getEventCount()帮助实现。用一个Queue，每次在eventOccured()里面把N秒以前的event去掉，然后在队尾加上最新的event，getEventCount()只要看queue当前的size就行。
第三轮 Leetcode 103 有一点不同的是只说tree
第四轮 Leetcode 460

```
from collections import deque
class countEvent(object):
    def __init__(self, N=300):
        self.windowLen = 300
        self.eventQueue = deque() #[timeStamp, hitCount]
        self.eventOccuredInWindow = 0

    def _removeOldCounts(self, timestamp):
        while self.eventQueue and self.eventQueue[0][0] <= timestamp -
self.windowLen:
            self.eventCountInWindow -= self.eventQueue.popleft()[1]

    def eventOccured(self, timestamp):
        if not ( self.eventQueue and self.eventQueue[-1][0]==timestamp ):
            self.hitQueue.append( [timestamp,0] )

        self.eventQueue[-1][1] += 1
        self.eventCountInWindow += 1
        self._removeOldCounts(timestamp)

    def getEventCount(self, timestamp):
        return self.eventOccuredInWindow
```

103 non-binary tree zigzag level order traversal

```
class TreeNode(object):
    def __init__(self, x):
        self.val = x
        self.children = []

def zigzagLevelOrder(root):
    if not root: return []
    res, temp, stack, flag=[], [], [root], 1
    while stack:
        for i in xrange(len(stack)):
            node=stack.pop(0)
            temp+=[node.val]
            if node.children: stack+=node.children
        res+=[temp[::flag]]
        temp=[]
        flag*=-1
    return res
```

Leetcode 460 LFU (Least Frequently Used Cache)

```
import collections
class LFUCache(object):
    def __init__(self, capacity):
        self.remain = capacity
        self.nodesForFrequency =
collections.defaultdict(collections.OrderedDict)
        self.leastFrequency = 1
        self.nodeForKey = {}

    def _update(self, key, newValue=None):
        value, freq = self.nodeForKey[key]
        if newValue is not None: value = newValue
        self.nodesForFrequency[freq].pop(key)
        if len(self.nodesForFrequency[self.leastFrequency]) == 0:
self.leastFrequency += 1
        self.nodesForFrequency[freq+1][key] = (value, freq+1)
        self.nodeForKey[key] = (value, freq+1)

    def get(self, key):
        if key not in self.nodeForKey: return -1
        self._update(key)
        return self.nodeForKey[key][0]

    def put(self, key, value):
        if key in self.nodeForKey: self._update(key, value)
        else:
            self.nodeForKey[key] = (value,1)
            self.nodesForFrequency[1][key] = (value,1)
            if self.remain == 0:
                removed =
self.nodesForFrequency[self.leastFrequency].popitem(last=False)
                self.nodeForKey.pop(removed[0])
            else: self.remain -= 1
            self.leastFrequency = 1 # should be one after adding a new item
```

第一轮：让design一下Pinterest的home feed system。主要讨论如何设计social graph和feed的存储https://medium.com/@Pinterest_Engineering/building-a-smarter-home-feed-ad1918fdfbe3

第二轮：给一个N长度的unsorted array，array中的每个数都是8bit的正数integer，以及M个query，每个query里有个start index和end index， 要求出原array中每个start point到end point的平均数。先说了一下brute force的解法，时间O(N*M)，小哥问能不能优化，给出了用累加array的解法，时间O(N+M)，小哥表示ok。follow up，如果找中数怎么找。这里有点卡住，小哥提示每个数都是8bit的这个条件还没用，想出了bucket sort的解法，说完时间不够了，没有写代码。

中间轮，一个美国小姐姐带着吃。

第三轮：Pinterest主页上有N个column，给一个set of pins，pins有score和length，每次把score最高的pin贴到最短的column上，return List<List<pins>> 表示每个column里的pins。用priority queue做，写完之后follow up：用户的手机屏幕只有M长，如果屏幕的顶点距离主页顶点距离为K，求出能显示出的pins。思路是用数组存一个column中每个pins到顶点的距离，然后找到最上面的和最下面的，再求中间的。小优化是用binary search找start point 和 end point。写了一半，时间不够了，问了点问题就结束了。

第四轮：给一个list of intervals， interval有start， end 和 weight，如果两个interval overlap了，overlap部分的weight相加变成新的interval，start和end point是overlap部分的两个边界。return原list overlap之后的新list of intervals。说了个priority queue的解法
http://www.voidcn.com/article/p-yvscwuip-d.html

实现文本编辑器, 支持 append, insert, delete; follow up: 数据结构怎样提高搜索的效率

LeetCode 14 Longest Common Prefix

```
 def longestCommonPrefix(self, strs):
        """
        :type strs: List[str]
        :rtype: str
        """
        if not strs:
            return ""
        shortest = min(strs,key=len)
        for i, ch in enumerate(shortest):
            for other in strs:
                if other[i] != ch:
                    return shortest[:i]
        return shortest
```

LeetCode 679 24 Game not brackets

```
from fractions import Fraction
from operator import add, sub, mul, truediv

def game24(nums):

    def dfs(nums):
        if len(nums) == 1:
            return nums[0] == 24

        x = nums.pop(0)
        y = nums.pop(0)

        res = False
        for op in [add, sub, mul, truediv]:
            if op in [add, mul]:
                res = res or dfs([op(x, y)] + nums)

            elif op == sub:
                res = res or dfs([op(x, y)] + nums)
                res = res or dfs([op(y, x)] + nums)

            else:
                if y != 0:
                    res = res or dfs([op(x, y)] + nums)

                if x != 0:
                    res = res or dfs([op(y, x)] + nums)

        return res

    return dfs([Fraction(num) for num in nums])
```

design:
- Design ads creation backend
- 从一个很大的文件里Sample 10k log; 如果分成很多不同大小的文件放在很多机器上怎么办；map reduce 怎么解决

第一轮：
给一个字符串， 给一个字典，字典里是不重复的单个字母。问这个字符串中最短的substring，条件是包含所有字典中的字母。
应该是leetcode的原题了。
做法是two pointer + hash

```
from collections import defaultdict
def find_min_window(S, T):
    left, right = 0, 0
    c = defaultdict(int)
    for ch in T:
        c[ch] += 1
    min_str = None

    while right <= len(S):
        if all(map(lambda x: True if x<= 0 else False, c.values())):
            if min_str is None or right-left <= len(min_str):
                min_str = S[left:right]
            if S[left] in c:
                c[S[left]] += 1
            left += 1
        else:
            if right == len(S): break
            if S[right] in c:
                c[S[right]] -= 1
            right += 1
    return min_str if min_str else ''
```

第二轮：
给一个sorted linked list，建一个balanced BST。
也是leetcode的原题 109
https://discuss.leetcode.com/cat ... -binary-search-tree

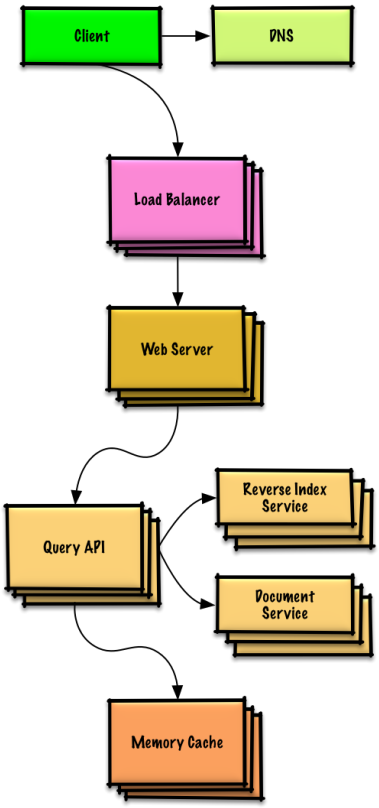
```
class TreeNode(object):
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None

class ListNode(object):
    def __init__(self, val):
        self.val = val
        self.next = None

# top-down approach, O(n*logn)
def sortedListToBST(head):
    if not head:
        return
    if not head.next:
        return TreeNode(head.val)
    dummy = ListNode(0)
    dummy.next = head
    slow, fast = dummy, head
    while fast and fast.next:
        slow = slow.next
        fast = fast.next.next
    root = TreeNode(slow.next.val)
    root.right = sortedListToBST(slow.next.next)
    slow.next = None
    root.left = sortedListToBST(head)
    return root
```

第三轮：
设计一个图片搜索系统 search engine
准备面试的时候 看到过类似的题目 也就比较顺利的和面试官好好交流了一下
中间涉及到的算法知识有 hash 和 toplogical sort
强推下面的github用于准备system design
https://github.com/donnemartin/system-design-primer

https://github.com/donnemartin/system-design-primer/blob/master/solutions/system_design/query_cache/README.md



第四轮：
机器学习 overfitting, bagging, randomforest
然后就是设计一个组成语句的API(马尔科夫链)
例子:
"one two three four"
"two three four"
"two five four"
那以"two" 开头的概率是2/3 "one" 开头的概率是1/3
那我们先选开头的这个词，
如果选到了two， 那在two这个阶段上再往后看，three跟着two的概率是2/3
five跟着two的概率是1/3
如果选到了three，.....
按照这样的模式直到结束。

```
import random

def wordsum(wordict):
    sum = 0
    for key, value in wordict.items():
        sum += value
    return sum

def randomWord(wordict):
    randindex = random.randint(0, wordsum(wordict))
    for key, value in wordict.items():
        randindex -= value
        if randindex <= 0:
            return key

def sentenceMarkovGenerator(sentence_lst, N):
    begin = {}
    nextword = {}
    prev = ''
    for sentence in sentence_lst:
        words = sentence.split(' ')
```

```
    for i in range(len(words)):
        if i == 0:
            if words[i] in begin:
                begin[words[i]] += 1
            else:
                begin[words[i]] = 1
            prev = words[i]
        else:
            if prev in nextword:
                if words[i] in nextword[prev]:
                    nextword[prev][words[i]] += 1
                else:
                    nextword[prev][words[i]] = 1
            else:
                nextword[prev] = {}
                nextword[prev][words[i]] = 1
            prev = words[i]

result = []
for i in range(N):
    if i == 0:
        result += [randomWord(begin)]
    elif result[-1] in nextword:
        result += [randomWord(nextword[result[-1]])]
    else:
        break
return result
```

1， system design，比较general 的 问题，数据库，mapreduce之类的
2， machine learning，recommender system 和 ads targeting
3， 给一个很大的string， 和一些比较短的string， 问是否是subsequence（不是substring，可以不连续）
4， binary tree， longgest path

find first occurrence https://www.youtube.com/watch?v=OE7wUUpJw6I

3: LeetCode 392

```
def isSubsequence(s, t):
    # find minimum index that is greater than target, return -1 if not exist
    def bs(nums, l, r, target):
        if l > r:
            return -1
        if l == r and nums[l] > target:
            return nums[l]
        else:
            mid = int((l+r)/2)
            if nums[mid] < target:
                l = mid + 1
                return bs(nums, l, r, target)
            elif nums[mid] > target:
                r = mid
                return bs(nums, l, r, target)
            else:
                if mid + 1 <= len(nums) - 1:
                    return nums[mid + 1]
                else:
                    return -1
    #main function
    def check(s, t):
        if s == "":
            return True
        dic = {}
        for ind, char in enumerate(t):
            if char in dic:
                dic[char].append(ind)
            else:
                dic[char] = [ind]
        pos = -1
        for char in s:
            if char not in dic:
                return False
            else:
                ref = dic[char]
                loc = bs(ref, 0, int(len(ref)-1), pos)
                if loc > pos:
                    pos = loc
                else:
                    return False
        return True
    return check(s,t)
```

Diameter of Binary Tree

Given a binary tree, you need to compute the length of the diameter of the tree. The diameter of a binary tree is the length of the **longest** path between any two nodes in a tree. This path may or may not pass through the root.

```
# Definition for a binary tree node.
# class TreeNode(object):
#     def __init__(self, x):
#         self.val = x
#         self.left = None
#         self.right = None
# time complexity: O(V+E)
# space complexity: O(bm) b: branch m: maximum length of any path
class Solution(object):
    def diameterOfBinaryTree(self, root):
        """
        :type root: TreeNode
        :rtype: int
        """
        self.ans = 0

        def depth(p):
            if not p: return 0
            left, right = depth(p.left), depth(p.right)
            self.ans = max(self.ans, left+right)
            return 1 + max(left, right)

        depth(root)
        return self.ans
```

1. domain knowledge。有没有什么用的熟的db啊？之后让我设计一个graph db，backend用mysql。按照TAO答的，问了很多API和schema的设计。

2. problem sovling，同胞面的，ranking system的index设计，按照fb的设计来答的，聊的很愉快。

3. 问了很多memcached和rocksdb的细节，很细的细节，细到这个锁该怎么拿，以及group commit具体怎么实现。

```
def get_user(self, user_id):
    user = cache.get("user.{0}", user_id)
    if user is None:
        user = db.query("SELECT * FROM users WHERE
user_id = {0}", user_id)
        if user is not None:
            key = "user.{0}".format(user_id)
            cache.set(key, json.dumps(user))
    return user
```

4. 找duplicate file

```
class Solution(object):
    def findDuplicate(self, paths):
        """
        :type paths: List[str]
        :rtype: List[List[str]]
        """
        M = collections.defaultdict(list)
        for line in paths:
            data = line.split()
            root = data[0]
            for file in data[1:]:
                name, _, content = file.partition('(')
                M[content[:-1]].append(root + '/' + name)

        return [x for x in M.values() if len(x) > 1]
```

5. system design，设计caching，有一点类似fb的memcached的paper

题目和Pinterest很相关问怎么怎么创建一种data structure，可以表述对于一个pin，用户和用户间repin的关系。一开始忽略了log是按照时间先后顺序进来的，一开始思路有点混乱，一会bfs，一会dfs的，不过20多分钟之后也写完了。写完之后小哥说，有没有简单点的方法，然后我就盯着log看了一下，发现不存在先创建很低下的subtree再创建上层的node，关键点还是在于log是根据时间先后放进来的，就知道怎么做了。写完之后被他挑了一个bug，不过他一说就改掉了。写完之后好像小哥对java的Map<String, Person> map = new HashMap<String, Person>() 这个很感兴趣，就问为什么前面不直接写HashMap，我就扯了扯java的polymorphysm， 以及这样用interface在stack上创建object的好处，然后时间就到了

和一个在growth team的engineer（听口音好像是法国人）做了cultural fit的interview。她一进来就说这只是cultural fit, 没什么大问题..之类的，让我放松不少，更像是一个behavioral interview。就讨论了下一些scenario，怎么处理之类的。我提到到了怎样提高Pinterest在男性用户中得到青睐， 比如多放一些男性用户repin过的pin，来让我们感兴趣。她还问如果和别的组员意见不一怎么办，我就说可以做下用户测试，比方说iOS的test flight，让1000个用户采取我的方式，另外1000个用户采取别的方式，看哪个方式比较好. stats speak louder than words

system design，具体讨论了下P家每个pin背后都有一个url，以及需要进行web crawling来得到关于这个pin的信息，设计一个这样的系统，有些什么tradeoff等等问题。

出了leetcode上一道类似找最大triangle的题目，就写了code，然后说有没有更优解，于是我就顿时卡住了，因为只做过暴力解... 于是她提示说用DP，然后再根据多一点提示才想出了DP的办法，可是她不要求写code，只是想要说思路，就跟她说了说思路，她说ok。从头到尾她似乎就不怎么care coding的感觉。

```
def largestTriangleArea(p):
        return max(0.5 * abs(i[0] * j[1] + j[0] * k[1] + k[0] * i[1]- j[0] * i[1] - k[0] * j[1] - i[0] * k[1])
            for i, j, k in itertools.combinations(p, 3))
```

1.1 CanWin. 给一个数组比如 [2,3,0,3,5]和一个index比如4，从这个index每次可以向左或者向右跳ary[index]的距离， 如果能跳到值为0的index则返回true。

1.2 Merge k sorted list. 比较麻烦的是要我自己实现一个heap， 写出push（）和pop（）两个方法。

```
def mergeKLists(self, lists):
    current = sentinel = ListNode(0)
    lists = [(i.val, i) for i in lists if i]
    heapq.heapify(lists)
    while lists:
        current.next = heapq.heappop(lists)[1]
        current = current.next
        if current.next:
            heapq.heappush(lists, (current.next.val, current.next))
    return sentinel.next
```

1.3 设计一个ACL serevice。 https://en.wikipedia.org/wiki/Access_control_list

2 系统设计。 设计个dashboard可以显示成千上万台机器的qps, 90%latency 和 request success rate in nearly real-time. 如果要追求精确度怎么办？ 答案是牺牲real time这个要求。。。

3 系统设计。 和2有点类似的题目， 在计算percentile的时候， 需要把每个数据都发送到aggregator， 如果单机数据量太大没法发送怎么办？ 答案是牺牲精确度使用histogram。 然后一堆followup。 --- 所有的设计其实都是tradeoff， 一切都是算计

4. 把以前做的某个项目详细过一遍。

5. Culture fit。 有没有项目失败的时候， 和老板/同事合不来怎么办, etc.

```
class Heap(object):
    def __init__(self):
        self.heapList = [0]
        self.currentSize = 0

    def swap_up(self, i):
        while i // 2 > 0:
            if self.heapList[i] < self.heapList[i // 2]:
                self.heapList[i], self.heapList[i // 2] = self.heapList[i // 2], self.heapList[i]
            i = i // 2

    def push(self, k):
        self.heapList.append(k)
        self.currentSize += 1
        self.swap_up(self.currentSize)

    def pop(self):
        min_val = self.heapList[1]
        self.heapList[1] = self.heapList[-1]
        self.heapList.pop()
        self.currentSize -= 1
        self.swap_down(1)
        return min_val

    def swap_down(self, i):
        while (i * 2) <= self.currentSize:
            min_index = self.find_min_index(i)
            if self.heapList[i] > self.heapList[min_index]:
                self.heapList[i], self.heapList[min_index] = self.heapList[min_index],
self.heapList[i]
            i = min_index

    def find_min_index(self, i):
        if i*2+1 > self.currentSize:
            return i*2
        else:
            if self.heapList[i*2] < self.heapList[i*2+1]:
                return i*2
            else:
                return i*2+1
```

1. 印度小哥
问了一个queue的问题 特定time window里面求事件发生的个数 如何多次query?? 数据特别大的话怎么办

2.国人小哥
求一个数的factor的集合 input： 12 output: {{2, 6}, {3, 4}, {2, 2, 3}} 结果最后忘记dedup了

3.设计题 设计Twitter 问怎么search by tags 感觉答的他不是很满意 不知道是不是要inverted table之类的

4. Lunch 和 HM

5. 白人小姐姐 涉及两个api getEventNUm() 在过去特定的time window多台机怎么处理?

2. LeetCode 254

```
# Time:  O(nlogn)
# Space: O(logn)


    def getFactors(n):
        result = []
        factors = []
        getResult(n, result, factors)
        return result

    def getResult(n, result, factors):
        i = 2 if not factors else factors[-1]
        while i <= n / i:
            if n % i == 0:
                factors.append(i);
                factors.append(n / i);
                result.append(list(factors));
                factors.pop();
                getResult(n / i, result, factors);
                factors.pop()
            i += 1
```

coding第一轮是论坛里常见面经 behavior tree。 后来打印我先写了level order traversal, 然后interviewer问这样怎么知道parent child 关系。好吧，我说那不能，那就写dfs吧。迅速写了，她说ok. by the way， PIN 家可以用电脑打，我因为面试一直是白板，也就干脆不搞电脑了。万一用apple不顺，反而窘

第二轮是design， notification system。 聊得很开放。也不知道我答到点子没。我自己就说了data model，还有cache，还有具体cache的数据结构 然后吃饭。

后面一轮coding,题目是combination of phone number 和 Tire 结合的变形题。给一个单词库，给个phone number，让你确定是否有个phone number代表的字母组合在单词库里，并且输出在单词库的字母组合

最后coding是一维数组每个index下的数字代表能向左或向右跳的步数，问给了起始位置，是否能跳到target 位置

```
def can_win(board, index):
    flag = 1
    visited = [False]*len(board)
    while True:

        if board[index] == 0:
            return True
        if 0 <= (index + flag * board[index]) < len(board):
            index += flag * board[index]
            if visited[index]:
                return False
            else:
                visited[index] = True
        else:
            flag *= -1
            if 0 <= (index + flag * board[index]) < len(board):
                index += flag * board[index]
                if visited[index]:
                    return False
                else:
                    visited[index] = True
            else:
                return False
    return False
```

1. Design homefeed
2. Longest increasing path in the binary tree
3. Number of islands 变种，要求输出每个Island的周长
4. Behavior/Project
5. Alien Dictionary 变种: 1. build graph 2. topological sort

Longest increasing path in the binary tree

```
class Node:
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None

# Returns the maximum consecutive path length
def maxPathLenUtil(root, prev_val, prev_len):
    if root is None:
        return prev_len
    curr_val =  root.val
    if curr_val == prev_val +1 : # consecutive, > for increasing
        return max(maxPathLenUtil(root.left, curr_val, prev_len+1),
                   maxPathLenUtil(root.right, curr_val, prev_len+1))
    newPathLen = max(maxPathLenUtil(root.left, curr_val, 1),
                     maxPathLenUtil(root.right, curr_val, 1))
    return max(prev_len , newPathLen)

# A Wrapper over maxPathLenUtil()
def maxConsecutivePathLength(root):
    # Return 0 if root is None
    if root is None:
        return 0
    return maxPathLenUtil(root, root.val -1 , 0)
```

Perimeter of the island

```
class Solution(object):
    def islandPerimeter(self, grid):
        """
        :type grid: List[List[int]]
        :rtype: int
        """
        if not grid:
            return 0

        def sum_adjacent(i, j):
            adjacent = (i + 1, j), (i - 1, j), (i, j + 1), (i, j - 1),
            res = 0
            for x, y in adjacent:
                if x < 0 or y < 0 or x == len(grid) or y == len(grid[0]) or grid[x][y] == 0:
                    res += 1
            return res

        count = 0
        for i in range(len(grid)):
            for j in range(len(grid[0])):
                if grid[i][j] == 1:
                    count += sum_adjacent(i, j)
        return count
```

1.longest increasing sequence in the binary tree
2.系统设计 inverted index。。
3.distributed system
4.min stack，要有popmin-》leetcode原题

系統設計：
設計一個餐廳推薦系統 (類似Yelp)，假設餐廳有textual features (name, description), image和location。使用者訊息有query, location, etc

午餐 with HM

ML系統設計：
設計一個ad ranking系統。中間交雜著問了一些ML概念，像是ROC, precision/recall，model evaluation等等

coding 1：
next permutation

coding 2：
有一道單選題，有L個選項，拿去問N個人，把大家的答案存在一個array，這個array的size會等於N，且array element可能的值為 {1, 2, ..., L}

求大家對問題的答案有共識的概率，如果不用概率closed-form解，要怎麼寫code估計這個概率?

有共識的定義如下：
如果有大於等於N/2個人，給了一樣的答案，則定義為有共識

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=371197&highlight=pinterest%2Bonsite

1. System Design/ ML. 本来应该要考SD的，结果一上来看了我的背景说，我们直接考machine learning design吧，然后就是问设计pinterest的推荐系统。

2. Hiring manager，主要就是谈为什么Pinterest，之前做过的project等等。

3. Engineer manager，另外一个manager，主要就是随便聊，因为是lunch，聊得还挺开心的，一直鼓励我好好面，虽然最后挂掉了。

4. 又一轮的ML的问题，主要是问了各种cold start的问题怎么做推荐。

5. LC 698的变形，输出的结果是分组的情况。没有做到bug free，中间进行了debug，最后run了两个test case，时间到。

6. 总共有三小问，第一问：给定一个很长的数组N，每个数字都是0-255的数值，和各种pair的index的数组M （可能是[[1,100],[5,1000],etc]）。M的长度最大是N*(N-1)/2. 问求M中所有pair的mean。用个presum，效率是O(M+N). 第二问：相同的数组，求M中所有pair的median，在预处理的时候给每一位计算当前总共有多少个0，多少个1，。。多少个255. 每一位用256的数组存储。效率O(M+N)。 第三问：当N特别大的时候，我们无法存进内存中，给定一个压缩指标K, K<<N， 问怎么在预处理的时候进行压缩，同时效率还是O(M+N). 答案是每次隔K个进行存储，这样在查找的时候每次需要K次，但是K是常数，所以效率还是O(M+N).

LC 698 partition to K equal sum subsets

```
class Solution(object):
      def canPartitionKSubset(self,  nums, k):
            # trivial case one subset
        if k==1: return True
        # trivial case, k must be k<=n
        n= len(nums)
        if k>n: return False
        # k*target = sum(nums)
        total = sum(nums)
        if total%k: return False

        target = total/k
        seen = [0]*n
        # speeds things up, as larger numbers are tried first if its not possible
        # to get k subsets we will know sooner
        nums.sort(reverse=True)

        def dfs(k,index,current_sum):
```

```
            # trivial, one group
            if k==1: return True
            # found one group, need more k-1 groups
            if current_sum == target:
                return dfs(k-1,0,0)
            # group can start with any number
            for i in range(index,n):
                # if we have not tried it before, and adding it
                # to current sum does not exceed target then
                if not seen[i] and current_sum+nums[i]<=target:
                    # we have seen it
                    seen[i]=1
                    # recursively build group from i+1
                    if dfs(k,i+1,current_sum+nums[i]):
                        return True
                    seen[i]=0
            return False

        return dfs(k,0,0)
```

第一轮 国人大哥 culture fit
问了简历, Why Pinterest, Why you, Your interest ...

午饭 美姐姐 在G很多年PM跳槽 刚入职没多久 随便聊一聊 兴奋到没什么胃口吃饭 不过Pinterest家伙食还是不错的 包午餐和晚餐

第二轮
简单聊两句 开始做题
题目是给两个json 输出两个的合并 合并的结果不包括两个json的intersection 写完了之后follow up了怎么improve 怎么test

第三轮 美哥
code reading和architecture 所以咱们先来reading 后面再design一个game
reading 一些function 让你说出这些function是干什么的
边看code 边说思路
design OOP 是 贪食蛇 pseudo code就可以

第四轮
有log file 记录了 user, time, platform 和 一些 operations 每个operation是递进的关系 要输出每一步没有进行到下一步的百分比 和 进行到最后一步的占总人数的百分比

1. System Design: Design Pinterest
2. Coding: LC76
3. Project
4. Coding shortest path between 2 nodes in the tree. LZ用的LCA 然后Heigh(p)+heigh(q)-2*heigh(LCA)
5. System Operation （这轮是DE专属的 SE应该是3轮coding） 找了个SRE来面我 我当时就GG了。。。。 说的各种tool一个没听过 说如何发现一堆cluster里某个box坏了 假如告诉你这个box坏了 你怎么trouble shoot云云 反正LZ只会卖萌跟说看log 各种SRE技能点全0 （靠妖啊！面毛SRE啊）

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=192500&highlight=pinterest%2Bonsite

```
class Node:
    def __init__(self, data):
        self.data = data
        self.right = None
        self.left = None

def pathToNode(root, path, k):
    # base case handling
    if root is None:
        return False
    path.append(root.data)
    # See if the k is same as root's data
    if root.data == k :
        return True
    # Check if k is found in left or right
    # sub-tree
    if ((root.left != None and pathToNode(root.left, path, k)) or
            (root.right!= None and pathToNode(root.right, path, k))):
        return True
    # If not present in subtree rooted with root,
    # remove root from path and return False
    path.pop()
    return False

def distance(root, data1, data2):
    if root:
        path1 = []
        pathToNode(root, path1, data1)
        path2 = []
        pathToNode(root, path2, data2)

        # iterate through the paths to find the
        # common path length
        i=0
        while i<len(path1) and i<len(path2):
            if path1[i] != path2[i]:
                break
            i = i+1

        return (len(path1)+len(path2)-2*i)
    else:
        return 0
```

1. 有一个function A，会被callback访问到，让实现一个funciton，可以统计过去N秒
这个A被call了几次。 经典题，circular array统计每秒call的次数
顺利写出来，不过提醒了一个bug
2. 经历丰富的国人大哥，一看就是大牛。给一个蹦了的jobID，让找出所有depend on
这个ID 的其他job。 实质就是图的遍历， BFS
3. data structure, add(), delete(), getRandom() in O(1), 非常老题，三哥跟一个白姐
4. culture fit，吃午饭忘了时间我迟到了15分钟
5. system design，follow feature, 就是谁关注谁。这个扯了一堆无法在此reproduce所有细节

http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=169765&highlight=pinterest%2Bonsite
http://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=158818&highlight=pinterest%2Bonsite