第二轮：给定一个中国棋盘棋盘（size is m*m），一个source位置， 和一个target位置，和一些blocks. 问题是从source开始，放一个“马”，于是只能走“日”（比如从（0，0）能走到（1，2）等8个位置），问这个马能不能从source走到target. Simple BFS.
第三轮：给一个string=abc3[b2[c]],要求输出其expansion,比如： abcbccbccbcc.
Simple recursion搞定
第四轮：system design,设计个订票系统

1. Implement find command in linux
find all file >5mb
find all xml
Assume file class
{
get name()
directorylistfile()
getFile()
create a library flexible that is flexible
Design clases,interfaces.

```
class File {
    String name;
    int size;
    int type;
    boolean isDirectory;
    File[] children;
}

abstract class Filter {
    abstract boolean apply(File file);
}

class MinSizeFilter extends Filter {

    int minSize;

    public MinSizeFilter(int minSize) {
        this.minSize = minSize;
    }

    @Override
    boolean apply(File file) {
        return file.size > minSize;
    }
}

class TypeFilter extends Filter {

    int type;

    public TypeFilter(int type) {
        this.type = type;
```

```
    }

    @Override
    boolean apply(File file) {
        return file.type == type;
    }
}

class FindCommand {

    public List<File> findWithFilters(File directory, List<Filter> filters) {
        if (!directory.isDirectory) {
            return new NotADirectoryException();
        }
        List<File> output = new ArrayList<>();
        findWithFilters(directory, filters, output);
        return output;
    }

    private void findWithFilters(File directory, List<Filter> filters, List<File> output) {
        if (directory.children == null) {
            return;
        }
        for (File file : directory.children) {
            if (file.isDirectory) {
                findWithFilters(file, filters, output);
            } else {
                boolean selectFile = true;
                for (Filter filter : filters) {
                    if (!filter.apply(file)) {
                        selectFile = false;
                    }
                }
                if (selectFile) {
                    output.add(file);
                }
            }
        }
    }
}
```

2. calculate string '5+2-3'. follow up ==> the formula will be with */ (parse the string and calculate it)
3. Design Compass system
4. Behavior Q

1. system design 国人，设计一个类似compass，redfin，zillow这样的地产网站，主要讲了里面的search功能，还讲了一些，怎么吧map置入到搜素结果页面的实现
2. Coding, 本质就是一个sort的题目，取前2个，可以全sort，也可以priorityqueue, comparator也是特殊的，根据面试官的题目背景来写的，总体不难，讲了很多software refactor上的面的优化，如果PM改要求了，怎么半啊，主要就是把代码抽出来做成一个个子方法，以适应需求的变更

3. Manager轮，主要面了简历上的问题及tech引申，有点像system design，这边答的不大好，最后问了BQ。这是我面过的最wired的面试官，声音巨小加轻飘飘，感觉抽了大麻才能回到正常水平的那种小，表情诡异。
4 Coding, 一个图的问题，背景是google map，给定API，求两点之间最快时间的路线，有三种不同的方式，走路，地铁和公交。代码量还是挺大的，本质上就是一个dijkstra，没有写完。小姐姐有点aggressive，喜欢打断我的讲话。感觉面的也不算好

两轮coding，一道是template render，根据一个hashmap把template里面相应的word替换；
另一轮是一道谷歌面经，map里面两点坐标，中间有障碍，求最短距离，BFS；如果map无限大，该怎么办。 lc 1036
一轮design，youtube play and recommend。
一轮manager bq，以及project dive deep，全程中文。

第一轮，nyc那边remote面试，codepad上面写的。input是string per page for a book和target word，然后返回word在所有page里面分别的序号。unit test已经提供了，需要handle一些特殊的字符。
第二轮，word ladder
第三轮，design a health monitoring service for a pipeline(multiple dependent services)
第四轮，聊聊项目然后就问了个tiny url。

设计数据结构：假设一个Cloud，有三个method，分别是：添加Server、删除Server和随机返回一个Server，要求每个method的时间复杂度是O(1) at worst case。

```
interface Clound {
   void addServer(Server s)
   void removeServer(Server s)
   Server randomSelect()
}
```

Onsite 3. 李口依伞酒，follow up是李口伊思林，但是只需要返回一个结果即可。

Onsite 4. system design，设计一个简单的Search功能：用户输入Address，返回周边的房屋信息，涉及到了前端的User Experience和后端的Performance等细节。不需要在白板上写，口述+讨论即可。

2轮算法，lc伞铃铃/尔耳，前面那道一开始问伞灵丝我说电面也是这道。。尔耳那道需要优化performance，虽然没写完不过国人小哥还是挺满意的（感谢
1轮vp纯behavior，1轮system design（把网上比较常见的那些system design准备好，考的是照片/视频上传下载管理），1轮resume deep dive + 根据你的project问follow up，烙印的director问的还是挺有水平的

1. 代码：离口幺儿柳/幺儿起

2. 代码: 给一片文章，要求统计出每一个词出现的页数。重点在于考察如何断字，以及处理这个文章里的特殊字符，比方说： “, . ' - _”... 本题跟算法没有太大的关系，直接拉低了整个面试的格调。
3. 行为问题 + 项目介绍
4. 行为问题 + 设计一个查询相关的service。

给一系列比赛结果，例如“a beats b”, “b beat c”，“c beat d”；输出最终排名，这个例子是abcd，算是alien dictionary的变形。
不存在circle，例如“d beats a”。然后问了time and space complexity，还有会用什么样的test case去测试。

```
public class GetGameRank {

    public static void main(String[] args) {
        // write your code here
        String[] test1 = new String[]{"a beat b", "b beat c", "c beat e"};
        getOrder(test1).forEach(x -> System.out.print(x + " ")); //a b c e
        System.out.println();

        String[] test2 = new String[]{"a beat b", "a beat c"};
        getOrder(test2).forEach(x -> System.out.print(x + " ")); //a b c OR a c b
        System.out.println();

        System.out.println(getOrder(null).size() == 0);


        String[] test3 = new String[]{"a beat b", "b beat c", "c beat e", "e beat f", "e beat d", "d
beat k"}; // a b c e d k f
        getOrder(test3).forEach(x -> System.out.print(x + " ")); //a b c e
        System.out.println();
    }

    public static List<String> getOrder(String[] games) {
        LinkedList<String> res = new LinkedList<>();
        if (games == null || games.length == 0) return res;
        Map<String, Integer> inDegree = new HashMap<>();
        Map<String, List<String>> graph = new HashMap<>();
        for (String game : games) {
            String[] team = game.split(" beat ");
            inDegree.put(team[1], inDegree.getOrDefault(team[1], 0));
            inDegree.put(team[0], inDegree.getOrDefault(team[0], 0) + 1);
            graph.computeIfAbsent(team[1], x -> new ArrayList<>()).add(team[0]);
        }
        Queue<String> queue = new LinkedList<>();
        Iterator<Map.Entry<String, Integer>> iterator = inDegree.entrySet().iterator();
        while (iterator.hasNext()) {
            Map.Entry<String, Integer> entry = iterator.next();
             if (entry.getValue() == 0) {
                queue.offer(entry.getKey());
                iterator.remove();
```

```
            }
        }

        while (!queue.isEmpty()) {
            String curr = queue.poll();
            List<String> next = graph.get(curr);
            res.addFirst(curr);
            if (next == null || next.size() == 0) continue;
            for (String n : next) {
                inDegree.put(n, inDegree.get(n) - 1);
                if (inDegree.get(n) == 0) {
                    inDegree.remove(n);
                    queue.offer(n);
                }
            }
        }
        return res;
    }
}
```