Debug 部分

1. 熟悉排序算法，BST，BFS，DFS，MAP，List 时间空间复杂度
2. 熟悉big O，递归迭代时间空间复杂度
3. 主动引导，总结

Coding 部分

1. 认真看已有框架
2. 选择数据结构合理 (对比不同数据结构)
3. 先暴力，在优化
4. clean code （java doc注释，骆驼命名，static final，初始化放进constructor，用interface, 要注释时间复杂度
5. 第一轮面谈：思路，进度和优化
6. 第二轮面试：总结遇到的问题和解决
7. 检查corner case，加上unit test

Coding 大致思路

1. Problem 1: get all possible shipping solutions:
2. [input (product_id, toCity); output(warehouse, List<Cost>)]

```
        (1) //declare 2 map for quick find
        public static Map<Long, List<Warehouse>> productLookUpTable; // Map<productId, Map<warehouseId, Warehouse>>
        public static Map<Location, Map<Location, List<Shipping>>> shippingLookUpTable; // Map<toLocation, Map<fromLocation, Shipping>>
    (2) //build two map
        mapProductIdToWareHouse(warehouseList);//Map productId -> warehouse
        mapTwoCityToShipping(shippingCostList);//Map <toCity, fromCity> to Shipping
    (3) //get output
        getShippingSolution()//find all warehouse with product_id using productLookUpTable, then find all shipping cost using shippingLookUpTable
```

2. Problem 2.1: reach maximum order delivered
   [input (List<Order>); output(List<Warehouse>)]

```
        (1)   //classify order with product_id, store with map
        Map<Long, PriorityQueue<Order>> productOrdersMap = new HashMap<Long, PriorityQueue<Order>>();//map productId and order
    (2) //build the product-order map, sort order with (1): increasing amount, (2): increasing shipping routes
        while (orderIterator.hasNext()) {…}
    (3) //get output
        for (Long productId: productOrdersMap.keySet()) {//for each kind of product, dispatch order in heap
                while (!productOrdersMap.get(productId).isEmpty() {//set dispatch location for this order
                        order = pq.poll()
                        for (all possible dispatch warehouse for this.order) {
                                update(warehouse)
                                if (satisfied) {break;}
                                else {put rest of amount order back to pq}
                        }
                }
```

```
}
```

2. Problem 2.2: reach maximum order delivered on time

   [input (List<Order>); output(List<Warehouse>)]

```
    (1).  //classify order with product_id, store with map
    Map<Long, PriorityQueue<Order>> productOrdersMap = new HashMap<Long, PriorityQueue<Order>>();//map productId and order
  (2) //build the product-order map, ignore too-late order, sort order with (1): increasing amount, (2): increasing shipping routes(also include time)
  (3) //get output, same
```

3. Problem 3: optimize cost based on Problem2

   [input (List<Order>); output(min_cost)]

```
    (1)   //first get non-optimized cost, easy
    getNonOptimalCost(locationList, orderList)
    (2)   //then get optimized cost
    getOptimizedCost(locationList, orderList)
      //(1) remain dispatch location choose least expansive shippment
      //(2) if a order has more than one potential departure point, if any other point has remain stock, replace shipment if cheaper
      //(3) affect other order, need flow algorithm
```

4. Tips:

   (1).Override hashCode(), equals() and toString() to ‘new Class’, and add new Test Class to it.

   (2).Don’t modified built data structure, use index++ and get() rather than remove()

   (3).Mark down key word on white board before interview

4. Self introduction:

Hello, I am Howie Wang, I am a graduate student from University of Texas at Dallas, majored in Computer Science . My concentration is objected oriented design, and data science. I am proficient in Java programing, that I use it in most of my projects. Also I am familiar with web techniques, as you can see from my resume, including html5, css3, javascript, bootstrap, and back-end like php, SQL, and Unix. Well, I am glad to to take this Amazon’s group assessment. Hope I can do my best today. Thank you.