Timeline:

8:00am: Travel around Amazon campus

9:00am: Go to meeting room (every table has three computers, each computer has windows and a virtual box version linux ubuntu), and three interviews/proctors will some presentations

9:30am-9: 45am: distribute questions, group discussion

9:45am – 10:00am: Read question, interviewer will come to ask initial design and task allocation, Coding time, mixed with group discussion

12:00pm: 1-on-1 interviews about the solution idea, future problems, data structure

12:30pm – 4:30pm Coding, plus one 15 minutes 1 on 1 discussion at around 3pm-4pm

4:30: Deadline and submit code, fill a survey

4:30 – 6:00: Communication with three interviewers, panel talk

Time for 30mins and 15mins discussion are not fixed, it depends on the real situation.

Tips and Important points:

1. Communicate with the three proctors, they will decide the offer.
2. **1-on-1 talking with interviewer is extremely important, you need to show you really know the question, you know how to solve it, risk/trade-off, efficiency, you know what other people are roughly doing.**
3. When interviewer ask group member questions, don't just keep coding, pretend you are listening as well and take some notes.
4. Start with simple but practical solution and then optimize.
5. Keep communicating
6. **Data Structure is extremely important, make sure you know why you choose this data structure and its advantages and disadvantages, and is it extensible.**
7. Make sure you have readme.txt to describe the structures.
8. Make the code and structure clear, add **comments**.
9. 算法题都没派上用场，同时自己工程经验不太丰富(Means should not focus too much on algorithm, data structure is more important)
10. **Again, Data structure is extremely important.**

11.Be careful about coding style, module, comments, unit test (even though it is empty, just let them know you are aware of that, add comments to unit test)

13. 代码要规范，命名，注释都要注意，群面过程要体现 cooperation and leadership. 要主动组织和引导小组讨论

14.大概能运行后，拼命写注释和 readme,最后把输出结果的 file,readme,code 分版本 全部提交

15.对于题目要迅速解题，用暴力破解让 code 跑出结果，快速优化，然后把思路和面试官讲清楚。在过程中要经常关注自己组员的进展，体现 leadership.

16. 没有多线程，文件读写他都给你写好了，你只需要在他给的一个函数里填代码就行

17. 几乎用不到 design pattern，因为框架都已经给你写好了，我感觉基本做的就是往他给的函数里写算法就行，

18.各种 collection, hashmap, linkedlist, iterator 之类的要熟 .代码写规范.考虑各种 corner case

19. 基本比较好的流程是第一次讨论前整理出思路，设计好需要的数据结构和算法，第二次讨论前完成所有代码编写，之后最后一个小时补充注释、修饰代码。

20. communication 和 leadership 非常非常重要。proctor 虽然坐在屋子中间，但他们在时刻注意大家的讨论情况，并且在默默记录。其次别闷头狂写，主动问候别人 how is going 甚至主动帮助别人解决问题都是你的 bonus。

21.

一定要跟 proctor 解释清楚自己的思路，用图，说慢点。重点说你用了什么数据结构，为什么用那个，有什么 risk，有什么好处，如果用 java，一定要熟练 Collection 里的数据结构，尤其两种 List，HashMap，PriorityQueue，知道他们各种操作的时间复杂度和优缺点；你用的算法是什么，复杂度是什么，还能不能更优化。这里的 bonus 是你知道这个 task 的多个解法，比如 greedy 和 DP，虽然 DP 可以得到 optimal 的 result，但是 greedy 算法的时间复杂度更低，可读性更好。要是能做到这点，offer 基本就到手了

22.

代码方面，proctor 告诉我他们其实之后基本不看代码，因为只要知道了你的算法，然后看到你的代码能 work，基本代码这块就够了。但是代码里注释要完全，代码一定要写得干净。

23.

熟悉下数据结构（特别是 Hashtable, Queue, ArrayList 这些常用的），问问自己什么时候该用哪种数据结构，好处和坏处，时间复杂度，底层的实现方式等等。

24.

上 topcoder 或者其他网站找一些群面类似的题目（真正题目很长，签了 NDA 也不方便说，但大致都是一些 shipping, schedueling 类的题目，就是 Amazon 会实际面临的问题的简化版），自己尝试着做一下，再看看别人是怎么做的。在做的时候一定要问自己这几个问题（我为什么要用这个数据结构/算法，有没有别的 alternatives 可以用，如果有，对比各种不同数据结构/算法的优劣。当前 solution

下，如果数据量很大怎么办？如果给自己很长时间（比如 3 个月半年），如何优化？在现实中还会遇到什么问题，现在的 solution 考虑到了吗，如果没有，该怎么 improve）。如果英语不是太好的话，在练习的时候尝试把自己的解法以及这些扩展问题都说出来，因为这是到时候你跟 proctor interview 的时候会讨论到的。

25.

读题目的时候要把所有的部分都读了，而且要思考下队友的部分，因为有可能你负责的部分跟他们的是有一定联系的，而且如何帮助队友也是群面考察的一个重要指标。

26. 看了题后，先想一个最简单的思路（比如暴力解），你要保证你提出的解法能在 2 个小时内 code 完并且跑通，千万不要想了一个高大上的解法，最后没能实现。什么 machine learning，prediction model 之类的都不要去想，根本不可能完成，也不是群面考察的重点。

27.

读完题小组讨论的时候不仅要对你自己的部分提出思路，也要对你队友的部分提出思路，帮助他们分析。proctor 一直都在观察每一个 candidate，你跟队友的沟通交流很重要。

28.

在以小组形式跟 proctor 讨论的时候，一定要把你的思路理清楚，告诉 proctor 自己的部分大概是要做什么，你用了哪些 assumption，需要用哪些数据结构，算法是什么样的，如果可以的话，大致用 pseudocode 写出来。这个时候 proctor 会给你提出来一些建议，一定要认真听，不明白就问。

29.

接下来写 code 的时候注意模块化，输入输出都已经给你写好了，适当加上一些 comment，写一部分就测试一部分，尽量在第一次和 proctor interview 的时候你有一个能跑的版本。写 code 的时候也要时不时的问问你的队友做的怎么样了，有没有什么需要帮助的。

30.

第一次和 proctor interview 的时候把自己的算法说清楚，并把之前说的那几个点也跟 proctor 讨论, 上 topcoder 或者其他网站找一些群面类似的题目（真正题目很长，签了 NDA 也不方便说，但大致都是一些 shipping, schedueling 类的题目，就是 Amazon 会实际面临的问题的简化版），自己尝试着做一下，再看看别人是怎么做的。在做的时候一定要问自己这几个问题（我为什么要用这个数据结构/算法，有没有别的 alternatives 可以用，如果有，对比各种不同数据结构/算法的优劣。当前 solution 下，如果数据量很大怎么办？如果给自己很长时间（比如 3 个月半年），如何优化？在现实中还会遇到什么问题，现在的 solution 考虑到了吗，如果没有，该怎么 improve）。如果英语不是太好的话，在练习的时候尝试把自己的解法以及这些扩展问题都说出来，因为这是到时候你跟 proctor interview 的时候会讨论到的。

31.

接下来有时间的话可以做一些 improvement。 improvement 可以是更 efficient 的算法，也可以是对题目更深层次的理解。因为题目都比较偏向 open question，不同的 assumption 有不同的解法。这时候你可以 make more realistic assumptions,

然后 develop 新的 solution 作为 improvement。当然这一切的前提都是你有一个能跑的版本。

32.

善用资源，能内推就不要网投。如果内推没有回音，我有一个小 trick，去 linkedin 上搜这个公司的 university recruiter，然后猜他/她的邮箱地址发邮件要面试（怎么通过名字和公司猜邮箱，请 google，大多数情况下是 firstname 的第一个字母加 lastname@公司)。或者可以找身边的同学问 recruiter 的邮箱，因为大多数 recruiter 都会在邮件中写希望把邮箱 share 给其他有兴趣的同学。这样直接要面试基本都能成功。

33.

看代码的时候，一边解释一边画图, optimization 最有意思的地方在于你有很多个 optimize 的标准。举个例子，要安排一个货物发货，可以按成本最低来 optimize，也就是尽量让货车装满；也可以按照时间最短来 optimize，也就是尽量让货物尽早发出去。

34.

熟悉下数据结构（特别是 Hashtable, Queue, ArrayList 这些常用的），问问自己什么时候该用哪种数据结构，好处和坏处，时间复杂度，底层的实现方式等等。

35.

第二次 proctor interview 的时候可以重点讲讲 improvement,就算你没有实现，也可以讲讲你的思路

36.

最后就是记得写好 comment/javadoc，写一个 readme 把你的思路和 improvement 的想法解释一下。

Coding Logistics :

1. Input data, testing data, output data is provided,
2. ThinkPad t400(windows keyboard, 14inch), Windows + VirtualBox Ubuntu,
3. Eclipse in both of Windows and Ubuntu.
4. You can google
5. choose from Java, C++, Python
6. Lunch is in a table, you can pick by yourselves, not good.

Topics in discussion:

1. Progress
2. What solution and method?
3. What data structure did you use, why? and way to optimize?
4. Complexity

Questions:

Very long questions, around 10 pages of A4. The question is divided into three parts and you need to choose and work on one of them. Some questions maybe related with machine learning, big amount of data processing

Some examples below:
1. 项目是一个库存管理和物流系统，因为是NDA，所以LZ不会告诉大家原题的请放心
假设京东商城有一个库存管理和物流系统，它在我国东部，西部，中部各有一个仓库，
假如西部有一个客户在商城上订了某件商品，要求商品N天内到达，那么京东就会从最
省钱又同时能满足需求的仓库为客户调货，举个例子：
客户订了商品A，要求货物发往中部，3天内到达
京东商城自己的物流系统如下：
中部到中部：1天xx元，2天xx元，3天。。。N天xx元
中部到西部：1天xx元，2天xx元，3天。。。N天xx元
。。。。。
（不一定同一区域发往同一区域是最省钱的方法，例如中部到中部的1天速递为10元，
但可能西部到中部的1天速递只需要9元）
然后现在有京东商城客户1-4月的订单记录（含有时间，发货目的地，数量，要求送达
时间等信息），要求推算出京东5月时各个区域的仓库中各种商品的进货量，条件是要
最小的运送花费，最少的无效订单（没货了，或者现有情况货物在规定时间内到不了）
以及最少的库存量

2.Prediction + Optimization

3.广告投放

4.网页布局

5. Inventory, Shipping, Lock, 大批量预测
问题很实际，我做的是 inventory, input 有几万行数据，关于产品销售信息，具体字段是什么不便透露但是你可以自己想想 产品能有什么字段，还有一张表示 order 表，以及运送费用的 cost 表。 要做的事儿就是预测某个地区某月要存某件商品多少件。 要保证运费低，又要保证商品不会卖空。只能说到这。 另外自己脑补。

6. Amazon 主要业务就是物流送货，所以做的 project 肯定是跟物流有关系

7. 多目标带权值的优化问题 (multi-objective optimization)

8. 题目是广告系统，就是阿妈总想要在网页上显示广告，网页被分成8个部分，不同的部分有不同的权值乘数（multiplier），从1到0.3。每一条广告都有自己的权

值，每个广告可以在特定的时间出现在几个可选的位置。这个project被分成了三部分：scheduling，selection，optimization，每个人选一个部分做。
LZ做的是selection，主要说这部分。有这么几个输入文件：广告的ID和权值；网页每部分的权值乘数；某一个时段需要出现广告的位置；某一时段广告可以出现的可选位置。目的就是要某一时刻网页上的权值与权值乘数的积的和最大。

9.
还是没变，就是那个schedule selection optimization
面试过程中，考官再三强调，你可以写一个简单的甚至sb的实现方法，但必须能跑通。
schedule：挺难的。记得有帖子推荐做这个，其实读完题目后我感觉还是有难度的.
selection：最简单的。说它简单是读完题目后我很快就想到一个solution，而且效率还是很不错的
Optimization（我选了这个）：最难的。但当时不知咋的，就想到了solution了，所以选了这个。这题要用hashmap

Task to do:

1. Heuristic algorithm, used to describe and solve the intuitive version(Done).
2. Weighted interval scheduling(Done)
3. Linear regression (Done), multivariate linear regression (Done)
4. Ad selection,scheduling(Done)
5. Constrained and unconstrained optimization
6. Quick class object default comparison(Done)
7. Implement Monte Carlo(Done), local greedy search(Done)
8. Gradient descent/derivative
9. inventory prediction(done)
10. MVC pattern, factory pattern, adapter pattern, singleton pattern(all done)
11. How to convert dfs to bfs and vice versa(done, swap queue and stack)
12. Backup all implementation into ut pub_html

My pipeline:

Understand the question, ask for correctness of assumption.

During talking, UML diagram to explain design pattern, cost analysis, alternative solution. Remember to mention academic interest(research area).

Answer format: What algorithm you use? Motivation(Advantage)? What Data Dtructure? Running time? Potential risk(when data grows huge..) and what you do when given more time? Alternative solution? Academic interests?

Questions to ask:

1. I think this question is a simplified version of a real world problem faced in Amazon. So I want to know more about the real problem so that I can keep my data structures and framework more flexible, extensible and robust?
2. How do you feel about working in Amazon and Seattle?
3. I heard that Amazon start food delivery in Seattle recently,
4. I think the first leadership principle in Amazon is customer obsession, so I was wondering what if

Topics during discussion:
1.Motivation (why good) and **Assumption (data size is around the input sample, if data grows huge, we will need other strategy, we will discuss that later)**
2.Algorithm Model (UML for class design, graph to represent idea, cost analysis, **pseudo-code** for combinatorial search and Dynamic programming)
3.Data structure (why? Advantage for data structure, extensible?)
4.Running time (based on current data structure)
5.Risk (**when data grows huge**? 1. Use greedy or another methods that has less time complexity, 2. Distributed computing, map reduce), **improvement** (second method)
6.Alternative solution (Given more time, or under different assumption)
Personal Research Interest

Physical order of doing things:

1. Quickly skim question, if meet before, make sure you get the question you want, if not meet before, trying to understand the question well and decide.
2. Discuss each part with group member, make sure to use the white board, and write task allocation on it.
3. Start coding, download MISC folder (put it in a hidden place), create unit test file (change all methods to pass first), readme file (current implementation and future optimization), check whether it is python2 (If not, download python2 and change the base code a little bit)
4. Generate and read pydoc and code carefully and make sure write docstring for module, class and functions description while coding.
5. Write UML class diagram (while coding, MVC) and put design pattern names aside.
6. Build a small testcase and test while coding.
7. Ask group member how is it going occasionally.
8. Get a working solution as soon as possible, generate pydoc
9. Get second solution, then compare running time, cost, mention strategy pattern to use different algorithms under different situations
10. Add unit tests
11. Comments