我搞了一个Google doc 里面已经有很多一亩下载的资料。 几个pdf 和 一些人的code.
https://docs.google.com/document/d/1fWjrlqOARLB6HzkBGuD3f5HfsjNaxmbULkz8oUuF0Kg/edit?usp=sharing

Wechat: sy9804

(1).频率统计
https://docs.google.com/document/d/1G189b8GWsvst4wEUXJP8EfjV8CECuZuRrxMYvRzNCBs/edit

(2). Github
https://github.com/allaboutjst/airbnb

(3).
地友在今年9月整理的面经，传送门：
https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=444503

(4). 深秋版
https://drive.google.com/file/d/1bJ6RIeeQPmQ22aKRAsS-BIvv8cRqFTCv/view

(5). **[面试经验] Airbnb的新题 - Board Score**
https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=462216&extra=page%3D1%26filter%3Dsortid%26sortid%3D311%26sortid%3D311

(6).
Slack: https://tinyurl.com/offertemple 找#airbnb

(7).https://www.1point3acres.com/bbs/thread-191081-1-1.html
airbnb 面经 phone interview & onsite 附录题库呦

(8). airbnb面试题汇总/
https://yezizp2012.github.io/2017/06/01/airbnb%E9%9D%A2%E8%AF%95%E9%A2%98%E6%B1%87%E6%80%BB/

(9). Python from 周同学

https://gist.github.com/yuzhoujr/83bb4bd5105fb32faa03f9bc8d8e99b4

(10).
https://www.1point3acres.com/bbs/forum.php?mod=viewthread&tid=453123&page=1&authorid=229357
低频滑雪题（https://goo.gl/eopNXw): 直接bfs不够高效，会有重复subpath，可以按照topo顺序来遍历

(Question1):17.5% 倒水
(Question2): 14.3% cheapest flights
(Question 3): 13.1% file system
(Question 4): 9.4% alien dictionary
(Question 5n): 5% sliding puzzle
(Question 6): 5% 2D list iterator
(Question 7): 3.75% combination sum/prices sum to k
(Question 8): 3.75% find the median in large file
(Question 9): 3.1% Palindrome Pairs
(Question 10): 3.1% Boggle game
(Question 11u): 3.1% IP to CIDR
(Question 12): 2.5% guess number
(Question 13): 2.5% queue using fixed size array
(Question 14): 1.9% wiggle sort
(Question 15): 1.3% meeting interval
(Question 16): 1.3% display page
(Question 17): 1.3% 给一个[["A", "B", "C"],["A", "C", "D"]
(Question 18): 1.3% find buddy
(Question 19): 1.3% csv parsing
(Question 20): 1.3 % bank system
(Question 21): 1.3% Minimum Vertices to Traverse Directed Graph

(Question1)::

17.5% 倒水 (follow up: 1. 打印倒水前后状态 2. 没有高墙，注意可能两边墙不一样高的时候自己跑一下，看看合不合理)

思路：
Link: https://github.com/allaboutjst/airbnb#13-water-dropwater-landpour-water
Sample code:
https://github.com/allaboutjst/airbnb/blob/master/src/main/java/water_land/WaterLand.java

“第一轮 pour water。两边没有无限高的墙 + 打印倒水前后的形状，这一轮有一个test case没过, 就是两边墙不一样高的情况，比如说[2,1,0,3,2,6],滴水数量100，位置任意。我当时的算法对这个case是不work的，建议大家准备的时候最好多试一些case”

Solution:

Leetcode755. Pour Water
http://tpcg.io/KN43f5

(Question2): 14.3% 所有有向图找最大最小问题(cheapest flights/10 wizards/滑雪/可能的任何变种) (follow up: 找出路径)

数据点：据说要写出dijkstra

Solution:
Leetcode787. Cheapest Flights Within K Stops

Airbnb_Q2_CheapFlights: (https://www.tutorialspoint.com)

http://tpcg.io/v1JEEZ

```
import java.util.*;

public class Solution{

	public static void main(String []args){

	// example1:
	Solution sl = new Solution();
	int n = 3;
	int[][] edges = {{0,1,100},{1,2,100},{0,2,500}};
	int src = 0;
	int dst = 2;
	int k = 1;
	int res = sl.findCheapestPrice(n, edges, src, dst, k);
	System.out.println("Example1: " + res + "\n"); // Expect: 200
	
	// Example2:
	n = 3;
	int[][] edges2 = {{0,1,100},{1,2,100},{0,2,500}};
	src = 0;
	dst = 2;
	k = 0;
	res = sl.findCheapestPrice(n, edges2, src, dst, k);
	System.out.println("Example2: " + res + "\n"); // Expect: 500



	}
	
	public int findCheapestPrice(int n, int[][] flights, int src, int dst, int K) {
	Map<Integer, List<Neighbor>> graph = new HashMap<>(); //<node, {neighbor,
dist}>
	for (int[] flight : flights) {
	if (!graph.containsKey(flight[0])) {
		graph.put(flight[0], new ArrayList<>());
	}
```

```
        graph.get(flight[0]).add(new Neighbor(flight[1], flight[2]));
        }
        Map<String, Integer> mapCost = new HashMap<>();
        PriorityQueue<Entry> pq = new PriorityQueue<>((a, b) -> a.cost - b.cost);
        pq.offer(new Entry(src, 0, 0)); //<node, k, cost>
        while (!pq.isEmpty()) {
        Entry cur = pq.poll();
        int node = cur.node, k = cur.k, cost = cur.cost;
        if (k > K + 1 || cost > mapCost.getOrDefault(k + "#" + node,
Integer.MAX_VALUE)) continue;
        if (node == dst) return cost;
        mapCost.put(k + "#" + node, cost);
        if (graph.containsKey(node)) {
                for (Neighbor next : graph.get(node)) {

                pq.offer(new Entry(next.node, k + 1, cost + next.dist));
                }
        }
        }

        return -1;


        }

        private class Neighbor {
        int node;
        int dist;
        Neighbor(int node, int dist) {
        this.node = node;
        this.dist = dist;
        }
        }

        private class Entry {
        int node;
        int k;
        int cost;
        Entry (int n, int k, int c) {
        node = n;
        this.k = k;
        cost = c;
        }
        }
}
```

(Question 3): 13.1% file system(follow up: 1. watch function)

“coding轮：file system，当时拿到题目的时候，内心是窃喜的，然而.... watch函数的参数是（String path, Runnable runnable)，和我之前准备的并不太一样，结果就改呀改，我说我不熟悉Runnable的用法，面试官小哥就让当场google，最后也没搞成他想要的样子”

(Question 4): 9.4%   alien dictionary (follow up 1. how to solve with DFS 2. 如何找出所有可能方案)

(Question 5): 5%　　sliding puzzle (注意需要自己定义board, move　follow up: 找出路径，自己表示，比如UP，DOWN....)

(Question 6): 5%	2D list iterator (follow up: 1. 如果不知道层数怎么处理)

(Question 7): 3.75% combination sum/prices sum to k (1. 先确定每个菜可以点几次以及是否有重复 2. input 不是double[] prices，而是菜单，要求找出所有菜名的组合)

(Question 8): 3.75% find the median in large file

(Question 9): 3.1%   Palindrome Pairs

真题要求与leetcode不同, given a list of strings, find all palindromic pairs of string and output the **concatenated palindrome string** in order.

Eg. [abc, bca], output: abccba, cbaabc

[aabc, cb], output: cbaabc

Solution:

1. Put all the reversed strings in input into a Map. The key is the reversed string, value is the indices of the string
2. For each string, get all its prefixes, if the map contains the prefix && the rest of the string is a palindrome, then we can find a pair where the first part is the current string, and the second part is the reversed prefix.
3. For each string, get all its postfixes. If the map contains the postfix && the first part of the string is a palindrome, then we can find a pair where the reversed order of the postfix is the first part, and the string is the second part of the pair.

Notice:

We need to keep track of the indices of each string to avoid corner case like "a", the string itself is a palindrome.

Sample Code:

```
import
java.io.*;
              import java.util.*;

              public class Solution{
               List<String> getPalindromaticPairs(String[] input) {
                 Set<String> result = new HashSet<>();
                 if (input == null || input.length == 0) {
                   return new ArrayList<>();
                 }

                 // Step 1: put the reversed order of each word into
```

```
the map
   Map<String, List<Integer>> map = new HashMap<>();

   for (int i = 0; i < input.length; i++) {
     String str = input[i];
     String reStr = reverse(str);
     if (!map.containsKey(reStr)) {
       List<Integer> indices = new ArrayList<>();
       indices.add(i);
       map.put(reStr, indices);
     } else {
       List<Integer> indices = map.get(reStr);
       indices.add(i);
     }
   }

   // Step 2: Iterate each word
   for (int i = 0; i < input.length; i++) {
     String str = input[i];

     // Get all the prefix of str, and append to the
end
     for (int j = 1; j <= str.length(); j++) {
       String prefix = str.substring(0, j);
       String postfix = str.substring(j);

       if (map.containsKey(prefix) &&
isPalindrome(postfix)) {
```

```
         if (map.get(prefix).size() > 1
|| !map.get(prefix).equals(str)) {
           String palin = str + reverse(prefix);
           result.add(palin);
         }
       }
     }

     // Get all postfix of the str, and insert to
front
     for (int j = str.length() - 1; j >= 0; j--) {
       String postfix = str.substring(j);
       String prefix = str.substring(0, j);

       if (map.containsKey(postfix) &&
isPalindrome(prefix)) {
         if (map.get(postfix).size() > 1
|| !map.get(postfix).equals(str)) {
           String palin = reverse(postfix) + str;
           result.add(palin);
         }
       }
     }
   }

   return new ArrayList<String>(result);
 }

 private String reverse(String s) {
   if (s == null || s.length() <= 1) {
     return s;
   }

   char[] array = s.toCharArray();
   int lo = 0;
   int hi = array.length - 1;
   while (lo < hi) {
     char temp = array[lo];
     array[lo] = array[hi];
     array[hi] = temp;

     lo++;
     hi--;
   }

   return new String(array);
 }

 private boolean isPalindrome(String s) {
   if (s == null || s.length() <= 1) {
     return true;
   }

   int lo = 0;
   int hi = s.length() - 1;

   while (lo < hi) {
       if (s.charAt(lo) != s.charAt(hi)) {
         return false;
       }

       lo++;
       hi--;
     }

     return true;
   }

   public static void main(String[] args) {
     Solution solution = new Solution();
     String[] input = new String[]{"abc", "cba", "c",
 "c"};

     List<String> result =
 solution.getPalindromaticPairs(input);

     for (String s : result) {
       System.out.println(s);
     }
   }
 }
```

(Question 10): 3.1%   Boggle game (本来就已经写不完了，follow up: 1. 斜着走)

(Question 11): 3.1%   IP to CIDR

(Question 12): 2.5%   guess number (这道题看似概率很低，但是最近好像比较高频，特别注意)

(Question 13): 2.5%   queue using fixed size array(follow up: 1. 实现pop

```
import java.util.*;
public class Q13_Queue {

	public static void main(String[] args) {
		System.out.println("Hello World");
		QueueWithFixedArray queue = new QueueWithFixedArray(5);
		queue.offer(1);
		queue.offer(2);
		Integer res = queue.poll();
		System.out.println("res (1): " + res);
		queue.offer(3);
		queue.offer(4);
		queue.offer(5);
		queue.offer(6);
		queue.offer(7);
		queue.offer(8);
		queue.offer(9);
		queue.offer(10);
		queue.offer(11);
		queue.offer(12);
		queue.offer(13);
		queue.offer(14);
		res = queue.poll();
		System.out.println("res (2): " + res);
		res = queue.poll();
		System.out.println("res (3): " + res);
		res = queue.poll();
		System.out.println("res (4): " + res);
		res = queue.poll();
		System.out.println("res (5): " + res);
		res = queue.poll();
		System.out.println("res (6): " + res);
		res = queue.poll();
		System.out.println("res (7): " + res);
		res = queue.poll();
		System.out.println("res (8): " + res);
		res = queue.poll();
		System.out.println("res (9): " + res);
		res = queue.poll();
		System.out.println("res (10): " + res);
		res = queue.poll();
		System.out.println("res (11): " + res);
```

```
                res = queue.poll();
                System.out.println("res (12): " + res);
                res = queue.poll();
                System.out.println("res (13): " + res);
                res = queue.poll();
                System.out.println("res (14): " + res);
                res = queue.poll();
                System.out.println("res (14): " + res);



        }
}

class QueueWithFixedArray {
        private int fixedSize;

        private int count;
        private int head;
        private int tail;
        private List <Object> headList;
        private List <Object> tailList;

        public QueueWithFixedArray(int fixedSize) {
                this.fixedSize = fixedSize;
                this.count = 0;
                this.head = 0;
                this.tail = 0;
                this.headList = new ArrayList < >();
                this.tailList = this.headList;
        }

        public void offer(int num) {
                if (tail == fixedSize - 1) {
                        List < Object > newList = new ArrayList < >();
                        newList.add(num);
                        tailList.add(newList);
                        tailList = newList; // (List < Object > ) tailList.get(tail);
                        tail = 0; // tail = 0 has stored the new element, so tail = 1 now
                } else {
                        tailList.add(num);
                }
                count++;
                tail++;
        System.out.println("headList: " + headList);
        System.out.println("tailList: " + tailList + " tail: " + tail);
    }

    public Integer poll() {
        if (count == 0) {
            throw new NullPointerException("!!! reaching the end of headList");
            //return null;
        }

        int num = (int) headList.get(head);
        head++;
        count--;

        if (head == fixedSize - 1) {
            List <Object> newList = (List<Object>) headList.get(head);
            headList.clear();
            headList = newList;
            head = 0;
        }

        System.out.println("headList: " + headList + "head: " + head);
        System.out.println("tailList: " + tailList);
        return num;
    }

    public int size() {
        return count;
    }
}
```

(Question 14): 1.9%   wiggle sort

(Question 15): 1.3%   meeting interval (问法1：merge interval problem， 问法2： 找出空闲区间， follow up: 找出至少k个员工工作的区间)

(Question 16): 1.3%   display page

(Question 17): 1.3%　给一个[["A", "B", "C"],["A", "C", "D"]

(Question 18): 1.3%   find buddy

(Question 19): 1.3%   csv parsing

(Question 20): 1.3 %   bank system

(Question 21): 1.3%   Minimum Vertices to Traverse Directed Grap