My notes on leetcode

Tuesday, June 16, 2015

发信人: fbrefer (fbrefer), 信区: JobHunting
标 题: facebook面试准备
发信站: BBS 未名空间站 (Tue Jul 22 11:01:17 2014, 美东)

关于面试流程
社招的话
电面1-2轮，一般就是coding
onsite一般是4轮，2轮coding，1轮design，1轮behavior+coding
校招的话，那轮design也变成coding了

关于准备
1) algo/coding
建议大家刷一下leetcode，基本上cover到了大多数常见面试题，而且有可能碰到原题。需要注意的是，仅仅解出来，做到bug free可能是不够的。代码的质量和速度也非常重要。网上有一些别人给出的答案可以参考，尽量做到代码简洁清晰。速度上leetcode上所有题都做到10分钟以内写完。

2) design
解这种题是个*交流*的过程，或者说是给出方案然后获取反馈的不断循环的过程。
一般的流程：
首先你要问清楚requirement；
然后可以讲一下high level architecture，就是分成哪几个component，互相之间如果interact，在白板上画一画；
之后面试官可能会让你深入某个component detail讨论；
也有可能变换requirement让你重新设计

另外，f家还喜欢让你估算机器之类的，做一些back-of-envelopme calculation。所以最好对一些计算机相关的基本常数，fb的用户量等等有个大概的了解。

准备的时候建议看看fb的design高频题。一方面有可能面试的时候刚好碰到这几个topic，另一方面其实很多design都是相通的。
之前有个帖子讲这个，原帖已经被删了，这儿有个备份
http://blog.csdn.net/sigh1988/article/details/9790337

另外补充一点我收集的材料

a) 首先你可以从整体上了解一下facebook的architecture
http://www.quora.com/Facebook-Engineering/What-is-Facebooks-arc
http://www.ece.lsu.edu/hpca-18/files/HPCA2012_Facebook_Keynote.
http://www.quora.com/Facebook-Engineering/What-have-been-Facebo
除了下面给出的一些资料，fb engineering page里还有很多不错的内容
https://www.facebook.com/Engineering

b) news feed
这里有个talk
http://www.infoq.com/presentations/Facebook-News-Feed
对应的slides
http://readme.skplanet.com/wp-content/uploads/2012/11/0-3_Faceb
还有一些quora上的讨论
http://www.quora.com/Activity-Streams/What-are-the-scaling-issu
http://www.quora.com/What-are-best-practices-for-building-somet
http://www.quora.com/What-is-the-best-storage-solution-for-buil

c) facebook chat
这里有两个notes，其中第二个里面还有相应的tech talk links
https://www.facebook.com/notes/facebook-engineering/facebook-chat/14218138919
https://www.facebook.com/notes/facebook-engineering/chat-stability-and-

Labels

backtracking bfs binary search concept dfs dp graph interval interview kmp math merge string system tree trie two pointers

About Me

Bin Mu

G+ Follow 23

View my complete profile

Blog Archive

- ▼ 2015 (22)
  - ▼ June (4)
    - 发信人: fbrefer (fbrefer), 信区: JobHunting 标 题: faceb...
    - 发信人: beanbun (豆包), 信区: JobHunting 标 题: 回报本版，前段时间骑...
    - 发信人: Dreamer (不要问我从哪里来), 信区: Dreamer标题: 不刷题进Goog...
    - 发信人: beidapig (做人要谦虚), 信区: JobHunting 标 题: 报F和G的o...
  - ► April (5)
  - ► March (4)
  - ► February (5)
  - ► January (4)
- ► 2014 (202)

scalability/51412338919

d) typeahead search & graph search
关于typeahead search的tech talk和notes
https://www.facebook.com/video/video.php?v=432864835468
https://www.facebook.com/note.php?note_id=365915113919
https://www.facebook.com/note.php?note_id=389105248919

关于graph search的paper, tech talk, notes。其中paper很值得一看。
http://db.disi.unitn.eu/pages/VLDBProgram/pdf/industry/p871-cur
https://newsroom.fb.com/Photos-and-B-Roll/4362/Graph-Search-Whiteboard
https://www.facebook.com/note.php?note_id=10151240856103920
https://www.facebook.com/note.php?note_id=10151347573598920
https://www.facebook.com/note.php?note_id=10151361720763920
https://www.facebook.com/note.php?note_id=10151432733048920
https://www.facebook.com/note.php?note_id=10151755593228920

e) facebook messages
两个tech talks
http://www.youtube.com/watch?v=XAuwAHWpzPc
http://www.infoq.com/presentations/HBase-at-Facebook
以及eng notes
https://www.facebook.com/note.php?note_id=10150148835363920
https://www.facebook.com/note.php?note_id=10150162742108920

f) photo storage
相关的papers和notes
https://www.usenix.org/conference/osdi10/finding-needle-haystack-facebooks-photo-storage
https://www.usenix.org/legacy/events/osdi10/tech/full_papers/Beaver.pdf
https://www.usenix.org/legacy/events/osdi10/tech/slides/beaver.pdf
https://www.facebook.com/note.php?note_id=76191543919

g) social graph data store
相关的note, video, paper
https://www.facebook.com/notes/facebook-engineering/tao-the-power-of-the-graph/10151525983993920
https://www.usenix.org/conference/atc13/technical-sessions/presentation/bronson
http://www.cs.cmu.edu/~pavlo/courses/fall2013/static/papers/117

h) tiny URL
这里有一些讨论
http://n00tc0d3r.blogspot.com/2013/09/big-data-tinyurl.html
http://stackoverflow.com/questions/742013/how-to-code-a-url-sho
http://stackoverflow.com/questions/3376163/what-are-the-things-

i) POI
参考这里
http://www.slideshare.net/mmalone/scaling-gis-data-in-nonrelati
http://www.mitbbs.ca/article_t/JobHunting/32476139.html

3) behavior，建议大家了解一下fb的culture，准备一下常见的behavior questions，面试之前rehearsal一下。

最后面试临近的时候，可以再刷刷面经，找找感觉。像glassdoor, mitbbs/jobhunting, careercup，这些上面就有很多。

Posted by Bin Mu at 2:22 PM No comments:

Labels: system

Sunday, June 14, 2015

发信人: beanbun (豆包), 信区: JobHunting
标 题: 回报本版，前段时间骑驴找马FGU等公司offer面经总结【已更新FGU】
关键字: 面经
发信站: BBS 未名空间站 (Sat Jun 13 17:27:31 2015, 美东)

前段时间骑驴找马终于告一段落，感觉本版的技术贴和面经贴帮助非常之大，也非常感谢共享资源的各路大牛。希望提供一些信息和个人感受给还在找工的童鞋，有帮助最好，但是毕竟本人资历尚浅，如果有不对的地方也请轻喷。

背景：
ms毕业不到两年

主要申请公司：
offer：facebook，google，uber，palantir，sumo logic，walmartlab，yahoo，amazon，apple
reject：dropbox

主要几个包裹：
U： 145k base + 25k股 RSU
F： 150k base + 40k signon + 10%bonus + 260k美元 RSU
W： 165k base + 50k signon + 20%bonus + 35k美元 RSU每年（这个略复杂，相当于每年35k美元RSU的refresh，但是每次refresh分四年给）

再上各个公司的面经和感受：

Yahoo：
最早面的公司，面的是Flurry Team，Yahoo去年收购的一家在城里的小公司，所以不一定有代表性。因为re-org我两个月之后才拿到offer，中间还给我match到其他team几次，Yahoo比较动荡，个人也不看好。

电面：
和director聊了有两个小时，无coding，问了很多之前project内容和hadoop相关的内容。
最后讨论了一道design，如何设计distributed key-value store，因为他们主要用HBase。

Programming Test:
Validate Soduko Solution，从文件读solution，尽量用production标准写程序。

Onsite：
五轮Onsite没有coding，全是问实际问题怎么解决和design。
1. 如何设计一个priorityqueue service，client可以submit job request然后server按照priority执行
2. 需要一个key-value store with 1M qps，most read，1ms 99% latency，如果用HBase的话会有什么问题，怎么解决
3. 给很多整数，如何用mapreduce找median，如果是很多float数，可以有一定的误差，如何找
4. Programming Test的扩展，如果soduku matrix非常之大怎么做
然后还有一大堆针对hadoop的各种情况下怎么optimize的问题

onsite完了之后他们director说very positive，然后就开始re-org两个月。Flurry做的东西其实挺有意思，mobile analytics platform #1，我感觉他们engineer人很nice，水准也非常不错，可惜没缘分。

##
#####
##########

Apple：
练手公司1，Apple可以同时面很多组，每个组有各自的recruiter。我把简历递了之后陆续有10个组联系我，然后每个组基本上都是onsite之前两轮phone，一开始没经验联系了4个组后来发现实在体力吃不消，光电面就8轮。最后3个组要onsite，这里我犯了一个错误，告诉他们我在面其他的组，一旦他们知道你在面其他的组就不跟进了，打死不回email。所以最终我只onsite了一个组。

电面：
1.给平面一堆点，把所有在同一条直线上的点group在一起，求出所有的group
2.一种encoding的方法，如果一个byte第一个bit是0，比如 00000000，那它自己表示一个字符，如果一个byte第一个bit是1，比如 10000000，那它和它后面紧跟的byte表示一个字符，现在给一个byte array，判断最后一个字符是一个byte还是两个byte组成。

3.parse message from byte stream，message format是前4个bytes组成的int值表示message的长度L，然后后面连续的L个byte是message真正的内容，每个message都是这样表示，需要一边读byte stream一边parse每个message
4. 两个table做join有哪几种方法，分别有哪些drawback
5. merge two sorted list
6. sqrt(double number, double epsilon)
7. auto completion implementation using trie
8. edit distance
9. Implement blockingqueue
10. how is a hive query transferred to mapreduce jobs

Onsite:
1. given a list of pairs, pair.first 表示parent, pair.second表示child, reconstruct the tree, return the root node.
2. auto completion - design the service
3. design a service, accept stream of events, each event has a type and timestamp, need to support the query of top k most frequent types in a query specified [start, end] time range.
4. closest number to target in BST
5. validate soduku / solve soduku, and optimizations
6. 给一个json object，给一个wildcard path with '?' as arbitrary name，比如a.?.b 找到所有符合path的objects

Apple一般onsite的时候4轮tech interview，中午的时候将来的manager带着吃午饭。如果tech这4轮面的好会有第5轮见到hiring manager，如果有这一轮基本说明offer没啥问题了，这轮会是一堆behavior。如果第5轮也没啥问题会有第6轮见大boss，继续behavior，会问之前做过的project有多牛叉，会吹就行。

同等级下Apple的offer远不如FG给力，而且match不上去，bonus也不会写在offer letter里面，虽然据说每年的refresh有些组相当多，但是感觉整体上跟FG还是差距比较大。而且组跟组工作强度差别也很大，有些组忙死有些组闲死，不过software的组一般都还好，感觉大部分人精神状态还是不错的。

就engineer水平来看，我有遇到水平相当不错的面试官，但是整体水准远不如FG。他们各个组做项目是完全分开的，基本没交流。做东西完全是product driven，不过engineer一般需要fullstack，需要自己end to end维护一个product，这点对有些人可能还比较有吸引力。

##########
#####
##########

Amazon：
练手公司2，我面的是marketing solution和ads相关的team。大公司周期很长，感觉recruiter不是很上心。

电面：
三哥，但是感觉还行没黑。
1.用trie来解决求dictionary里面所有符合given prefix的word。然后又扩展到prefix里面有wildcard的情况，然后继续讨论如果要design a system做这个事情怎么搞，需要注意哪些问题。

Onsite：
居然没有遇到三哥，除了一轮老中外其他都是老白，每一轮开始都是至少15分钟的behavior，而且每个人还能换着花样问不一样的问题，感觉大部分脑细胞都花在这些没用的东西上面了，所以感觉很不爽。
1. OOD Restaurant Reservation System
2. Merge K Sorted List
3. K Sized Sliding Window Sum/Minimum Value
4. 给一个css file里面很多class，然后class name里面其实很多重复的，怎么compress用尽量最小size的string来表示，这样传输的byte比较少。
5. shorten url system design

6. longest palindromic substring
7. robot moving from topleft to bottomright corner of a matrix，matrix里面有些cell是障碍物不能通过，只能往下或者往右走，有多少种方法。
8. 之前做的项目，和我之前坑爹公司的architecture

相比起他们的behavior问题，我觉得亚麻的engineer水平相当一般，很多design principle都不知道，可能因为他们内部都直接用aws很多细节都不需要考虑，也有可能跟我面的组有关系，如果面的是aws会好些吧。

亚麻最后给我senior title，但是package跟其他几家比起来差距略大，所以也就没再继续谈。

##
#####
##########

WalmartLab：
我面的是walmartlab里面仅存的几个不是三哥的组，通过靠谱的朋友内推。
面试题整体难度也还好，算法基本上都是常见题目，国人面试官都非常非常非常nice。
只说其中几轮比较有意思的吧
1.topological sort
2.design web crawler system，how to scale，what would be the bottle neck and how to solve the problem
3. 如何用semaphore或者condition variable实现3个process p1, p2, p3，p2必须要p1结束才能运行，p3必须要p2结束才能运行
4. bloom filter 如何implement，estimate false rate
5. what is the best design pattern do you think and why

他们onsite有一轮会是跟product manager聊天，就是瞎扯。一个小时我都在绞尽脑汁找话题，应该是类似culture fit吧，看看你是不是比较容易融入team。

walmartlab是第一个给我比较decent offer的公司，cash给的很多，所以其实我很感激，而且我面的组的work life balance极好，我见过的最好的没有之一，onsite居然有两轮是video因为面试官WFH。平时干活也非常自由，没有OKR，没有deadline（是的你没看错，啥都没有，performance完全老板说了算）。
不去walmartlab的原因是我觉得他们实在缺有经验的engineer，而且很多做的很多东西都是实验性质的，没有明显的business impact，现阶段我还是比较想去一个大腿比较多的地方抱一下。

##
#####
##########

Sumo Logic：
一开始看到这家公司里面好多MIT毕业的人，而且听说他们bar很高，所以一开始也只是想拿来做一下benchmark。他们基本上都用scala，如果懂一点scala效果会比较好但是不懂对面试也完全没有影响。

他们的面试是先一轮phone，然后两次onsite，第一次onsite2轮，第二次onsite3轮，第一次onsite过了才会有第二次onsite。第二次onsite每一轮会有两个面试官，每个面试官都会出一道题目。

电面：
1. 两个binary tree，每个node存的值有两种可能，1或者0，把两个tree对应node做or操作。
极为简单，扯了一下immutable data structure然后聊了一会之前做的东西就过了。

onsite 1：
1. 纯聊project和讨论他们现有的data ingestion架构，刚好他们最近想用Kafka所以就这个话题聊了一个小时，最后没时间做题就结束了
2. 小三哥，但是也不黑。
given a list of intervals，query if another interval is totally covered by

the list of intervals。
totally covered是指整个区间都被某些已有的区间 cover了。
比如如果有 list of intervals = 【（1， 4），（2，8）】
given interval【3，6】就被完全cover了。
然后扩展到design a system来做这个事情，可以query，也可以insert interval，假设query操作的频率远远大于insert操作，并且interval的数量非常非常多。

onsite 2：
1. 有意思的题目1，设计Bi-directional LRU cache data structure，既可以lookup key to get value，也可以lookup value to get key，还支持set(key, value)操作，后面又加了条件，concurrent的情况下，会有什么问题，如何改进，假如set这个操作的频率远远小于get这个操作的频率，需要写代码实现。
2. robot from topleft to bottomright LC原题，无障碍和有障碍
3. given a list of sets，find all pair of sets having any intersection
4. 有意思的题目2，设计caltrain system，要实现caltrain上车下车刷卡扣钱整个功能，assume每个station都跟一个central server相连，要处理如果有network partition怎么办，eventually车费还是要charge到账户上，但是不能影响partition的station正常运作。要处理某些人下车没刷卡怎么办，followup可以非常多
5. 有意思的题目3，仍然是设计一个cocurrent环境下的time leased cache，但是有些区别，假如delete操作是一个daemon thread来做不用太多考虑，但是get(key)操作的逻辑是如果key不在cache里面，需要一个非常expensive的操作把对应value load进来，如何让这个load的操作对同一个key尽量少发生，需要写代码实现。

这家的题目我觉得非常有意思，engineer都超级nice，感觉我见过的人的能力都非常不错，年轻一点的反应非常快，年长一点的经验非常丰富。整体上看三哥并不多，虽然engineering vp是三哥。
这家很有诚意，最后给我的base跟walmartlab差不多，再加上很难估值的option。他们觉得他们的bar很高，能过他们面试的人不多，所以一旦你过了他们面试，要做好被他们的recruiter不停骚扰的准备。
有关这个公司，在其他帖子里面我提到过，虽然engineering vp是个三哥，但是感觉还比较靠谱，不像某些三哥吹牛没有边际，对于整个公司发展的前景比较有数，business model也很promising，最近刚刚拿到一笔80M的投资。

##
#####
###########

Palantir：

号称湾区面试最难的公司。但是again我运气比较好没有碰到很难的题目。我觉得这家公司有点吹的过大了，本身做的东西根本没有什么技术含量，里面都是一群没经验的stanford小年轻，都是自我感觉超好。另外去这家公司要做好准备每周工作60hours。估值150亿了还给option我也是醉了，能上市不？我的看法就是这家公司基本就是坑，从哪个角度来讲都不值得去。

他们的onsite上午会有3轮，然后中午吃完饭后会有一个小时的demo（因为实在没什么意思所以我差点睡着了），如果上午过了下午还会有1 - 2轮，一般下午会有一轮system design，另一轮是见hiring manager，如果上午没过demo结束就可以回家了。

电面：
万年不变的电面题，给一个array，问有没有duplicate
follow up1，只要index的距离 < k并且value相同就算duplicate
follow up2，只要index的距离 < k并且value的绝对值差 < d 就算duplicate
follow up3，follow up2能不能有time complexity O(n)的解？

Onsite：
1. OOD astroid game，就是飞机打石块的游戏，石块可以任意形状可以移动，飞机撞上就挂了，飞机可以发射子弹，子弹打上石块会把石块分成多个小石块按照不同方向和速度移动。要写伪代码。
2. 每个person有一个list of intervals，表示busy的时间段，问最busy的一段时间分别都是谁busy。
3. 一个描述起来不算简单的题目，但是算法不难，在版上看到过但是细节记不清了，

好像是给一堆stock profile然后算profit
4. 一个2d matrix，被分成好几个区域，区域之间都是value为0的cell，每一堆connected的非0cell算是一个区域，问和最大的区域是哪个，要设计API，怎么用json return结果。
5. system design又是 distributed key-value store，万年不变的题目，后来没啥好聊的只好跟面试官扯他们的那个atlas，distributed transaction layer，没办法想拿offer跪舔还是需要的。

基本上每个面试官都是一副老子很牛逼的样子，一问他们到底做了什么牛逼的东西马上支支吾吾说不出个所以然。他们的offer也没诚意，150k的base + 25k signon + 55000option，没谈就直接拒掉了。

##
#####
##########

Dropbox：

Dropbox的面试题都是从题库出的，但问题是他们的题库并不大。所以，我可以负责任的说，你在这个版上找到的面经题目，你在面试过程中绝对能碰到。另外他们复杂的算法题目并不多，但是大部分是跟concurrency有关的问题。

一般标配是 2轮电面 + 6轮onsite，6轮onsite中居然有两轮是behavior和culture fit

另外，他们面试的要求都是要写能run的code，要写完整的solution，不能写个主要function就完事。

电面：
1. 给一堆file，如何比较有效率的把内容完全相同的file group到一起，file可能非常大
2. 被人面过无数次的电话号码转成string，然后再word break那个题目

Onsite：
1. log_hit(), get_last_5mins_hits()那个题目，concurrent怎么搞
2. token bucket，假设每x秒提供一个token，然后外面可以申请任意数量的token，如果token不够就block，要求concurrent情况下，不能有专门的thread产生token，怎样用最简单的方法实现
3. web crawler，要分析可能的bottleneck，然后转化成concurrent运行的版本，写runnable代码。
4. system desgin那一轮是两个三哥，轮流轰炸了一个小时，把我之前做的所有东西完全推翻了，所以这一轮没结束我就知道肯定挂了。

##
#####
##########

后面这三个公司是整个面试过程中给我感觉最好的三个公司。

Uber：

Uber的效率不是一般的牛叉。我从刚开始被Uber联系到最后拿到offer基本在一个周之内搞定。面完了Uber之后真的有点心动，因为面我的人我觉得都很牛逼，人也都很超nice，非常乐于提供很多关于Uber的信息，整个氛围非常积极向上。老板虽然是个三哥但是也没有任何能吐槽的地方，他手下现在也基本都是老中。

电面：
一般电面会是hiring manager，除了问了一下之前做过什么之外只有一道题目：
OOD card deck，要现场deug，需要能运行
电面后一个小时通知我可以onsite

Onsite：
onsite一般是5轮，中间老板带着吃午饭

5轮中必然有一轮是只讨论之前做过的project，要做好准备，一定要对自己之前做的东西特别熟
另外我面试过程中问了不少怎么设计一个系统解决Uber实际问题这种题目，很新颖很有意思
1. 问了我不少关于storm的问题，比如storm怎么保证exact once/at least once semantic，如何做timed window join，因为我简历上有相关的东西，然后让我用storm来做一个比较简单的sliding window count。
2. big integer multiplication，要求现场运行代码。
3. longest increasing subarray，longest increasing subpath in a tree，path只能从root到某个leaf
4. boggle game，given a boggle board and a dictionary，find all words on the board，
follow up，如果dictionay 不变但是board不停的变怎么优化
follow up，如果board不变但是dictionary不停的变怎么优化
5. given a matrix only containing 1 or 0，find how many rectangles are 4个角都是1
6. how to design a system to automatically detect hotspot on geo graph, a hotspot is an area such that 打车的request远多于available driver的数量
7. how to design a system to detect if dispatch algorithm has some bug，dispatch主要是收集所有打车request和available driver的信息然后决定哪个driver哪个客人

Onsite过后两个小时通知我有offer了，如果onsite过后一两天之内没通知的话，基本上说明你的waiting list上，要等排在你前面的人据掉offer才可以继续下一步。

##
#####
##########

Facebook：

initial round我是直接去onsite的，但是根据其他朋友的经验似乎电面或者onsite影响也根本不大，因为第一轮基本只要没有太大的纰漏都会过。

Onsite：
一共5轮，如果是4级的话会是3轮coding，1轮behavior和1轮system design。因为偏infra, 所以我有3轮是三哥，当时已经做好挂的准备了。
1. move all 0s to right end of the array
2. decode way
3. binary tree inorder iterator
4. determine if there is a subarray sum to target number
5. convert integer to string，1000 to “one thousand”
6. system design - design facebook music system，只需要design service tier，两个API
get_top_10_list_music_ids(int64 userid) - return top 10 most frequent listened music ids for a given user last week. 这个call在load页面的时候要进行，所以对latency要求比较高。
record(int64 userid, int64 musicid, int64 timestamp) - 每当user听一首歌，就需要记录下来，这个可以asynch进行，需要eventually consistent，但不需要每听一首歌马上就能反映到上一个call中。要做各种spec和resource的estimation。
7. 抄dropbox那个问题，get_hits_last_5mins(), record_hit()，但是后面又扯到system design，如何thread safe，如果是districuted syste怎么搞，能想到几种方法
8. behavior那一轮基本上围绕着的主题是，你之前碰到什么难解决的问题，怎么解决的，你学到了什么，production有过什么比较傻叉的bug，怎么避免的。你之前做项目有没有cross team的，你怎么说服其他team听你的，等等。聊得过多导致最后没有时间所有这一轮没有coding

我觉得我的运气很好，再次没有碰到很难的题目，尤其是算法。

##
#####
##########

Google：

狗家如果真的想快的话还是可以的，我从开始被recruiter联系到offer也是一个周之内搞定。
狗家和F家整个感觉都很好，面试官都很乐意帮忙，而且明显感觉到水平跟其他公司不一样，技术功底非常扎实。

再次运气很好所以没有碰到很偏很难的题目，基本上就是水过了。其中几道比较有意思的题目：
1. 一个正整数可以表示成其他几个正整数的平方和，给任何一个正整数，求最少的那几个正整数，平方和是给定的数，比如14 = 1^2 + 2^2 + 3^2，如果给的数是14，应该返回（1，2，3）
2. 给一个dictionary，然后可以support的query是，给一个string，返回在dictionary里面包含给定string的所有character的最短的string
3. 如何设计google login system
4. web crawl的时候如何判断两个document是相同/相似的。

抱歉很多细节实在记不清了，表达能力也有限没办法在这个帖子里面说的很明白。如果大家有问题我会尽我所能回答，谢谢。。

--
※ 修改:·beanbun 於 Jun 14 02:34:35 2015 修改本文·[FROM: 50.]

发信人: beanbun (豆包), 信区: JobHunting
标 题: Re: beanbun大牛，请问你leetcode刷了几遍，到什么程度，还刷lintc
发信站: BBS 未名空间站 (Mon Jun 15 02:58:59 2015, 美东)

算法上我基本上就是两块：
leetcode和本版面经

leetcode我大概刷了3，4遍，面试的时候的状态大概是3天可以过一遍
本版面经和技术贴过去半年的我都仔细看过，也都按照公司和类型保存下来了，有用的信息非常之多，如果真的能总结好吃透这些基本拿offer不会太困难。所以我说本版对我的帮助很大。

lintcode这种下三滥的东西我不感兴趣，没用过。
真正的大牛面试是不需要刷题的，我觉得刷题这个东西需要正确对待。
我刷题是因为现在面试还是不得不刷，但是刷题不能作为自己的主要任务，刷题只能决定offer的下限，有的时候连下限都决定不了，但是绝对决定不了offer的上限。
--

发信人: beanbun (豆包), 信区: JobHunting
标 题: 准备面试篇，无干货
发信站: BBS 未名空间站 (Tue Jun 16 01:41:10 2015, 美东)

首先，无干货，可略过。

其次，我的经历不一定对所有人适用，也不是说我这么做就是对的，我工作时间也不长所以有些问题看的也肤浅，主要目的是抛砖，一不小心又码字码多了，有耐心的同学可以看看，欢迎指正和建议。

＃＃

再说一下我的背景，既然很多人感兴趣，但是再细节就没有了。。

北美cs top25水校ms不到两年
之前在一只湾区的三哥驴（非L），版上已经有人猜出来了
做的东西还算可以，大数据的infra

＃＃＃

除了刷题之外的准备。。

真正开始准备找工作是半年之前，我相信我在的驴比版上大部分公司都忙，所以开始的时候进度比较慢，最开始的时候并没有主要刷题，而是列了一些我觉得必须要了解到一定程度的system和framework来学习，我花了大概三个月时间来看一些paper，opensource project的文档，presentation，source code和engineering blog。因为工作中都在用，所以其实没有非常痛苦，但是尽量从design的角度来看问题会学到更多东西，很多时候问问自己别人为什么要这样做，再结合自己真正的经历会收获很多。这段时间也是自己对整个knowledge base查缺补漏的重要时间，只要看到不是很理解的概念基本上都要查清楚，design很多时候其实是考察你的knowledge base和基本功是不是扎实，没有knowledge base是很难做好design。

后面我还专门花时间来看跟Java concurrency有关的内容，joshua bloch的那本书(Java Concurrency in Practice)我看了一遍，然后又看了一遍Java concurrent library里面几个经典的数据结构的实现，
这个我觉得对我的帮助非常之大，很多东西以前模模糊糊突然会变得清楚很多，理解了别人是怎么实现的，其实也能学会很多时候各种常见的优化是怎么做的。甚至很多concurrency design和实现的技巧都是在这里学到的，比如之前不知道IntAccumulator，AtomicIntArray，再比如我们都知道blockingqueue简单来说怎么实现，但是Java的LinkedBlockdingQueue其实比较精巧，throughput较高，然后再跟之前接触过的disruptor queue做比较，总结下来现在无非就是从最早的compareset busy wait浪费cpu再到用wait condition节省cpu再到compareset busy wait浪费cpu但是提供更好的throughput。另外就是直接对memory进行volatile读操作可以在很多时候节省读的时候的锁。新的concurrenthashmap大量用到了这些，其实也提供了很多在做concurrent的题目的时候一些重要的优化的方法。

做完这些事情之后本来想把Kafka的源码再读一下，但是时间不够了，虽然之前design还是大体有些了解，但是我觉得hadoop，storm，kafka，Hbase，Cassandra这几个非常典型的framework是我面试中必然要非常了解的东西，之前接触Kafka并不是非常多，所以特意又把Kafka的paper和design doc研读了一下，然后又看了大量的其它公司的engineering blog来了解别人都在做什么，都有什么问题。

之前没有做过很多跟web service相关的东西，所以这个类型的东西还是要看一下，thanks god，版上有位F的大牛分享了很多有用的资料，有关最经典的几个系统的design和实现，跟F相关的网上能找到的视频我基本上都看过一遍，文档我也都看过一遍，基本上类似的问题都能用相同的原则来解决。

有些同学说design没有经验搞不定， 这个也对也不完全对，没有搞过确实缺乏第一手资料，会不知道可能会出现哪些问题，但是不代表最常见的问题你没有其它途径可以知道，大家对于各个系统的改进都是基于现有系统出现的常见问题，没做过可以，但是不能作为不会做的借口，想要了解别人是怎么做的不是一件很难的事情。现在大部分常见的系统整体来说都是大同小异，有一些最根本的原则其实大家都在遵循，然后区别往往是针对不同use case的个别的优化。

所以这里我觉得比较有用的准备方法是，在弄明白一个design之前，先要做好几个准备

1. 先把一个process或者一个系统是怎么工作的搞清楚，这里是指，design一个service需要cpu，memory，disk，network等等很多component协调工作，这些东西分别都在什么时候用到，为什么要有这些东西，分别有什么特点。
相信大家都很熟悉有一篇文章叫做The numbers eveyone should know，在没有这篇文章基础上的design都是瞎扯。

2. 要清楚这个design到底是为了解决什么问题，use case是什么，design一个系统根本上讲是为了解决一个存在的problem，这个problem会有general的要求，比如latency，比如throughput，比如load，比如哪种操作比较频繁，比如有没有consistency要求，是不是reactive，是不是需要highly available，等等等等，这样跟第1点相结合才能明白瓶颈可能在哪里，哪些东西可以tradeoff，进而才会有design的solution

3. knowledge base的储备要尽量够，操作系统，distributed system，concurrency这

些东西很难啃，我也曾经自学过几个大学的distributed system公开课，很多同学想绕过这些走捷径，但是越难的东西就越有价值。知识量不够不是问题，看一点补充一点，只要能坚持下来，到了一个时间点基本上还是可以有质变的。

所有的套路都是建立在这些东西基础之上，慢慢总结下来就会明白，在什么情况下可以怎么做来解决什么样的问题。很多时候不需要你自己去想新的solution，但是对于现有的solution能够做到灵活运用也不是一件很简单的事情。

简而言之，就是靠平时积累打好知识的基础＋多偷学别人现有的东西＋自己多总结多站在解决问题的角度来思考，而不仅仅把这些当作面试题。

另外，我觉得就准备一般面试而言，版上有两大神贴，这两大神贴里面的内容相当的赞，而且我也完完整整的读了所有的内容，这两大神贴现在还在第一页上

1. 就是beidapig大牛的总结贴(here)
2. 就是另一个facebook大牛的总结经验内推贴(here)

我对web service这一块的总结基本上就靠这两个帖子里面的内容，所以特别感谢这两位。

以上这些事情其实工作之后断断续续一直都在做，但是集中精力做大概是持续了四个月时间。
然后我觉得需要开始集中强化一下算法和coding。

＃＃＃

有关算法coding：

在之前一个帖子说过了，LC＋本版过去半年的面经。

不好意思，我不太擅长把东西整理的很有条理，所以基本上现在这些东西还是处于只有我一个人能明白是什么的状态，非常之乱所以不好意思献丑。

但是我这里想说的是，总结的结果没有那么重要，过程才是最重要的，如果你看别人面经的目的就是为了明白这几道题目或者期望面试碰到原题，我觉得面经是看不完的。这个版上的资源非常之丰富，其实都不需要完全消化就能很容易拿到offer。

看一道题目就理解一道题目而且能够跟之前类似的问题融汇起来才是目的，其实看完版上半年的面经并没有那么难，我这件事情坚持了一个多月时间，每天晚上看四，五个小时，每个帖子每道题目，所有的回帖，都仔仔细细看过，这是我感觉算法突飞猛进的一个重要时间段。

还有现在很多面试的很多题目不是偏向于算法而是coding的基本功，这个没办法就是多练习，越是麻烦不好写的题目越是要多练习，其实都有规律可以遵循。

版上的难题我一个都没有遇到，我觉得LC中等到中等偏上难度的题目应该是大部分面试的平均水平，花费大量时间在某些难题上面不一定有意义，还不如把基础打的更佳牢固些。如果你面试中被问到难题，基本说明面试官在面试前就对你不是非常认可，需要用一些比较难的题目来考察你。

我本身也面过不少candidates，所以我相信这个是大部分面试官常见的思路，其实真实的事情是很多时候面试官想让你过你就能过，不想让你过你怎么样也过不了。面试不是考试，不是题做出来就能100分，况且有很多题目都没有评分标准。

所以我说，刷题只是整个面试过程中最最基本的部分，只是必要的一个条件，远远不是全部，如果能力够了刷题刷的不好也照样拿offer，大家互相什么水平随便聊两句都知道个差不多，并不是所有人都喜欢面算法面刷题。

＃＃＃

后面就真正进入面试的阶段，然后为了心无旁骛破釜沉舟我把之前没用的假期都用了，然后开始全职专心致志搞。

面试的时候我的策略是先面练手的公司，然后中间状态最好的时候面最想去的公司，最后再冲击难一点的hot preipo，不过发现面试的时间基本上自己也说了不算，之前做好的计划基本没用。

最后三个周我基本上没有太刷题，最多就是看看之前掌握的感觉不是特别好的内容，然后随便东戳一下西戳一下看看新的帖子，每天再练习两个设计题目，如果需要保持手感就手写几道题目。每个公司onsite之前我还会把glassdoor上的面经浏览一下，主要是为了心里有数，不会紧张。
所有的onsite基本上都在最后的两个周时间，这段时间比较艰难。

有关其他：
相比刚毕业的时候，这次找工作还是有一些感触比较深的地方：

1。简历非常重要，即使是去面FG这种大公司，很多时候面试结果在真正面之前就决定了大半，如果简历还拿得出手自己对做的东西非常熟，有很大加成，所以请大家还是要刷刷简历，好好准备。我有好几次都是聊简历相关的东西一轮面试就糊弄过去了，面试官一般也不会为难。

2。找工作请找靠谱的朋友内推和找目标公司的recruiter，recruiter大部分是非常帮忙的，所以请在一开始的时候对他们好一点，他们如果觉得你有戏会尽力帮你拿到offer。recruiter在面试中起到的作用可以非常大，他们帮你安排面试官，他们可以看到你的feedback，他们甚至可以有比较好的私人关系帮你match好的组，所以，在拿到offer之前，注意是之前，装装孙子没有坏处。拿到offer之后主动权就在你自己手里了，大局已定后双方的地位会互换，negotiate offer这个环节其实就一条。。。有compete offer你就牛逼，没有你就。。。

3。面试过程也是不断学习的一个过程，这也是我为什么拼了命也要面这么多公司的一个原因，因为我想多知道一些细节别的公司是怎么做的，所以面试的时候不要担心大胆问，很多问题都是他们要解决的真实存在的问题，也是你将来可能会碰到的问题，如果面下来10个公司只是这些总结下来的东西已经可以帮你再搞定一个面试了

4。另一个我觉得很有帮助的是有一群志同道合的朋友和能够指点自己的大腿，在整个工作过程中我觉得从我的同事身上学到了很多很多东西，帮助很大，这点我不得不赞一下我之前的驴的所有中国人，可能因为都被三哥压迫所以大家特别团结，平时对于各种技术问题的交流都很到位，没有人会有所保留。现在我那一拨的人基本走的差不多了，我算走的比较晚的，一般都是越牛的人走的越早，我最终的offer在所有人里面也就是个中等水平吧。

5。运气，很重要，同样的人换一个环境可能是完全不同的结果。我之前问过一个大牛找工作最重要的是什么，曰：运气。现在我很相信这个。。。

All in all，我不是牛人，我不是国内top20毕业也不是北美top20毕业，我本科也不是学cs的，但是我特别相信版上之前一位前辈的话，大家能来到美国读一个decent学位说明大家的智商都没任何问题，很多时候结果怎样只取决于自己的决心和毅力有多强大。只要肯努力，结果就不会太差。

Posted by Bin Mu at 7:27 AM No comments:

Labels: interview, system

Thursday, June 11, 2015

发信人: Dreamer (不要问我从哪里来), 信区: Dreamer
标 题: 不刷题进Google的经历
发信站: BBS 未名空间站 (Thu Jun 11 18:34:25 2015, 美东)

没有马甲，又不想被认出，所以跑到这里发帖，希望有人能转到Jobhunting板上。

在Jobhunting板上混了很久了，看到大家的共识就是：不管你工作多久，想去FLG必须

刷题。（例外也有人提到，但是似乎不是Google research的职位，就是功成名就的大牛，都不是普通码工的情况）我自己和周围认识人的经历似乎也验证了这一点。不过最近我终于在没有刷任何题的情况下拿到了G家的offer，看起来这种“共识”也并不是100%正确的。由于Jobhunting板上这种经历似乎不多，所以详细写一下，供大家分享，也给像我一样不愿刷题的人鼓励一下。这个帖子主要侧重分享面试经历，面经记不太清了，不是太多，放在最后。

我自己四年前也曾经认真刷过0.9遍Leetcode题目，去过G家on site一次。当时自我感觉答得还不错，但是最终还是被拒。不过从那以后，每年都会被G家recruiter骚扰，说上次你表现很好，已经很接近offer了，要不要再试一次啊之类的。（不知道是不是他们对每个人都这么说？）我对自己背景相关的专业只是还是比较有信心的，但是对Leetcode类型的算法题是真心不喜欢，所以不到万不得已实在不想再刷题了（一方面年纪大了，另一方面家里小孩也忙）。所以每次G家recruiter来骚扰，我的第一个问题就是“能不能不走BT的general hire，少问算法题，多问实际有用 的东西？”但是第一次、第二次都得到否定的回答，然后就没有然后了。到了去年（第三年），那个recruiter说，我们现在改革了，不是general hire了。于是我就高兴地答应再试一次。结果转到Mountain View的recruiter后（虽然我人就在湾区，但是似乎每次主动联系我的recruiter都是Texas的，如果确定要开始面试了，就会被转到Mountain View的recruiter继续）被告知面试的问题还是一样的算法题，只不过面试官可以从跟你背景相关的组里面找。听到还是一样的面试问题后，我就直接放弃了。

今年已经是第四年了，不出所料，recruiter再次如期来骚扰。这次recruiter再次信誓旦旦地说，一定不会是general hire了，一定会主要考察背景和工作经验。不过有了去年的经验后，我对此也是将信将疑。不同的是，这次recruiter说先帮我在内部找对我背景感兴趣的组，先跟他们电话聊聊，双方多了解一下。我对电话聊聊一向都不拒绝，就把简历发给他让他去找组。说起来这个recruiter确实很尽力，找的组确实都跟我的背景兴趣很match的。先找了一个组，先后跟组里的三个人电话聊过（有manager也有比较senior的engineer）。不是电面那种问问题的，就是介绍一下他们组做什么（其实不用他们介绍我也基本都已经知道了），然后问问我的背景经历啥的。感觉三个人应该都还聊得不错，而且这种做法也确实跟我经历过的其他 公司的“正常招聘流程”差不多，所以感觉这次说不定真的是他们改革了。但是最后recruiter说，尽管他们很喜欢你，不过他们在你之前已经跟另一个candidate进行到offer阶段了，所以不能跟你继续了。不知道是真实情况还是婉拒的客套话。

不过这个recruiter并不气馁，接下来很快的时间又找来两个组，说都对我感兴趣。然后一周内安排了跟两个组的头分别电话。两个电话同样是聊天性质的，同样都感觉还聊得不错。然后recruiter就告诉我希望进行on site。由于有两个组，所以他说会从两个组里面各找几个人，最后一共5个面试官一次面完。但是，再次被转到Mountain View的recruiter安排具体面试时间时，再次被告知面试问题仍然是算法题，白板写code，一切似乎跟四年前没有两样。因为我已经四年完全没有刷过任何题了，所以当时就想放弃了。不过这次我当时正在跟另一家（非FLG）公司进行面试，而且很快就可能会拿到offer，所以想不管如何还是去G家试一把吧，大不了跟四年前一样被拒也没啥，所以就催着recruiter尽快定了面试时间。

面试当天果然如他所言，五个面试官没有一个人提到过有关我背景和现在工作的一个字，完全是上来就问问题。问题除了Leetcode类型的题目就是所谓的设计题。Leetcode的题目大家都知道不刷题的情况直接上后果如何了，设计题也都是我以前几乎完全没涉及过的多线程、网络相关的东西，所以最后可以说答得一塌糊涂。回去跟LD讲了经过，用LD的原话说就是“如果这样还能拿到offer简直没有天理了”。

回来后给两个recruiter写信抱怨，他们回信说很抱歉，但是我们想找什么都能干的人，所以问的问题还是以基础为主啥啥的（可是德州那个recruiter之前跟我保证一定会考察专业背景的啊，还说问那种算法题是ridiculous的）。不过反正已经这样了，也没啥好说的了。一周后加州的recruiter说已经拿到所有面试官的feedback，期间还要了我在G家工作的朋友的内部意见，说会送到hiring committee那里去。

再过一周还没有动静，这时我已经拿到另一家公司的offer，被催着做决定了。所以赶快联系G家问结果。结果recruiter打电话过来说今天才刚把所有材料送给HC，HC明天上午11点会开会讨论，明天下午再给我打电话通知结果。我说你不是上周就说送HC了吗？怎么今天才送？他说因为他在等hiring manager的意见。那个hiring manager去度假了，一直等到今天才拿到他的supporting opinion。他说他觉得如果没有hiring manager的意见就送到HC那里的话可能机会不大（好吧，我知道我on site答得很烂）。看起来这个加州的recruiter确实也是在尽力帮忙。

第二天，说好下午2点打电话，结果一直到快5点才终于等到电话。虽然已经做好被拒的准备，但是结果竟然被告知HC通过了！下一步就是要了三个external reference的联系方式，问了我现在收入情况和pending offer的情况，说下周一送交compensation committee讨论。第二周，本来说好周四SVP如果review通过后就会有最终结果，结果被告知SVP当天没有进行每周一次的review，一直等到周五下午快下班时，终于等到了offer。虽然整个package比网上大家报的差不少，但是毕竟大家都是有多个FLG级别的offer互相compete来的，我这种只有一个普通offer的拿到这种package也算不错了（而且后来negotiate之后还涨了一些）。毕竟我这次没有刷题，能拿到offer已经是预料之外的事了，而且这个offer还是比我现在高了不少的，所以也就愉快地接受了。

我这次的经历说明，只要背景match，就算是非Google research的普通码工职位，也是有可能不刷题拿到offer的。希望能给象我一样不愿刷题的人多一个参考的样本。

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝
以下是记得的几个题目

* Multi task design
用户可以法请求要求某一个task在某一时间开始执行。这样的请求可能很多。设计一个系统处理这样的请求。问如果处理系统是local的（和发请求的在一起）或者是remote的有哪些设计上的不同。
这个没怎么实际做过，只能随便侃侃，简单写了几行伪代码。

* Quad-tree intersection
一个quad－tree表示一个2D的黑白图，每个节点都是平行于坐标轴的矩形，节点的value 0和1表示黑和白。如果一个节点全黑或全白就是叶子，否则就继续剖分成四份。
要求写一个函数求两个quad－tree的交。
这个比较简单，写了一个递归的程序，不确定是否有bug。

* Base64 encoding
先解释了一下何谓Base64 encoding（http://en.wikipedia.org/wiki/Base64），然后要求写一个函数将一个字符串按Base64编码。
用位操作实现，写了简单的代码，不确定是否是他想要的答案。

* Swizzle sort
输入一个数组，要求输出满足：a[0]<=a[1]>=a[2]<=a[3]>=…
O(n)，一边扫描即可。发现不符合条件的只要跟前面一个数对调就可以，说明了一下原因，没时间写代码了。

* Prefix string
给定一个字典，输入一个字符串，输出字典中所有以该字符串开头的单词。
只知道可以用Trie来做，但是具体怎么做记不清了，毕竟工作中没用过。对方讲解了一下Trie的基本概念，然后假设Trie已经建好了，让写代码找出所有单词。代码匆匆写完，估计很难bug free，最多大体逻辑正确就不错了。

* Linked list operation
先简单问了一下double linked list进行insert，remove, rank（判断某一节点是第几个节点）操作的时间复杂度。答分别是O(1), O(1)和O(n)。

然后说如果允许纪录每个节点的位置（也就是rank值），时间复杂度分别是多少？答分别是O(n), O(n), O(1)。
然后问能否有什么方法平衡一下，让三者复杂度差不多？根据提示构造一个二叉树，同时每个节点纪录自己左子树中节点的数目，从而使三者的复杂度都为O(log n)。
最后要求写出rank的代码。基本都写出来了，但是期间无数提示。

* DNS design
描述一下如何设计DNS系统，以及如果某一网址的IP更该了，如何更新各DNS。
没接触过这方面的东西，所以只能随便瞎聊。

* Maze design
假设你是大学里的算法TA，老师出一个走迷宫的题目，要求你
1. 设计一个函数头，让学生补充内容；
2. 设计一个maze generator用来检查学生提交的程序。
1写了一个很简单的，不知道还有什么值得补充的。2实在不知道他想要啥，沟通半天也没弄明白，只能随便敷衍两句。

* 3 sum变种
输入一个数组和一个数x，要求输出满足a+b+c<=x的triplet (a, b, c)的个数。
先写了一个naive的三重循环O(n^3)的，要求改进。写了一个先排序再两重循环O(n^2log n)的。要求将用到的binary search写出code来。然后问如果查找的数(x-a-b)有重复或者不在数组种怎么办，根据两种情况修改code，不确定最后是否完全正确。
然后问能否进一步改进，一直弄到时间结束也没弄出来。最后被告知可以不用binary search达到O(n^2)，也没明白怎么做。
--

Posted by Bin Mu at 4:40 PM No comments:

Newer Posts Home Older Posts

Subscribe to: Posts (Atom)