阿里云 开发者社区 | 面经

算法笔试模拟题精解合集

程序员面试宝典

“涵盖70+算法题目、近30种大厂笔试常考知识点
算法学习看这本书就够了”

免费刷题神器
海量题目等你来挑战

阿里云开发者“藏经阁”
海量免费电子书下载

目录

一、算法思想

1.1 排序

算法笔试模拟题精解之“数组变换”

简介：本题要分情况讨论，根据不同的情况变换不同的解决方式。

题目描述

等级：中等

知识点：排序、贪心

查看题目：数组变换

给出一个长度为 n 的数组，和一个正整数 d。

你每次可以选择其中任意一个元素 a[i] 将其变为 a[i] + d 或 a[i] − d，这算作一次操作。

你需要将所有的元素全部变成相等元素，如果有解，请输出最小操作次数，如果无解请输出 −1。

输入数字 n、数字 d，和一个长度为 n 的数组 a。1 <= n <= 100000，1 <= d <= 100, 1 <= a[i] <= 100000。

输出一个数字，表示最小的操作次数，如果无解输出 −1。

示例 1

输入：

5

2

[3,5,7,1,9]

输出：

6

注意

最优解为全部变为 5，共 1 + 0 + 1 + 2 + 2 = 6 次。

解题方法：

首先判断无解的情况，可以发现 a[i]，a[i]+d, a[i]−d 在 模 d 情况下的余数不会发生改变，因此如果原数组中的存在任意两个数字它们对 d 取余结果不同，那么此时无解。

设余数为 r。判断完无解之后，需要求出最小值。

先将数组 a[i] 排序，然后除以 d，得到从 r 变成 a[i] 需要的步数。

枚举元素 a[i]，将所有元素全部变成 a[i] 需要考虑两部分，i 之前和 i 之后 : 对于 i 之前的元素，假设都是 r，那么需要 (i−1)*a[i]，但是因为并不都是 0，所有我们可以用一个变量 val 存放前 i−1 项的和，然后我们在减去 val 就是前 i−1 个元素真正需要操作的步数。

对于 i 之后的元素，也是类似的。我们假设 i 之后的所有项和为 val，假设我们要将它们变为 r，则消耗即为 val，但是我们只需要将其变为 a[i]，因此需要减去 (n−i)*a[i]。

看完之后是不是有了答题思路了呢，快来练练手吧：数组变换

算法笔试模拟题精解之“打怪兽”

简介：根据题意，本题可使用贪心算法完成，策略是每次打怪兽都选择代价最小的一只。

题目描述

题目等级：容易

知识点：排序、贪心

查看题目：打怪兽

现在有 3 只怪兽，他们的都有自己的血量 a,b,c(1<=a,b,c<=100)，当 Tom 打死第一怪兽的时候花费的代价为 0，其余的怪兽的代价为当前的怪兽的血量减去上一个怪兽的血量的绝对值。问 Tom 打死这些怪兽所需要的最小代价？

分别输入三只怪兽的血量；

输出打死三只怪兽的最小代价。

示例 1

输入：

2
5
8

输出：

6

解题思路：贪心

根据题意，本题使用贪心算法完成，策略是每次打怪兽都选择代价最小的一只。

由于第一次打怪兽的花费为 0，所以第一次可以打血量最小的（最大的也可以），接下来每次选择怪兽的时候就可以选择花费代价最小的。由于每次打怪兽的代价都是：当前怪兽的血量和上一个怪兽的血量的差的绝对值。于是可以得出结论，最小代价为所有怪兽血量的最大值减最小值。

时间复杂度： O(1)

空间复杂度： O(1)

趁热打铁，题目直达链接：打怪兽

1.2 贪心

算法笔试模拟题精解之“最大边权和”

简介：根据题意，最终需要将 n 个点连通并达到最大边权，而边权为两个点的点权之和的一半，所以一个点加入连通图的最大边权就是和点权最大的点连通。

题目描述

题目等级：容易

知识点：贪心

查看题目：最大边权和

现在有 n 个点 (1<=n<=1000)，每个点都有一个值称为点权 ai(ai 为偶数，1<=ai<=1000)，现在可以将任意两个点相连，连起来以后这条边也有一个值称为边权，这个边的边权为这两个点的点权之和的一半。现在需要你添加 n−1 条边，问将这 n 个点连通以后 (连通是指任意两个点都能互相到达) 的最大的边权和是多少？

输入点的数量 n；和 n 个数，表示点权的值

输出最大的边权和

示例 1

输入：

5
[2,4,6,8,10]

输出：

30

解题思路：贪心

根据题意，最终需要将 n 个点连通并达到最大边权，而边权为两个点的点权之和的一半，所以一个点加入连通图的最大边权就是和点权最大的点连通。

因此想要得到最大边权和，只要所有点都与点权最大的点相连即可。

实现过程中，首先求出最大的点权，然后计算出其他点与这个权值最大的点的边权之和即可。

时间复杂度：O(n)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：最大边权和

算法笔试模拟题精解之“最强的团队”

简介：根据题意，最强团队即团队中每个小队的能力值都是最高的。即解决这道题需要找出数组中连续最大值的个数，若有多个连续最大值，选择个数最多的。

题目描述

题目等级：简单

知识点：贪心

查看题目：最强的团队

有一个阵营，里面有 n 个小队 (1<=n<=100)，每个小队都有他们的能力值 ai(0<=i

现在有一个紧急情况，需要从这些小队中选出连续的几个小队，组成一个最强的团队。最强的团队的定义为这个团队的所有小队的平均能力值最高。如果有多个最强团队，则选包含小队最多的一个。

现在请你写个程序，输出这个最强的团队包含的小队的个数。

输入小队的数量 n，和 n 个数，分别代表各小队的能力值 ai

输出一个数表示这个最强团队包含的小队的个数。

示例 1

输入：

6
[1,2,3,3,2,1]

输出：

2

解题方法

根据题意，最强团队即团队中每个小队的能力值都是最高的。即解决这道题需要找出数组中连续最大值的个数，若有多个连续最大值，选择个数最多的。

具体实现时，可以先找出数组中最大的能力值是多少，然后设置一个标记 tag。接着遍历数组，比较每个数组元素和最大值，数组元素等于最大的值的时候，将 tag 标记为 1，数组元素不等于最大值时，将 tag 置为 0。

在 tag 等于 1 时统计连续最大值的数量，若统计到多个最大值，则记录最大的那个。

时间复杂度：O(n^2)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：最强的团队

算法笔试模拟题精解之“Tom 爱吃巧克力”

简介：根据题意，可以得知这道题可以运用贪心算法，策略是每次都去买最便宜的巧克力。

题目描述

题目等级：容易

知识点：贪心

查看题目：Tom 爱吃巧克力

Tom 非常喜欢巧克力，他上次买的巧克力吃完了，所以他打算再去买 k 块巧克力回来 (1<=k<=1e5)，他又是一个非常节俭的一个人，所以他想花最少的钱去买巧克力。

现在有 n 家卖巧克力的店 (1<=n<=1e5)，每个店的巧克力都限购 bi 块 (最多只能买 bi 块 ,1<=bi<=1e5)，每块的价格是 ai(1<=ai<=1e9)，请问 Tom 买 k 块巧克力最少要花多少钱？题目保证 n 个 bi 的总和大于等于 k。

输入卖巧克力的店的个数 n(1<=n<=1e5)；打算去买的巧克力块数 k(1<=k<=1e5)；和一个数组 m, 其中 mi =ai, bi 表示第 i 家巧克力店的巧克力的价格和限购块数

输出一个数，表示 Tom 买 k 块巧克力花的最少钱数

示例 1

输入：

2

5

[[4,5],[2,4]]

输出：

12

解题思路：贪心

根据题意，可以得知这道题可以运用贪心算法，策略是每次都去买最便宜的巧克力。

由于题目给的二维数组是乱序的，可以根据巧克力的价格对二维数组从小到大进行排序，便于 Tom 选择最便宜的巧克力。Arrays 类中只提供了一维数组的排序，如果要用 Arrays 对二维数组排序，需要重写 Comparator 里的 compare 方法。

排序完成后，接下来操作就比较简单了。遍历数组，优先买便宜的巧克力，直到达到限购块数，再去下一家巧克力店。最终买到 k 块巧克力时 Tom 花的钱最少。

时间复杂度：O(n)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：Tom 爱吃巧克力

算法笔试模拟题精解之“吃奶酪”

简介：根据题意，如果要花费最少时间，则每个奶酪都让离奶酪最近的人去拿，因此，坐标 <=50000 的奶酪让 Tom 去拿，坐标 >=50001 的奶酪让 Jerry 去拿。

题目描述

题目等级：容易

知识点：贪心、枚举

查看题目：吃奶酪

Tom 和 Jerry 都很喜欢吃奶酪，现在有 n 块奶酪散落在坐标轴上 (1<=n<=100000)，他们分别在 a1,a2,a3...an(1<=ai<=100000, 一个点可以有多块奶酪) 上，Tom 和 Jerry 分别在 1 和 100000 两个点上，他们每走一步需要花费 1s，问他们拿到所有的奶酪至少要花费多少时间

输入奶酪数量 n，和 n 个奶酪的坐标

输出一个数，表示他们拿到所有奶酪所用的最短时间

示例 1

输入：

4
[350,2000,80000,99999]

输出：

20000

解题方法

根据题意，如果要花费最少时间，则每个奶酪都让离奶酪最近的人去拿，因此，坐标 <=50000 的奶酪让 Tom 去拿，坐标 >=50001 的奶酪让 Jerry 去拿。

具体实现时，可以设置一个 time 值，然后遍历数组。判断每一块奶酪的坐标范围，根据坐标判断应该让谁拿，再计算拿到这个奶酪需要多长时间，如果时间大于 time，则用这个值替换掉 time 的值。

用这种方法，遍历整个数组后的 time 值即为 Tom 和 Jerry 拿到所有奶酪所用的最短时间。

时间复杂度：O(n)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：吃奶酪

算法笔试模拟题精解之“一的个数”

简介：根据题意，本题的所有数字应从二进制角度考虑。

题目描述

题目等级：容易

知识点：位运算、贪心

查看题目：一的个数

给你两个数字 l、r，问在区间 [l,r] 内的所有数中，二进制表示下“1”的个数最多的一个数是多少，如果有多个相同的，输出所有符合条件的数中最小的一个数。

输入一个整数 l, 和一个整数 r。(1<=l<=r<=10^9)

输出一个数字表示 [l,r] 内二进制下“1”的个数最多的数。如果有多个，输出符合条件的数中最小的。

示例 1

输入：

5
10

输出：

7

解题思路：二进制

根据题意，本题的所有数字应从二进制角度考虑。

所求数字可分为两部分，高位部分和低位部分，高位部分的值等于 l, r 高位相等的部分，在区间 [l,r] 中的所有数的高位部分都应该与其相等，即

high = r & (−1<

低位的部分计算过程如下：

如果 r−high 的值的二进制全为 1，则低位部分为 r−high。如果不是全为 1，则低位部分为

(1<<(count−1)) − 1

时间复杂度：O(log_2 n)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：一的个数

算法笔试模拟题精解之“Bob 的花束”

简介：本题充分理解题意后，直接模拟这个“选取最大值”的过程就可以得到结果了。

题目描述

等级：中等

知识点：贪心

查看题目：Bob 的花束

Bob 和 Alice 是青梅竹马。今天，Bob 终于要鼓起勇气向 Alice 表白了！说到表白，自然是少不了买花了。Bob 来到了花店，花店一共提供了 9 种花，每一种花都有对应的价钱。但是 Bob 的零花钱有限，不能把所有的花都买下来送给 Alice。

为了方便挑选，Bob 给这 9 种花分别标号 1-9，Bob 希望买到的花按照编号可以排出尽可能大数字，请问 Bob 能够排出的最大的数字是多少?

输入一个正整数 value，代表 Bob 拥有的零花钱。(0<=value<=10^6)

和有 9 个数字的数组 a，ai 代表第 i 种花的价格。(1<=ai<=10^5,1<=i<=9)

输出一个数字，表示 Bob 可以排出的最大数字。如果 Bob 不能排出任何一个数字，则输出 -1。

示例 1

输入：

2

[9,11,1,12,5,8,9,10,6]

输出：

33

注意：

花店的每一种花都可以视为无限多。

解题方法：模拟选花过程

本题充分理解题意后，直接模拟这个“选取最大值”的过程就可以得到结果了。

首先，选取最大值，意味着**首先这个结果的“位数”要足够多**，比如假设所有的花价格都是 1 元钱，则 11111111 是花 9 块钱能买到的最大值，而不是 333 或者 9 这样的方案。这样相当于根据输入，输出的位数可以用很小的时间代价来确定：**“用可用钱数，除以最便宜的花的价格，并向下取整”**。假设这里的位数为 m。

其次，在位数确定的情况下，高位数字越大，结果也就越大，比如同样是 8 元钱，只能购买价格为 5 的 5 号花，和价格为 3 的 3 号花时，购买 35 就是最差的方案，而 53 才是正确答案。而且因为每个花的数量是无限的，所以可以模拟这个 **“从高位开始，逐个选取能买得起的最大的数字；同时每选完一位后，要确保剩下的钱，依旧可以买到 m 位数字的组合方案。”**

提示：根据上面逻辑写出的答案，在充分理解优化后，至少可以达到 2 遍扫描数组得到结果。

时间复杂度：O(9n)

看完是不是有了新思路了呢，快来试一下吧：查看题目：Bob 的花束

算法笔试模拟题精解之“钱庄”

简介：可以先分析示例是如何实现的，以此为突破点。

题目描述

等级：中等

知识点：贪心

查看题目：钱庄

钱庄每天能够收到很多散钱，第 i 个散钱的值 2^wi。为了便于管理，钱庄每天都会向中央银行申请兑换钱币，假设钱庄有一些散钱使得 2^k1+2^k2+...+2^km=2^x（x 为非负整数），那么就可以将这些散钱兑换成一个大钱币，问在钱庄收到的这些散钱最终最少能变成几个钱币?

输入一个整数 n，表示一共有 n 个钱币 (1 <= n <= 10^6)；再输入 n 个整数 wi，表示有价值 2^wi(0 <= wi <= 10^6) 的钱币。

输出兑换后最少的钱币数

示例 1

输入：

4
[1, 1, 2, 3]

输出：

1

注意

2^1+2^1+2^2+2^3=2^4，因此兑换后最少为一个钱币

解题思路一

大致思路：对于 [1, 1, 2, 3] 样例，答案 1 是怎么算出来的？按照题目的思路，应该这样算：2^1+2^1+2^2+2^3=2^4, 因此为 1，但是如果这样做，肯定会超时，我是这样算的：对于底数为 2 幂数相同的两个数想加，是不是底数不变，幂加 1，因此两个 1 组成一个 2，此时共有两个 2，这两个 2 组成一个 3，此时共有两个 3，两个三组成一个 4，最后只剩一个 4，因此答案为 1.

具体过程：遍历一遍 m 找出 m 中的最大数 max，定义一个数组 **arr[max+2],** 用于统计出 m 数组中，每个数字出现的次数，即 arr[m[i]]++（**m 中的元素为下标，arr 数组中保存出现的次数**）;

再次遍历 arr 数组 (0<=i<=max)，如果当前元素 arr[i]==0，那就下一个，

如果不为 0，arr[i+1]+=arr[i]/2;(**每两个 arr[i] 就能凑一个 arr[i+1])**

如果 arr[i] 为奇数，那说明剩余一个 arr[i]，这最后也就剩下了，所以 ans++；

遍历完后 if(arr[max+1]!=0) ans++; 然后 return ans;

注意：加粗部分一定要理解

解题思路二：贪心

遍历钱币数组 m, 只要数组中的当前元素 x 可以兑换一个大金币（有两个以上相同的钱币）就自底向上兑换，直到无法兑换；

遍历结束后，对剩下的钱币个数做统计（而上述操作保证了所有钱币均不可兑换）;

复杂度分析

空间复杂度

开辟了 100000 大小的 map 数组 (因为价值 wi 取值为 :0 <= wi <= 10^6)

所以**空间复杂度为 : O(100000) ~ O(1)**

时间复杂度

针对钱币数组遍历一次 , 每一次会做自底向上的钱币转换遍历 ;

所以最差情况下时间复杂度 , 每次都会自底向上发生转换 ;

所以**时间复杂度最差为 : O(n^2)**, 其中 n 为钱币数 ;

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：钱庄

算法笔试模拟题精解之“移动射击”

简介：首先理解题意，题目说的“发动之后只能改变一次方向”是干扰你的，因为即使你在中间过程中左右摆，但宏观上还是最多改变了一次方向。

题目描述

题目等级：中等

知识点：DP

查看题目：移动射击

你正在数轴上跟小精灵对战。你拥有一个十分强力的技能称为移动射击，但是这个技能有一个缺点是在你发动之后只能改变一次方向。

你可以认为你的位置在数字 0 的位置上，在数轴的正方向上有 n 只精灵，负方向上有 m 只精灵。移动射击可以造成 w 点伤害。每个精灵都有自己的血量，当血量降为 0 时死亡。

在最开始时你可以选择向正方向或负方向释放移动射击，并且可以在任意时刻改变技能的方向。请问你最多可以击杀多少只小精灵 ?(n，m，w 以及精灵的血量均在 [1, 100000] 范围内)

输入内容为五个，前三个为三个数字 ：正方向上的精灵个数 n、负方向上的精灵个数 m, 移动射击可以造成的伤害 w；第四个是一个长度为 n 的数组 a，表示正方向上的 n 个精灵的血量；第五个是一个长度为 m 的数组 b，表示负方向上的 m 个精灵的血量。

输出一个数字，表示最多能够击杀的精灵数量。

示例 1

输入：

3

4

10

[1, 2, 3]

[4, 3, 2 ,1]

输出：

4

解题思路

大致思路：首先理解题意，题目说的“发动之后只能改变一次方向”是干扰你的，因为即使你在中间过程中左右摆，但宏观上还是最多改变了一次方向。

如果说：我先杀左边一个，然后转头杀右边一个，再转头杀左边三个，又回头杀右边 1 个，看起来是不是改变了三次方向，其实呢，相当于我先杀左边四个，再回头杀右边两个，效果是一样的。因为你想这个问题的时候，可以忽略这个限制。

具体过程：先遍历数组 a，a[i] 表示数组 a 前 i 个数的和，当 a[i]>=w 的时候，记住此时的位置 index_a=i，退出循环，退出后加上这句 if(i==n||a[i]>w) index_a--；因为 index_a 指向的是刚好不超过 w 的位置，而且不能越界。

对 b 数组也是如此，然后开始从 index_a 往后一步一步走；

走一步，看看 b 数组的情况，k 为 b 数组的下标，初始 k=0;

```
while(k<m &&a[i]+b[k]<=w) k++;
```

然后和当前最长的长度比较

```
ans=Math.max(ans,i+k+1);
```

当 index_a 一直走到底，可返回 ans.

看完之后是不是有了想法了呢，题目直达链接：查看题目：移动射击

算法笔试模拟题精解之“相似数组”

简介：要解出相似数组的最长长度，即要求相似数组中的每个元素尽可能的小，把握这一点，结合下来的算法过程理解一下。

题目描述

等级：中等

知识点：贪心、尺取法

查看题目：相似数组

现在有两个数组 a 和 b ，长度分别为 n 和 m。你可以对两者进行任意次数（包括零次）的下述操作：

任选一段连续的区间 [l, r]，将其替换为这段区间的所有数字的和。比如，对于 [1, 3, 4, 5, 11, 9]，你可以选择区间 [3, 5]，并将其替换为 4+5+11=20，操作后的数组为 [1, 3, 20, 9]。

你现在需要通过上述操作将两个数组变成相同的数组，相同的定义是：对于两个数组 a，b 必须长度相同，不妨设为 l，并且对于 1 <= i <= l，必有 a[i]=b[i]。

如果这两个数组可以变成相同的数组，那么我们称这两个数组是相似数组，否则不是相似数组。我们并不在意操作的次数，我们只在意在这两个数组经过操作之后变成相同数组的时候最长的长度是多少，如果它们本来不相似请输出 −1。

输入内容为四个部分，先两个数字 n、m（1 <= n, m <= 100000），分别表示数组 a 和 b 的长度，再分别输入含有 n 个数字的数组 a 和含有 m 个数字的数组 b，其

中 1 <= a[i], b[i] <= 1000000000。

输出一个数字，表示最长的长度。

示例 1

输入：

5
4
[7,2,5, 11, 13]
[9 ,16 ,6 ,7]

输出：

3

解题思路

要解出相似数组的最长长度，即要求相似数组中的每个元素尽可能的小，把握这一点，结合下文的算法过程理解一下。

设两个指针，分别指向数组 a 和 b 的第一个位置（即 i=0，j=0），定义两个变量，分别表示数组 a 和 b 的当前区别的和（sum_a=a[0], sum_b=b[0]），然后遍历数组。

如果 sum_a==sum_b：

则结果加 1（即 ans++），然后对两个指针分别加 1(i++, j++)。

判断：如果两个指针都等于各自数组的长度（即 i==n&&j==m），则返回结果 (return ans)；

如果两个指针仅有一个等于数组的长度（即 i=n || j=m），则返回 -1（表示不是相似数组）；

如果以上两个条件都不满足 (即两个指针都小于数组长度)，则当前区间和更新为此时指向的元素 (即 sum_a=a[i], sum_b=b[j])。

如果 sum_a 则数组 a 的指针向前移动 (即 i++)，判断 i 是否越界 (即 i==n 为真表示越界)，如果越界，那么返回 −1(表示不是相似数组)，

如果没越界，给区别和加上当前元素 (即 sum_a+a[i])

如果 sum_a>sum_b:

则数组 b 的指针向前移动 (即 j++)，判断 j 是否越界 (即 j==m 为真表示越界)，如果越界，那么返回 −1(表示不是相似数组)，

如果没越界，给区别和加上当前元素 (即 sum_b+a[b])

重复以上过程，即可求解。

是不是有思路了呢，点击链接立刻答题：相似数组

算法笔试模拟题精解之“过吊桥”

简介：根据题意，要知道 B 同学还能在桥的一头逗留的时间，需要先求出什么时候有连续的两块木板坏掉，或者第一块或者最后一块木板坏掉。

题目描述

题目等级：容易

知识点：贪心

查看题目：过吊桥

B 同学在机房敲了半个多月的代码之后终于打算出门玩一玩了。这天他准备去爬山，当爬到了半山腰时，发现了一个吊桥。

这个吊桥总共由 n 块标号为 1-n 的木板组成，由于年久失修，这些木板有些已经快要坏掉了，每块木板都有一个值 ai 表示第 i 块木板还有 ai 分钟就要坏掉了，即在第 ai+1 分钟将无法站上这块木板。

B 同学过吊桥时一步只能走一块或两块木板，但是他想在吊桥的这边多玩一会。

请问他在吊桥这边最多可以玩多长时间? (可以认为 B 同学能在一分钟内通过吊桥) 特殊的，如果第一块或者最后一块木板坏掉的话吊桥就直接无法通过了。

输入一个整数 n, 表示总共有 n 块木板 (1<= n <= 10^5)。

再输入一个包含 n 个整数的数组，第 i 个数表示第 i 块木板还有 ai 分钟就要坏掉了 (1 <= ai <= 10^9)。

输出一个整数表示 B 同学还能在桥的一头逗留的时间。

示例 1

输入：

4

[10,3,5,10]

输出：

5

注意

在第五分钟，还剩 124 三块木板可以通过，在第六分钟只剩下 14 两块木板就无法通过了。

解题思路

根据题意，要知道 B 同学还能在桥的一头逗留的时间，需要先求出什么时候有连续的两块木板坏掉，或者第一块或者最后一块木板坏掉。

首先将数组存入 HashMap 中，以数组的数为 key，索引的位置为 value（因为数组中有可能会有相同的数字，而 value 值要存储所有数字的索引，所以索引只能以 List 的形式存在 value 中）。这样存储以后，只要知道数组中的数字，就能快速找到其索引。

接下来对数组进行升序排序，再创建一个大小相同的数组 status 用来记录木板的状态，然后遍历数组。

每遍历一个数，就通过 HashMap 查找其索引，在 status 数组中根据这个索引将这个木板的状态置为 -1，代表木板坏掉了。然后判断是否符合不能通过桥的条件，若符合条件，此时在数组遍历的数即为可以逗留的时间。若不符合条件，则继续遍历数组，重复上述步骤，直到符合条件为止。

时间复杂度：O(n)

空间复杂度：O(2n)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：过吊桥

算法笔试模拟题精解之“完美排列”

简介：本文通过两种解法描述 86 题“完美排列”的解题过程，更有对应的时间和空间复杂度帮助理解。

题目描述

等级：容易

知识点：贪心

查看题目：完美排列

完美排列的定义为一个长度为 n 的数组，n 个元素各不相同且每个元素均在 [1,n] 的范围内。

现在给你长度为 n 的数组，你每次可以进行如下操作：任选数组中的一个元素，将其加一或者减一，问最少需要多少次操作才能够使得该数组为一个完美排列。

输入一个整数 n，表示数组的长度 (1 <= n <= 10^4)；

再输入含有 n 个数的数组，第 i 个数表示数组中的第 i 个元素为 ai (1 <= ai <= 10^5)。

输出一个整数表示将该数组变成一个完美排列的最少操作次数。

示例 1

输入：

2

[3,0]

输出：

2

注意：

3->2

0->1

总共需要操作两次。

解法探究

解法一：正常贪心思路

本题是一道典型的贪心算法题，问题可以通过每步的最优策略分治解决。如果将 n 个大小未知的正整数，通过题目中的规则“填充”到槽 1~n 中，我们不妨从最小的数字槽 1 开始做起。

显然，这 n 个正整数中最小的数字 a（无论这个最小的数字是 1 或是 100），是填充槽 1 的最佳选择。其它 (n-1) 个数字和 1 的“距离”，都必定大于等于 a，距离槽 1 的距离都不如 a 更优，所以可以将 a 填充进槽 1，并在后续选择中排除掉它。

当填充槽 2 时，依旧用上面的思路就可以了。用剩下的 (n-1) 个数字中最小的数字去通过加减 1 进入槽 2，必定是填充槽 2 所有方式中的最佳策略。

将上面的思路重复应用，就得到了结果。复杂度上需要扫描 n 次数组。

时间复杂度：O(n^2)

空间复杂度：O(1)

解法二：排序优化

上面的反复扫描非常浪费时间，不如提前对数组排序，然后从排序后递增数组的第一项开始，逐个比较与槽 1~n 的距离，最后加到一起，得到答案。

时间复杂度：O(logn)

空间复杂度：O(1)

看完之后是不是有了答题思路了呢，快来练练手吧：查看题目：完美排列

算法笔试模拟题精解之“采摘圣诞果”

简介：我们定义数组 a[i] 表示第 i 天可以采摘的刚刚结出来的果子，数组 b[i] 表示第 i 天可以采摘的已经过了一天的果子。根据输入先初始化 a[]。

题目描述

等级：中等

知识点：贪心

查看题目：采摘圣诞果

圣诞节马上就要来了，果园里的 n 棵圣诞树马上就要结果子了，每棵圣诞树会在第 a[i] 天结出 b[i] 个果实。果园里有许多圣诞小精灵，它们非常喜欢吃圣诞果，如果在结果后两天内也就是第 a[i] 天和第 a[i]+1 天，没有将果实采摘下来，那么将会被小精灵们偷吃掉。

你，作为圣诞树的看守者，必须采摘尽可能多的圣诞果，但是你每天最多只能采摘 v 个圣诞果，当然，可以是不同的果树上的。现在你需要判断自己最多可以收获多少圣诞果。

输入圣诞树棵树 n、每天最多采摘的圣诞果数量 v 和一个数组 m，其中 m[i]=[a[i], b[i]] 表示每棵圣诞树第 a[i] 天结出 b[i] 个果实（1 <= n，a[i]，b[i] <= 3000）。

输出一个数字，表示最多可以收获的圣诞果数。

示例 1

输入：

3

3

[[1,3],[2,5],[3,4]]

输出：

12

注意

你可以在第一天在第一棵树上摘三个，第二天在第二棵树上摘三个，第三天在第二棵树上摘两个，然后再在第三颗树上摘一个，第四天在第三棵树上摘三个。

解题思路描述

我们定义数组 a[i] 表示第 i 天可以采摘的刚刚结出来的果子，数组 b[i] 表示第 i 天可以采摘的已经过了一天的果子。根据输入先初始化 a[]。

假设我们当前处于第 i 天，那么显然我们应该采摘 b[i] 中的果实，然后再采摘 a[i] 中的果实，因为如果不采摘 b[i] 中的果实，则它们会在下一天被偷吃。在更新之后 a[i] 中剩余的果实会在下一天变成 b[i] 中的果实。

时间复杂度：O(3000)

空间复杂度：O(3000)

看完之后是不是有了答题思路了呢，快来练练手吧：采摘圣诞果

算法比赛模拟题精解之“Tairitsu and Dynamic Objects”

简介：根据题意，分析得知，Hikari 和 Tairitsu 每次会优先选择 ai+bi 的值最大的物品，当物品的 ai+bi 值相等时，选择 bi 大的那个。

题目描述

题目等级：容易

知识点：贪心

查看题目：Tairitsu and Dynamic Objects

Hikari 和 Tairitsu 面前有 n 个物品，这些物品编号为 1,2,...,n。

每个物品有两个属性。第 i 个物品的两个属性分别为 ai, bi 。

初始 n 个物品均可被选取。Hikari 与 Tairitsu 会轮流选取当前可选取的物品中的一个，并把它拿走，这个物品之后不可被选取。第一轮 Hikari 先选取。

设 Hikari 选取的物品编号的集合为 H ，Tairitsu 选取的物品编号的集合为 T 。

所有物品均被选取完之后，Hikari 得分为 $\sum ai(i \in H)$ ；而 Tairitsu 得分为 $\sum bi(i \in T)$ 。

Hikari 和 Tairitsu 都希望自己的得分比对方大，你需要求出双方都使用最优策略的情况下，双方各会获得多少分。

注意：若对于某个人来说，剩余的物品中有多个对两人分数大小关系影响相同的物品，那么他会优先选择 bi 最大的那个。

输入一个正整数 n，表示物品个数。

再输入两个数组 a 和 b，分别表示 n 个物品的 A 属性和 n 个物品的 B 属性。（保证 1<=n<=2*1e5, 0<=ai,bi<=1e9）

输出一行，两个整数分别表示 Hikari 和 Tairitsu 的得分。

示例 1

输入：

5
[1,2,3,4,5]
[5,4,3,2,1]

输出：

[9,6]

解题思路：

根据题意，分析得知，Hikari 和 Tairitsu 每次会优先选择 ai+bi 的值最大的物品，当物品的 ai+bi 值相等时，选择 bi 大的那个。

因此，可以对物品进行排序，排序有两个依据，优先依据 ai+bi 的值，然后是 bi 的值。降序排好后，Hikari 和 Tairitsu 依次按顺序选择物品，Hikari 先选择，选择好以后，分别计算 Hikari 和 Tairitsu 的分数即可。

时间复杂度：O(n)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：Tairitsu and Dynamic Objects

算法笔试模拟题精解之“Codancer 的炸弹引爆”

简介：花费 8 电力引爆第 3 枚炸弹，那么第 1 枚就会被自动引爆，那么第 2 枚也会被自动引爆。这种方案的花费是最小的。

题目描述

等级：困难

知识点：贪心、优先队列

查看题目：Codancer 的炸弹引爆

Codancer 终于抵达了恶龙的城堡。现在他在城堡周围摆放了 n 枚电力炸弹，每个电力炸弹有两种属性 m 和 p，只有已经引爆了 m 枚电力炸弹或者 Codancer 直接花费 p 的电力，第 i 枚炸弹才会被引爆，现在 Codancer 想使用最少的电力引爆所有的炸弹，请计算最少需要多少电力?

第一行是一个正整数 n，代表有 n 枚电力炸弹。接下来输入 n 行，每行两个正整数 m 和 p，代表炸弹的属性。(1<=n<200000,1<=p<=100,1<=m<=n)

输出最少花费多少的电力。

示例 1

输入：

3
[[1,5],[2,10],[2,8]]

输出：

8

注意

花费 8 电力引爆第 3 枚炸弹，那么第 1 枚就会被自动引爆，那么第 2 枚也会被自动引爆。

解题方法：

花费 8 电力引爆第 3 枚炸弹，那么第 1 枚就会被自动引爆，那么第 2 枚也会被自动引爆。

这种方案的花费是最小的。

炸弹的引爆顺序如图所示。

本题使用贪心策略，考虑按照所需要的电力从大到小排序，假设从第 i+1 枚到第 n 枚炸弹已经用电力引爆的炸弹个数为 cnt，由于在 [1,i−1] 最多再引爆 i−1 枚炸弹，如果此时 `i-1+cnt<mi`，说明就需要再花费电力引爆，但是必须要选择需要的电力最少的，因此可以用优先队列维护，直接取出栈顶即可。

时间复杂度：O(n*log（n）)

看完之后是不是有了答题思路了呢，快来练练手吧：Codancer 的炸弹引爆

算法笔试模拟题精解之“学习小组”

简介：因为题目中说了 ai 是一个非递减的数列，所以我们可以推导出一个式子。

题目描述

等级：中等

知识点：贪心

查看题目：学习小组

在一个课堂上，有 n 个学生 (1<=n<=3e5)，每个学生都有他们自己的学分 ai(1<=ai<=1e9,ai<=ai−1)，现在老师想将他们分为 m 个小组 (1<=m<=n)，为了方便交流，所有的小组都是由相邻的学生组成 (abc 相邻 , 不存在 ac 一个小组 b 在另一个小组的情况)，现在老师想让每个小组的学分差值尽量小 (最大值减去最小值)，请你帮助老师来分一下组，输出最后的每个小组的最小的差值的总和。

第一行和第二行输入两个数 n、m 表示有 n 个学生要分成 m 个小组，再输入 n 个数，表示每个学生的学分。

输出一个数字，表示最后分出的 m 个小组的最小的差值的总和。

示例 1

输入：

5

3

[1,3,5,7,9]

输出：

4

解题方法：

因为题目中说了 ai 是一个非递减的数列，所以我们可以推导出一个式子。

比如我们要将一个数组分成 3 组，那么可以假设为，[a1,ai],[ai+1,aj],[aj+1,an] 这三组，然后我们所要求的值就是 (ai − a1) + (aj − ai+1) + (an − aj+1) .

那么就可以推导出 **an − a1 + aj − aj+1 − ai − ai+1**，所以就是 an − a1 减去最大的 m−1 个相邻的差值，那么 ai−ai+1 这个显然是差分的性质，所以我们对于原数列求一个差分数组出来，然后对差分数组进行排序，贪心的去减去前 m−1 个最大的就好了。

看完之后是不是有了答题思路了呢，快来练练手吧：学习小组

算法笔试模拟题精解之“恢复字符串”

简介：本题首先可以确定最小值一定为 0。接下来要分两种情况来讨论。

题目描述

等级：中等

知识点：贪心

查看题目：恢复字符串

给出两个仅包含“+”、“−”两种字符且长度相同字符串 s1、s2，你需要通过填充数字将这两个字符串恢复成一个合法的表达式。并且只能填入正整数，恢复后的表达式的值必须非负。

例如对于字符串“+−”，你可以将其变成“1+1−2”，但是不可以变成“1+1−3”，也不可以变成“1+0−1”。定义你的消耗为你填充的所有正整数的和。比如“1+1−2”的消耗为 1 + 1 + 2 = 4。你需要将这两个字符串都恢复成合法表达式，并且尽量的使它们的差值最小，于此同时你还需要使你的消耗最小。

相信你通过思考已经发现最小差值总是 0，因此你只需要求出差值为 0 时的最小消耗即可。字符串长度都小于 100000。

输入两个字符串 s1 和 s2。

输出一个数字，表示最小消耗。

示例 1

输入：

“++−”

“−−+”

输出：

10

注意

样例最优解为“1+1+1−1”，“3−1−1+1”。

解题方法：

首先可以确定最小值一定为 0。分两种情况讨论。

1. 两个字符串都没有负号的时候，最优解的所有位置都填 1。最小消耗为 2*（n+1）。
2. 表达式中仅需要有一个负号，表达式的值就可以为任何值。此时两个表达式可以相互调整，因此最小差也是 0。

表达式中除了第一位以外每位数字填 1 可以得到最小消耗，因为值加在哪一位都是等效的。

不妨假设都加在第一位。计算出 s1,s2 的正负号数量的差，分别为 x 和 y。假设第一位分别为 pa，pb，我们只需要使（pa+x==pb+y）即可。

假设最终表达式的值为 final，则 final=max(max(x, y) + 1, 0)，则 pa=final−x，pb=final−y。

看完之后是不是有了答题思路了呢，快来练练手吧：恢复字符串

1.3 DP/ 动态规划

算法笔试模拟题精解之“矩阵最小路径和”

简介：本题可以用动态规划的方法来解决。计算一个格子到右下角的最小路径需要两个数据，一个是右边格子到右下角的最小路径，一个是下边格子到右下角的最小路径，两个数据的较小值加上当前格子的数值即为最小路径。

题目描述

难度：简单

知识点：数组、DP

查看题目：矩阵最小路径和

给定一个矩阵，大小为 m，从左上角开始每次只能向右走或者向下走，最后达到右下角的位置。路径中所有数字累加起来就是路径和，返回所有路径的最小路径和。

示例 1

比如输入矩阵为

4 1 5 3
3 2 7 7
6 5 2 8
8 9 4 5

最小路径为

23

解题思路：动态规划

本题可以用**动态规划**的方法来解决。

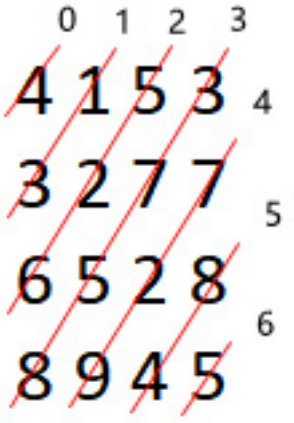
计算一个格子到右下角的最小路径需要两个数据，一个是右边格子到右下角的最小路径，一个是下边格子到右下角的最小路径，两个数据的较小值加上当前格子的数值即为最小路径。

即 dp[i, j] = min(dp[i + 1, j], dp[i, j + 1]) + m[i, j]

由于计算当前格子最小路径需要右边和下边格子的最小路径。因此，需要从底向上进行决策。

本题用动态规划法的难点之一是从底向上进行决策的顺序。

如下图所示，通过观察可以发现，同一对角线上的数字的横纵坐标和是相等的，我们以对角线的方向为顺序，从右下角向左上角计算出每个格子的最小路径。最后可计算得出 dp[0, 0]。



是不是有思路了呢，点击链接立刻答题：矩阵最小路径和

算法笔试模拟题精解之“寻找等比数列”

简介：最简单的方法，就是遍历数组中所有可能的三元组，逐个检验每组数是否符合公比为 k 的等比数列的性质。对于较长的用例，这么做显然是超时的，不通过。

题目描述

等级：中等

知识点：DP

查看题目：寻找等比数列

Alice 和 Bob 都是数学高手，有一天 Alice 给了 Bob 一个长度为 n 的数组 a，要求 Bob 在数组中找出 3 个数，要求三个数能够组成一个公比为 k 的等比数列，且不改变三个数在数组 a 中的位置关系。

为了降低难度，Alice 只要求 Bob 回答数组 a 中有多少个这样的等比数列。

输入一个整数 n 表示数组长度、公比 k(1<=n,k<=2*10^5) 和包含 n 个数的数组 a(−10^9<=ai<=10^9)

输出一个数字，表示数组 a 中包含 Alice 要求的等比数列的数量。

示例 1

输入：

5
2
[1,1,2,2,4]

输出：

4

题解思路

方法 1：逐个遍历（超时）

最简单的方法，就是遍历数组中所有可能的三元组，逐个检验每组数是否符合公比为 k 的等比数列的性质。

对于较长的用例，这么做显然是超时的，不通过。

时间复杂度：O(n^3)

空间复杂度：O(1)

方法 2：动态规划

本题充分利用下面两个隐藏规律，就可以使用动态规划：

1. “长度为 3 的等比数列”，满足了动态规划的最优子结构性质

数列中每个数字只有 4 种可能的状态。其中带 * 号的状态可以同时存在，且靠后的状态可以由前面的状态转换而来：

- 同时属于一个或多个等比数列的第 1 项
- 同时属于一个或多个等比数列的第 2 项
- 同时属于一个或多个等比数列的第 3 项
- 无法与前后数字组成任何等比数列

2. “不允许调换顺序”，满足动态规划的子问题不重叠性和无后效性性质

表示数字处于上面哪种状态，完全由这个数字及其之前的数字决定，它的状态与它后面数字的情形无关。

这就意味着，一次遍历，同时用 map 将每个遍历的数字归属到前面 4 种状态中的 1 个或多个里。遍历结束，输出第 3 种状态“同时属于一个或多个等比数列的第 3 项”中含有的数字数，就完成了动态规划的过程。

时间复杂度：O(n)

空间复杂度：O(kn)

看完之后是不是有了答题思路了呢，快来练练手吧：寻找等比数列

算法笔试模拟题精解之“字符配对”

简介：本题可以使用动态规划来解决，对于第 i 个字符，有选和不选两种，如果选，则和第 i−1 个字符组合创造权值，如果不选，就单独出来没有权值，取两者中加权最大的选择。

题目描述

题目等级：中等

知识点：DP

查看题目：字符配对

给你一个字符串，字符串中仅包含 "A","B", 现在有四种字符串 "AA","AB", "BA","BB"，每种字符串都有他们的权值，问从给出的字符串中能够得到的最大权值为多少 (一个字符只能属于一个子字符串) ?

输入一个字符串 s (1 <= |s| <=10^6) ;

再输入一个数组，数组中包含四个数字 a,b,c,d，依次表示字符串 "AA","AB", "BA","BB" 的权值 (1 <= a,b,c,d <= 100)。

输出权值的最大值。

示例 1

输入：

"ABA"
[1 ,2, 3, 4]

输出：

3

解题方法：DP

大致思路：动态规划问题，对于第 i 个字符，有选和不选两种，如果选，则和第 i−1 个字符组合创造权值，如果不选，就单独出来没有权值，取两者中加权最大的选择。

具体过程：定义一个字典 map，key 分别为 "AA","AB","BA","BB"，value 为对应的权值。定义大小为 n 的数组 dp，dp[i] 表示前 i 个字符组成的最大权值，毫无疑问，dp[0]=0，dp[1]=map.get(s.substring(0, 2));

从 i=2 开始遍历字符串，对于 dp[i]，有两个选择，一种是让第 i 个字符独立，即此时的权值 dp[i]=dp[i−1]，因为第 i 个字符没有创造价值。

另一种选择是让第 i 个字符有价值，即第 i−1 个字符和第 i 个字符组合创造价值，此时 dp[i]=dp[i−2]+map.get(s.substring(i−1,i+1)). 针对这两个选择，取权值最大的即可。

因此，dp[n−1] 为最终答案。

时间复杂度为 O(n)

空间复杂度为 O(n)

看完之后是不是有了想法了呢，题目直达链接：查看题目：字符配对

算法笔试模拟题精解之“数组染色”

简介：可以采用链表的思想，定义一个数组 temp 来存放每个递增的子串，题目需要求出最少的递增子串有多少个，采取的思路是递增的子串越密集越好。

题目描述

等级：中等

知识点：DP、LIS

查看题目：数组染色

codancer 现在有一个长度为 n 的数组 a, 对于每个这个数组的每个数字，我们必须染上颜色，但是 codancer 有一个限制条件 : 如果 i 输入一个正整数 n 和数组 a。

（1<=n<=100000,0<=a[i]<=1000000000）.

输出最少需要的颜色种类数。

示例 1

输入：

5

[2,1,4,5,3]

输出：

2

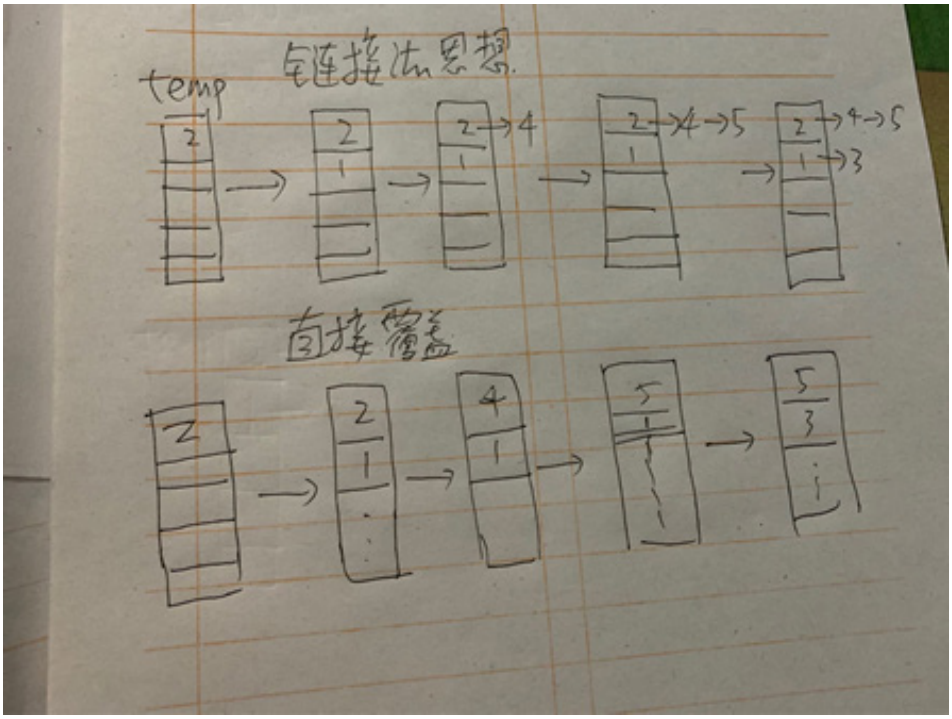
注意

把 a[1],a[3],a[4] 染同一种颜色，把 a[2] 和 a[5] 染成另外一种颜色。

解题思路

大致思路：可以采用链表的思想，定义一个数组 temp 来存放每个递增的子串，题目需要求出最少的递增子串有多少个，采取的思路是递增的子串越密集越好，即若有 2，4，3，5。那么我最优选择的是密集子串 2，3 和 4，5. 而不是 2，4 和 3，5；即使对这个测试用例来说，答案都是一样的，但是就题目而言，密集子串是最优的选择。

具体思路：定义一个数组 temp 来存放每个递增的子串，那么我先将第一个元素放在 temp 数组的第一个位置，从 i=1 开始遍历 x 数组，将 x[i] 与 temp 数组的第一个元素开始比较（j=0），如果 x[i]>temp[j], 那么我就将 x[i] 元素链接在 temp[j] 元素后面，又因为这是个数组，不是链表，而且链接后 temp[j] 值再也用不上，和链表的最后一个元素比较，因此可以直接覆盖 temp[j] 元素，即 temp[j]=x[i]。我说的链接如下图所示



可以发现，**每次比较都是和链表最后一个元素比较**，所以前面的元素可以覆盖，如下面的直接覆盖法，也是用数组实现，很简单。而且还可以发现，**temp 数组永远是递减的，这样也满足密集子串，**因为从上往下走，找到满足的即可停止，不需要继续往下。最终，temp 数组有几个元素，答案就是多少。可以用 ans 一致指向 temp 数组的最后一个元素的位置。

完全理解本解法需要理解我定义的**密集子串和链接法转直接覆盖**等概念。好好理解一下，很简单，有什么问题可以在评论区交流。

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：数组染色

算法笔试模拟题精解之“连绵的群山”

简介：可以将山化分为几个小连续区间，每个区间保证越来越高，并且保证每个区间尽可能的长。除第一个和最后一个区间，中间的其余区间，有移除可能的是每个区间的最小值和最大值，第一个区间有移除可能的是最小值，最后一个区间有移除可能的是最大值。这样才能找出最长的山的区间。

题目描述

等级：中等

知识点：DP

查看题目：连绵的群山

小森家的北面有一条连绵的山脉，山脉高低起伏。小森很好奇这些山中最多有几座连续的山头是越来越高的，当然这对小森来说太简单了。

他又想，假如可以移除其中一座，这个答案最大又会是多少？当然了，他也可以选择不移除任何一座山峰。1 <= n <= 100000, 1 <= a[i] <= 1000000000。

输入内容为两行，第一行为山峰的数量 n，第二行为 n 个数字，表示每个山头的高度。

输出一个数字，表示最大值。

示例 1

输入：

5
[5,10,6,7,8]

输出：

4

注意

可以去掉 10，形成数组 [5, 6, 7, 8]，长度为 3 。

解题思路：

大致思路：可以将山化分为几个小连续区间，每个区间保证越来越高，并且保证每个区间尽可能的长。除第一个和最后一个区间，中间的其余区间，有移除可能的是每个区间的最小值和最大值，第一个区间有移除可能的是最小值，最后一个区间有移除可能的是最大值。这样才能找出最长的山的区间。

具体过程：初始化两个数组，oneIndex 和 arr，arr 保存当前增区间的长度，oneIndex 保存 arr 中数组元素为 1 的索引位置，对于 [5,10,6,7,8] 例子，arr={1,2,1,2,3}（5,10 递增，6，7，8 为一个增区间），oneIndex={0, 2}（因为 arr[0]=1,arr[2]=1）。这下明白这两个数组的含义了吧。

接下来遍历 oneIndex 区间，也就是找出 arr 中所有为 1 的值的索引（arr 数组的存在可以不用遍历 arr 找 1 的位置，而是可以直接定位，用空间换时间）

所有为 1 的值的索引的元素（除第一个 1）和其前面的那个元素都有移除的嫌疑，移除后计算长度。

先移除 1 前面那座山，也就是上一个区间最高的山

```
if(a[oneIndex[i]-2]<=a[oneIndex[i]])
ans=Math.max(arr[oneIndex[i+1]-1]+arr[oneIndex[i]-2],ans);
```

Else ans=Math.max(arr[oneIndex[i]−1],ans);

再移除 1 这座山，也就是当前区间最低的山（有同学可以不明白为什么最低的山

也有移除的可能性，看看下面的例子）

可能出现这种情况

```
2，4，5，6，3，8，9，11，13，15
```

这样的话，即使 3 是第二个区间中最小的，也应该移除，因为这个区间的其他数字都很大，而且可以和前面的区间对接上。

```
if(oneIndex[i]+1<n) {
  if(a[oneIndex[i]-1]<=a[oneIndex[i]+1])
    ans=Math.max(arr[oneIndex[i+1]-1]+arr[oneIndex[i]-1]-1,ans);
  Else ans=Math.max(arr[oneIndex[i]-1],ans);
 }
```

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：连绵的群山

算法笔试模拟题精解之“难住 Tom 的问题”

简介：拿样例来说 2 3 100, 说明有 3 个桶，第 0 个桶一定是空的，那么符合条件的有 12 种方案，我们需要从中推出转移方程。

题目描述

等级：困难

知识点：DP

查看题目：难住 Tom 的问题

Jerry 和 Tom 在公园里感到很无聊，于是就开始研究起来题目了，Jerry 有一个 n+1 个桶 (1<=n<=300)，这些桶的编号为 0,1,2,...,n，然后让 Tom 去构造这些桶 (每个桶可以看作 list)，必须要满足以下的三个要求：

1. 对于第 i 个桶里，它里面有 i 个数 ai，并且这些数不能大于 x(1<=ai<=300)。
2. 对于第 i−1 个桶，它必须是第 i 个桶的子序列。
3. 对于第 i 个桶，它的字典序要大于第 i−1 个桶。

问满足以上要求的方案有多少总，答案对 y(2<=y<=1e9) 取模。

输入三行数据，分别为 n,x,y，意思同题意。

输出一个数，表示最终的方案数，答案对 y 取模。

示例 1

输入：

2

3

100

输出：

12

解题方法：

拿样例来说 2 3 100, 说明有 3 个桶，第 0 个桶一定是空的，那么符合条件的有以下 12 种方案：

({0},{1},{1,1}),
({0},{1},{1,2}),
({0},{1},{1,3}),
({0},{1},{2,1}),
({0},{1},{3,1}),
({0},{2},{2,1}),
({0},{2},{2,2}),
({0},{2},{3,1}),
({0},{3},{3,1}),
({0},{3},{3,2}),
({0},{3},{3,3}),

这只是简单的列举出了所有的情况，我们需要从中推出转移方程。

以 {0},{2} 为例，要构造出第三个桶就相当于往第二个桶中插入一个数字；

那么如果这个数字插在 2 的左边，一定是要比 2 大的数才可以，如果插到右边对于这个情况来说没有限制条件，如果对于位数较多的数列来说也要考虑到字典序的因素。

所以我们考虑 fi[p] 中，i 表示当前的长度，j 表示取到的数字，p 表示有 p 个位置可插。

一共有三种状态，当你的 p 没位置了就没有地方可插，那么这个数就不插了 dpi[i] += dpi[p], 这里的第三维为 i 是因为所有的数都比 j+1 小，所以插到哪里都是可以的，所以为 i。

如果 p 不为 0 的话说明有地方可插，这时可以考虑插在这个地方或者不插这个地方，如果不插这个地方 dpi[p−1]+=dpi[p]，如果插这个地方 dpi+1[p]+=dpi[p]，这里还需要乘以 p+1 是因为还有 p 个位置。

$$\begin{cases} dp[i][j][p-1] \mathrel{+}= dp[i][j][p] & p \neq 0 \\ dp[i][j+1][i] \mathrel{+}= dp[i][j][p] & p == 0 \\ dp[i+1][j][p] \mathrel{+}= dp[i][j][p] * (p+1) \end{cases}$$

时间复杂度：O(n^3)

空间复杂度：n^3

看完之后是不是有了答题思路了呢，快来练练手吧：难住 Tom 的问题

算法笔试模拟题精解之“变化的字符”

简介：把所有数据处理一遍再求一遍最大值即可。

题目描述

等级：困难

知识点：DP

查看题目：变化的字符

Tom 又碰到了一道字符串的题目。

有一个字符串 s(1<=|s|<=3e5，|s| 为奇数)，这个字符串只包含 0,1 和字符 '.'，这个 '.' 字符可以任意变为 0 或者 1。

现在可以通过一些操作来缩短这个字符串，每次操作可以任意选择连续的三个字符，然后将这个连续的三个字符变成出现数量最多的那个字符 (比如 001 变为 0，101 变为 1，1.0 可以变为 0 也可以变为 1)。

通过更改字符 '.'，问通过 (|s|−1)/2 次操作后最终这个字符串只剩下一个 1 的方案有多少种，答案对 1e9+7 取模。

输入一行字符串 s

输出一个数表示方案数。

示例 1

输入：

"1.0.1"

输出：

4

解题方法：

操作次数为$\frac{\text{字符串长度}-1}{2}$，1.0.1 长度为 5 就是可以操作 2 次。

分别可以把字符串变为 10001 10011 11001 11011，这四种都可以使得最后只剩下 1，所以答案为 4 种。

因为最后是剩下 1 所以我们肯定优先消去 0，dp i,j,k 的含义就是前 i 个字符中把第 j 个字符变成 k 的方案数。

把所有数据处理一遍再求一遍最大值即可。

$$dp[i][j][k]\begin{cases} dp[i-1][j][k]+1, & if\ s[i]==\text{'.'} \\ dp[i-1][j][k], & if\ s[i]!=\text{'.'} \end{cases}$$

时间复杂度：O(24n)

空间复杂度：12n

看完之后是不是有了答题思路了呢，快来练练手吧：变化的字符

算法笔试模拟题精解之“跳房子”

简介：这是一个动态规划的问题。设 f(i) 为到第 i 个位置处的最小步数。初始化 f 的每个位置步数都是一个足够大的值。

题目描述

题目等级：中等

知识点：DP

查看题目：跳房子

给出一个长度为 n 的数组 a ，数组中的值 a[i] 表示你在第 i 个位置最多能够移动到第 i + a[i] 个位置，并且不能超过 n。你可以选择移动到的位置包括：i, i+1, ... i+a[i]。你需要确定移动到第 n 个位置的最小移动次数。如果无法移动到第 n 个位置请输出 −1。

输入数组的长度 n（1<=n<=100000），和含有 n 个数字的数组 a，a[i] 表示第 i 个位置最多能够移动到第 i + a[i] 个位置（0 <= a[i] <= 100）。

输出一个数字，表示最小的移动次数。

示例 1

输入：

5
[2, 3, 1, 4, 5]

输出：

2

注意

最优路径为：1->2->5

解题思路：动态规划

这是一个动态规划的问题。

设 f(i) 为到第 i 个位置处的最小步数。初始化 f 的每个位置步数都是一个足够大的值。

Nums(i) 表示第 i 个位置最多能够移动的距离。

对于一个位置 j 来说，它的最小步数应该是前面所有可以一步到达它这里的位置中最小的 f(i)+1。

设 f(1) = 0。后面的状态转移是 j = 1−nums(i)，f(i+j) = min(f(i)+1，f(i+j))

空间复杂度 O(n)

时间复杂度 O(n^2) 每个位置都可以直接走到终点的时候。(但是可以判断是否到达终点提前结束)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：跳房子

算法笔试模拟题精解之“寒假活动”

简介：本题充分利用四个开门状态，就可以使用动态规划。

题目描述

等级：中等

知识点：DP

查看题目：寒假活动

小森马上就要迎来自己长达 n 天的寒假了，为不让自己无聊，他决定寒假每天都尽量出去运动。小森最喜欢的运动是滑雪和游泳。

但是并不是每天滑雪馆和游泳馆都开门，我们定义 a[i] 表示第 i 天的开门状态：

1. a[i] = 0, 滑雪馆和游泳馆都不开门。
2. a[i] = 1, 滑雪馆开门，但是游泳馆不开门。
3. a[i] = 2, 滑雪馆不开门，但是游泳馆开门。
4. a[i] = 3, 滑雪馆和游泳馆都开门。

但是小森又是一个讨厌重复的人，因此他不会连着两天做同样的运动，但是可以连续两天都不运动。

也就是说，只有当滑雪馆开门并且前一天他没有去滑雪的时候他才能去滑雪。游泳同理。因为运动是非常累的，所以小森每天最多只能做一种运动。

现在小森已经得到了寒假时候滑雪馆和游泳馆的开门安排，即数组 a，他现在想知道自己不运动的天数的最小值。

输入假期天数 n1 <= n <= 100000，和包含 n 个数字的数组 a，表示场馆开门安排。

输出一个数字，表示不运动的最小值天数。

示例 1

输入：

5
[3,3,1,2,0]

输出：

1

注意：

小森的最优策略是第一天滑雪，第二天游泳，第三天滑雪，第四天游泳，第五天不运动。

题解思路：动态规划

本题充分利用四个开门状态，就可以使用动态规划：

小森只有 3 种可能的状态 (不运动、滑雪、游泳)。本题要求不运动的最小值天数，可以反向思维，求运动的最大天数，然后用总天数减去最大运动天数即为最小运动天数

dpi 表示第 i 天，选择休息 (j=0)、游泳 (j=1)、滑雪 (j=2) 时的最大运动天数

状态转移方程为

dpi = max(dpi−1, dpi−1, dpi−1)

如果游泳馆开门

dpi = max(dpi−1+1, dpi−1+1)

否则

dpi = max(dpi−1, dpi−1, dpi−1)

如果滑雪馆开门

dpi = max(dpi−1+1, dpi−1+1)

否则

dpi = max(dpi−1, dpi−1, dpi−1)

遍历天数，完成上面状态转移，就完成了动态规划的过程。

第一天的起始值也需要判断开门情况

这里需要指出 dpi 只是表示第 i 天滑雪时的最大运动，而不是第 i 天一定会滑雪

时间复杂度：O(n)

空间复杂度：O(kn)

是不是有思路了呢，点击链接立刻答题：>> 寒假活动

算法笔试模拟题精解之“最短路”

简介：做这道题的思路是，假设一开始没有道路，只有 n 个水库。建立一个连通图，初始状态连通图中只有水库 1 一个结点。然后将逐步将道路加入连通图中，道路加入连通图的条件是道路两端的水库至少有一个在连通图中，道路加入连通图后，道路两端的水库都加入连通图。

题目描述

题目等级：容易

知识点：最短路、DP

查看题目：最短路

《缺氧》是 Klei Entertainment 所制作并发行的一款模拟游戏。这是一个太空殖民地模拟游戏，玩家需要管理你的复制人，帮助他们挖掘、建立和维护一个地下的小行星基地。你需要水、食物、氧气、适当的调节压力和适宜的温度来维持他们活着并满足他们。

在这里，氧气是必不可少的，没有氧气，复制人就会无法生存。电解制氧是一个非常实用的制氧方法，将水通过电解器之后，电解器会消耗水产生氧气和氢气，氧气可以供小人呼吸，氢气则可以进行氢气发电。

复制人在进行了一段时间的挖掘之后，在地图里发现了 n 个水库，他们的命名方法非常暴力，水库 1，水库 2，...，水库 n。他们挖掘了 m 条道路将这 n 个水库联通，现在复制人想知道水库 1 到水库 n 的最短距离。

输入水库数 n、挖掘的道路条数 m (1<= n,m <= 1000) 和一个二维数组 a，其中 a[i]=x,y,z 表示水库 x 与水库 y 之间的道路长度为 z。

输出水库 1 到水库 n 之间的最短距离。

示例 1

输入：

5

7

[[1,2 ,3],[1 ,2 ,1],[1 ,5 ,10],[2 ,3 ,4],[2, 4, 10],[2, 5 ,8],[3 ,3 ,9]]

输出：

9

解题思路：DP

本题需要用到动态规划算法。

首先可以创建一个数组记录水库 1 到其他各个水库的最短距离。做这道题的思路是，假设一开始没有道路，只有 n 个水库。建立一个连通图，初始状态连通图中只有水库 1 一个结点。然后将逐步将道路加入连通图中，道路加入连通图的条件是道路两端的水库至少有一个在连通图中，道路加入连通图后，道路两端的水库都加入连通图。

每当有一条道路加入连通图时，计算水库 1 到此道路两端的水库的最短距离是否可以因为这条道路的加入而更短。如果可以，更新水库 1 到这些水库的距离。

当所有道路加入连通图后，数组中记录的水库 1 到水库 n 的距离即为所求最短距离。

时间复杂度：O(n)

空间复杂度：O(n)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：最短路

算法笔试模拟题精解之“codancer 上楼”

简介：这是一个动态规划问题。对于每层需要保存两个值。一个是这层选择选择走楼梯的最小花费，记为 Ta(i)。另一个是这层选择坐电梯的最小花费，记为 Tb(i)。

题目描述

题目等级：中等

知识点：DP

查看题目：codancer 上楼

codancer 来到了一栋大楼前，现在他要上楼。

如果 codancer 从第 x 层走楼梯到第 y 层 (y>x)，那么他所花费的时间是 a[x]+a[x+1]+…+a[y]；

如果他从 x 层坐电梯到第 y 层，那么他所花费的时间是 c+(b[x]+b[x+1]+…+b[y])，因为他等电梯的时间为 c。

现在 codancer 想知道 **从第 1 层到第 n 层需要最少需要多长时间?**

有四个入参，第一个输入一个正整数 n，表示要上到第 n 层楼；

第二个输入一个整数 c(1<=n<=100000,1<=c<=1000)，表示等电梯花费的时间；

接下来输入两个数组 a 和 b，数组中 n-1 个数字代表数组 a 和 b(1<=a[i],b[i]<=1000)，分别表示爬楼梯和乘电梯每层楼花费的时间。

输出 codancer 到达第 n 层最少需要的时间。

示例 1

输入：

4

1

[3,2,3]

[1,2,3]

输出：

7

注意

直接坐电梯从 1 楼到 4 楼即可，答案是 1+1+2+3=7

解题思路：动态规划

对于每层需要保存两个值。一个是这层选择选择走楼梯的最小花费，记为 Ta(i)。另一个是这层选择坐电梯的最小花费，记为 Tb(i)。

状态转移（字母与题干中所给含义相同）

```
Ta(i+1) = min(Ta(i)+a(i+1), Tb(i)+a(i+1))
Tb(i+1) = min(Ta(i)+b(i+1)+c, Tb(i)+b(i+1))
```

这样一直计算到最后一层即可。

时间复杂度 O(n)

空间复杂度 O(2*n)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：codancer 上楼

算法笔试模拟题精解之“木棒拼接”

简介：考虑使用动态规划算法，令 dp[i][j] 表示在第 i 中状态中最后一根木棒是第 j 根木棒的方案数。

题目描述

等级：困难

知识点：二进制枚举、DP

查看题目：木棒拼接

Codancer 现在由 n 根木棒，第 i 根木棒的长度为 l[i]，并且每根木棒只有红、黄、蓝三种颜色。

现在 codancer 想得到一根长度为 L 的木棍，codancer 可以选择其中的一些按照一定的顺序把这若干根木棒拼接起来，但是 codancer 要求相邻的两根木棒的颜色不得相同并且木棒的长度总和必须是 L。

现在 codancer 想知道有多少种拼接方案（只要是存在顺序不同或者木棒不同就不是一同种方案）。

由于答案比较大，因此你只需要输出答案对 1000000000+7 取余的结果即可。

输入 n 和 T，代表木棒的总数和所需要构造的木棍的长度 T。

接下来 n 行，每行两个数组 l[i] 和 c[i]，代表第 i 根木棒的长度和颜色。(1<=n<=15,1<=L<=225,1<=l[i]<=15,1<=c[i]<=3)

输出方案数对 10^9+7 取余的结果。

示例 1

输入：

3

3

[[1,1],[1,2],[1,3]]

输出：

6

注意

所有的排列方案为：

[1,2,3]

[1,3,2]

[2,1,3]

[2,3,1]

[3,1,2]

[3,2,1]

解题思路：动态规划

根据样例所有的排列方案为：

[1,2,3]

[1,3,2]

[2,1,3]

[2,3,1]

[3,1,2]

[3,2,1]

考虑使用**动态规划**算法，令 dpi 表示在第 i 中状态中最后一根木棒是第 j 根木棒的方案数：

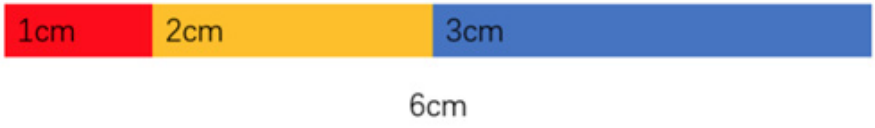
那么可以得到转移方程为：

```
dp[i|(1<<k)][c[k]]=(dp[i|(1<<k)][c[k]]%mod+dp[i][2]%mod+dp[i]
[3]%mod)%mod;(c[k]==1)
```

```
dp[i|(1<<k)][c[k]]=(dp[i|(1<<k)][c[k]]%mod+dp[i][1]%mod+dp[i]
[3]%mod)%mod;(c[k]==2)
```

```
dp[i|(1<<k)][c[k]]=(dp[i|(1<<k)][c[k]]%mod+dp[i][1]%mod+dp[i]
[2]%mod)%mod;(c[k]==3)
```

最后枚举所有的可行性状态计数即可，最终时间复杂度为 n*(2^n)



看完之后是不是有了答题思路了呢，快来练练手吧：木棒拼接

算法笔试模拟题精解之“Codancer 的数组封印”

简介：我们逆向思考，对于给定的数组每次删去一个数，相邻两次操作的答案不会超过 1。

题目描述

等级：困难

知识点：DP、LIS

查看题目：Codancer 的数组封印

Tom 有一个长度为 n 的排列数组 a，即 a 中的每个数字的范围都在 [1,n] 中并且每个数字都不重复。但是现在 Codancer 把整个数组给封印了！

现在 Codancer 给了 Tom 一个解封序列 b, 即 bi 代表第 i 次解封 a[b[i]]。

接下来 Tom 每次都会解封一个位置的数字，令 L[i] 代表第 i 次解封后所有被解封的数字构成的数组的 LIS 的长度，现在 Codancer 想让 Tom 计算 L[1]+L[2]+L[3]+…+L[n] 的值是多少?

第一行是一个正整数 n，代表 a 数组和 b 数组的长度，接下来第一行输入数组 a，第二行输入数组 b。(1<=n<=50000)

输出 L[1]+L[2]+…+L[n] 的值。

示例 1

输入：

4

[4,2,1,3]

[1,2,4,3]

输出：

6

注意

L[1]=1 解封后为 {4}

L[2]=1 解封后为 {4,2}

L[3]=2 解封后为 {4,2,3}

L[4]=2 解封后为 {4,2,1,3}

解题方法：

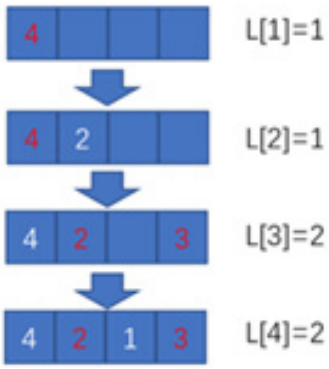
第一步解封后数组为 {4}，因此 L[1]=1

第二步解封后数组为 {4,2}，因此 L[2]=1

第三步解封后数组为 {4,2,3}，因此 L[3]=2

第四步解封后数组为 {4,2,1,3}，因此 L[4]=2

因此 L[1]+L[2]+L[3]+L[4]=6



红色数字构成的序列即为 LIS。

我们逆向思考，对于给定的数组每次删去一个数，相邻两次操作的答案不会超过 1，

对于第 i 次操作我们随便求一个 LIS, 如果第 i+1 次操作删除的数字在这个 LIS 内，

我们就要重新求 LIS, 否则答案保持不变，**由于随机生成的全排列 LIS 长度的期望为 sqrt(n)**，因此最多计算 sqrt(n) 次 LIS。

时间复杂度：O(nsqrt(n)*log(n))。

看完之后是不是有了答题思路了呢，快来练练手吧：Codancer 的数组封印

算法笔试模拟题精解之“Jerry 的异或运算”

简介：本题可以通过推导出一个公式来解答。

题目描述

等级：困难

知识点：DP、数学

查看题目：Jerry 的异或运算

Jerry 最近在研究异或运算，异或也叫半加运算，其运算法则相当于不带进位的二进制加法。

Jerry 给你一个数字 N，他想知道在 0 到 N 之间有多少对数字 (a,b) 满足 u+v=a, u ⊕ v = b, ⊕ 表示按位异或，由于答案将会非常大，请将结果对 1e9+7 取模。

输入一个整数 N (0<=N<1e18)

输出整数对 (a,b) 的数目，结果模 1e9+7

示例 1

输入：

3

输出：

5

解题方法：

由于 u+v=a,u ⊕ v = b, 根据异或的性质我们可以知道：u+v = ((u&v)<<1) + (u ⊕ v)，所以 u+v >=u ⊕ v，那么只要 u+v<=n, u ⊕ v 也必定小于等于 n。于是这道题就转化成了对于 u+v<=n , 求 (u,v) 的对数。

定义 dp[n] 表示 u+v<=n ,(u,v) 的对数，当 n=0 的时候 dp[0] = 1，n=1 时，dp[1] = 2。他们的 (u,v) 分别是 {(0,0)} 和 {(0,0),(1,0)}。

dp[n] 再往下递推的时候，要考虑 u+v<=n 时 二进制是如何由小到大转化过来的。

从二进制上来看 u 和 v，两者的末位只有 0 或 1，那么二者的末位和有 0,1,2 三种情况。

设 u+v=x，令 u = u1*2+u0，v = v1 *2+v0，u0+v0 即为二者的末位和，又有 x<=n，故 (u1+v1)*2+u0+v0<=n, 所以 u1+u2<=(n−u0−v0)/2。

由 u1+u2<=(n−u0−v0)/2 到 (u1+v1)*2+u0+v0<=n 可以推出来递推式：dp[n] = dp[n/2]+dp[(n−1)/2]+dp[(n−2)/2]。

看完之后是不是有了答题思路了呢，快来练练手吧：Jerry 的异或运算

算法笔试模拟题精解之“小明的数学作业”

简介：关键在于对题目的了解，数学老师让小明求的是 n 个数中最长的等差数列长度，可以用 dp[i][j] 表示以 a[j] 和 a[i] 为结尾的等差序列的最长长度。

题目描述

等级：容易

知识点：DP、尺取法

查看题目：小明的数学作业

众所周知，小明是一个数学小能手，有一天数学老师给了小明一个长度为 n（2<=n<=5000）的序列，其中第 i 个数是 ai（0<=ai<=1e9），数学老师想知道这个序列排序后，其中最长的等差子序列的长度是多长，聪明的你能帮小明解决这个问题吗?

其中等差子序列的定义为：一个长度为 length 的等差序列 b1、b2……blength，并且序列 b 是序列 a 排序后的子序列。

请注意子序列的定义：在原来序列中找出任意数量，任意位置的元素，在不调换这些选出的元素前后顺序的情况下，组成的新序列，称为原序列的子序列。

第一行为序列的长度 n（2<=n<=5000），接下来一行是 n 个数，其中第 i 个数是 ai（0<=ai<=1e9）。

输出一行，最长的等差子序列 b 的长度 length。

示例 1

输入：

6

[0,1,3,5,6,9]

输出：

4

解题思路：

首先来理解一下题目：数学老师让小明求的是 n 个数中最长的等差数列长度，可以用 dpi 表示以 a[j] 和 a[i] 为结尾的等差序列的最长长度。

因为我们肯定要保存一个公差作为状态量，但是直接枚举 0-1e9 又不现实。

所以我们巧妙的设计出了这个状态，使得我们的公差就被 a[i]-a[j] 表示了。

因此我们的转移就应该是在三个数中转移。 即 a[i],a[j],a[k]，i 根据等差数列的性质，若 a[i],a[j],a[k] 构成等差序列，那么 a[i]+a[k]=2*a[j]; 每次转移更新答案即可。

是不是有思路了呢，点击链接立刻答题：>> 查看题目：小明的数学作业

算法笔试模拟题精解之“奇偶数列”

简介：对于这道题来说，只需要考虑是奇数还是偶数，而不用理会具体的值。

题目描述

等级：中等

知识点：DP

查看题目：奇偶数列

Tom 有一个长度为 n(1<=n<=100) 的数列，数列中只包含 1−n 且不重复的数字。

这个数列是一个不完整的数列，意思是说数列中有几个位置是空的，Tom 想让你帮他把空的位置填上数字，当然是有要求的，对于一个数列来说，如果相邻两位的数字奇偶性不同，这个数列的权值就 +1，请你算出来将空位补上以后，权值最小为多少？对于这个数列，空位用 0 表示。

输入数列长度 n(1<=n<=100) 和 n 个数 ai(0<=ai<=n)

输出填上空位以后的数列的最小权值。

示例 1

输入：

5
[0,5,0,2,3]

输出：

2

注意：

对于样例解释说明，符合题意的数列可以为 1 5 4 2 3

解题思路一：贪心算法

对于这道题来说，只需要考虑是奇数还是偶数，而不用理会具体的值。

下面用 0 代表偶数，1 代表奇数，-1 表示空位。

这道题对应的空位只有下面 5 种情况（连续空位的不同个数没有影响）：

1. 空位在左右两侧（以左侧举例）-10 也就是最边上是连续的空位，然后接一个偶数；
2. 空位在左右两侧（以左侧举例）-11 也就是最边上是连续的空位，然后接一个奇数；
3. 空位在中间，两侧都是奇数 1-11；
4. 空位在中间，两侧都是偶数 0-10；
5. 空位在中间，两侧一奇一偶 0-11。

计算在 5 种情况下对应最少可能增加的权值

1. 可能增加 0(填偶数) 或 1(偶数不够，要填奇数)
2. 可能增加 0 或 1 与上面对应
3. 可能 0(填奇数) 或 2(奇数不够填偶数)
4. 可能 0 或 2 与上面对应
5. 只能为 1

给 5 种空位重要性排序（确定贪心策略）

第 5 种，因为不管填什么都增加 1，所以排在最后

第 3，4 种，因为填错了会增加 2，比一二种多，所以排第一（相同情况按照空

位个数排序，越少越靠前）

第 1，2 种，排在 3，4 后面。

计算过程

1. 遍历一次，从原数组中提取出这 5 种空位，同时计算没有填入的奇数偶数分别为多少个
2. 按照 (3，4)(1，2)(5) 分成三组，对前两组按照空位从少到多排序
3. 按照贪心策略的顺序

先判断 (3，4) 中有多少可以增加 0 的，并消耗掉对应的奇数偶数个数，增加为 2 的直接跳过，不消耗个数。

然后判断 (1，2) 中有多少可以增加 0 的，并消耗掉对应的奇数偶数个数，增加为 1 的直接跳过，不消耗个数。

最后计算 (3，4) 剩下的个数 `*`2+(1，2) 中剩下的个数 `*`1+(5) 中剩下的个数 `*`1

解题思路二：动态规划

样例是 0 5 0 2 3，那么两个空位只能填 1 或者 4，所以有 1 5 4 2 3 和 4 5 1 2 3 两种情况。

由于前者的权值要小于后者的权值，所以 1 5 4 2 3 是最优解，如果我们考虑贪心的做法会有很多的情况要讨论，而且也避免不了一些冲突。

我们可以将问题抽象一下，其实就是求对于当前这一个位置之前有多少个奇偶对，可以进一步转换成求前一位的奇偶性，不妨考虑**动态规划**。

我们设 dpi[k]，其中 i 表示当前所在第 i 位，j 表示前 i 位一共有 j 个奇数 (那么就有 i−j 个偶数)，k 只能为 0 或 1，当 k 为 0 表示第 i 位放 0，k 为 1 表示第 i 位放 1，然后我们同时更新第 i 位放 0 和放 1 的情况的最小值。

那么对于当前的位来说如果填偶数就有 dpi[0] = min(dpi−1[0], dpi−1[1] + 1)，如果填奇数就有 dpi[1] = min(dpi−1[0] + 1, dpi−1[1])，最终的状态就是 min(dpn[1], dpn[0])。

$$\begin{cases} dp[i][j][0] = \min(dp[i-1][j-1][0], dp[i-1][j-1][1] + 1) & a[i] != \text{奇数 and } j > 0 \\ dp[i][j][1] = \min(dp[i-1][j][0] + 1, dp[i-1][j][1]) & a[i] != \text{偶数} \end{cases}$$

其中$i \in [1,n]$, $j \in [0, i]$

时间复杂度：O(2n^2)

空间复杂度：2n^2+n

看完之后是不是有了答题思路了呢，快来练练手吧：奇偶数列

1.4 剪枝

算法笔试模拟题精解之“斐波那契字符数”

简介：本题应充分利用斐波那契数列的性质，自顶向下对问题逐步剪枝，定位需要判断的数字位置。

题目描述

题目等级：中等

知识点：递归、剪枝

查看题目：斐波那契字符数

Tom 发现了一种神奇的字符串 – 斐波那契字符串 , 定义 f[1]=0,f[2]=1, 对于所有的 i>2 都有 f[i]=f[i−2]+f[i−1]，其中“+”代表拼接，比如 01+10=0110，现在对于字符串 f[n]，请判断 f[n] 的第 k 项是 0，还是 1。

输入两个数字 n 和 k(n<=300,k<=1000000000)。

输出 f[n] 的第 k 位，如果 k 大于 f[n] 的位数，则输出 −1。

本题在答题过程中很多小伙伴反应会出现超内存的现象，导致 JVM 崩溃。但其实这道题目是不可以暴力解题的，首先来看一下一般思路解法。

方法 1：直接法（超时并超出内存限制）

这是一道找规律的题目，最简单的想法可以是自底向上地构建出从第一个字符串 f[1] 直到最后的 f[n]，然后查看 f[n] 的第 k 项是 0 还是 1。

由于斐波那契数递增速度非常快，当 n 较大时，这种暴力拼接法无法完成。

下面介绍一种优化的算法。

方法 2：找规律（提交通过）

本题应充分利用斐波那契数列的性质，自顶向下对问题逐步剪枝，定位需要判断的数字位置。

比如，对输入 n=19, k=98：

1. 先算出 n=19 对应的字符串长度 len，也就是斐波那契数列的第 19 项。考虑到 k 是 int 类型，故可以先算出 n=1~47 对应的斐波那契数组成的数组（n=47 时，斐波那契数大于 2^31−1，无需再提前计算更多的斐波那契数），然后直接在计算好的斐波那契数组中取出第 19 项。
2. 找到 n−2=17，n−1=18 对应的斐波那契字符串长度（也就是第 17/18 个斐波那契数）len1，len2。同样也是在提前计算好的斐波那契数组中找到。
3. 对于 k=98，如果 k>len1，说明要找的数字在 n=18 对应的斐波那契字符串中，也就是 n=19 对应的斐波那契字符串的后半部分；如果 k<=len1，说明要找的数字在 n=17 对应的斐波那契字符串中，也就是 n=19 对应的斐波那契字符串的前半部分。
4. 根据判断结果，将 n 赋值为 n−2 或 n−1，同时将 k 赋值为 k 或 (k−len1)，完成剪枝。回到第 1 步，递归向下搜索。直到 n=1 或者 2，这时 k 也变为 1，定位完成，结束递归，直接返回结果。

进一步优化：

当输入的 n > 47 时，斐波那契字符串的长度超出了 int 型变量 k 的表示范围。需要提前进行推断剪枝。你能根据 k 的 int 表示范围（小于等于 2^31−1）这一特性，对 n 较大的那些用例进行优化么?

方法 3：递归

这个问题可以使用**递归的方法**来求解。

题目中给出了字符串拼接的规律：f[i]=f[i−2]+f[i−1]。可以看出对于第 i 个字符串，它的前半部分属于第 i−2 个字符串，后半部分属于第 i−1 个字符串。

这样，当给定具体的 n 和 k 时，我们可以首先判断 k 位于第 n 个字符串的前半部分还是后半部分，然后递归的把 k 传递给第 n−2 或 n−1 个字符串。

计算过程：

1. 为了判断属于前半部分还是后半部分，需要计算每一个字符串的长度。

一个 len 数组。每个位置为对应字符串的长度。

len[1] = len[2] = 1;

len[i] = len[i−2]+len[i−1];

1. 如果 k > len[n]，则输出 −1。
2. 设置递归函数 int find(int i, int k); 这个函数返回第 i 个字符串中 k 个位置的字符
3. 判断 k 在前半部分还是后半部分。

a) K <= len[i−2] 前半部分，调用 find(i−2, k)

b) K > len[i−2] 后半部分，调用 find(i−1, k−len[i−2])

上面的计算在第一步时会遇到问题，当 n 大于几十的时候，len 就会超出 int 的范围。

对这个问题的解决基于对递归过程的观察，第 4 步 a）中 k 始终是不变的，i 的奇偶性也保持不变。所以在计算第 1 步 len 时可以提前停止，当发现 len[i]>=k 且 i

与 n 的奇偶性相同时。第 4 步中的递归起始也可以使用第 i 个字符串开始，而不是第 n 个。

时间复杂度 O(2*n)

空间复杂度 O(n)

是不是感觉有思路了，立刻点击这里开启答题！

算法题目的解法并不唯一，这里介绍部分解题思路给大家，希望可以给大家启发 ~

1.5 尺取法

算法笔试模拟题精解之“超级区间”

简介：使用 尺取法 对搜索空间进行遍历。设当前区间为 [L, R]。初始 L = R = 0；使用尺取法需要判断什么时候调整 L 和 R。

题目描述

题目等级：中等

知识点：二分查找、尺取法

查看题目：超级区间

Tom 现在有一个长度为 n 的数组，Jerry 给 Tom 定义了一种超级区间，如果区间 [l,r] 满足 (a[l]+…+a[r])>=k, 则区间 [l,r] 被称为超级区间，现在 Jerry 想让 Tom 告诉他数组中有多少个超级区间。

输入整数 n, 整数 k(1<=n,k<=100000)，和一个大小为 n 的数组，数组的每个元素的大小都在 [1,1000] 之间。

输出输入数组的超级区间的个数。

示例 1

输入：

3
5
[2,3,5]

输出：

4

注意

样例满足条件的超级区间为

[1,2],[2,3],[1,3],[3,3]。

解题思路：尺取法

使用 **尺取法** 对搜索空间进行遍历。

设当前区间为 [L, R]。初始 L = R = 0；**使用尺取法需要判断什么时候调整 L 和 R。**

数组中的所有数都为正数。

情况 1：假设对于某个区间 [L1, R1] 满足超级区间的定义。因为所有数都为正数，所以保持 L1 不变，依次增加 R1 直到 n 为止的所有区间都满足超级区间的定义。

情况 2：假设对于某个区间 [L2，R2] 不满足超级区间的定义。则需要保持 L2 不动，增加 R2 才可能满足超级区间的定义。

情况 3：是情况 2 拓展。如果 [L2，n] 不满足超级区间的定义，则任何 i>L2，区间 [i, n] 都不满足超级区间的定义。

计算过程

1. 初始 L = R = 0；sum = 0; 用来计算满足条件的区间个数
2. 判断区间 [L, R] 的情况，满足情况 1，则 sum++; R++。满足情况 2，L++。满足情况 3，结束计算。

对于情况 1，因为 L 不变时，后面的所有 R 都满足条件，所以可以修改为 sum+=n−R+1；L++。

时间复杂度 O(n^2) 修改之前最差为 O(n^2)

空间复杂度 O(1) 记录当前区间数组元素的和

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：超级区间

算法笔试模拟题精解之“调整数组”

简介：首先理解题意：经过 k 次操作后能出现次数最多的元素，以及这个元素的最小值；然后将示例仔细分析一下，就可以找到解题的方法了。

题目描述

等级：中等

知识点：尺取法

查看题目：调整数组

A 同学有一个长度为 n 的数组，对于这个数组，他能够进行 k 次如下操作：任选数组中的某个元素将其数值加一（一个元素可以被加多次）。

他想在操作不超过 k 次的情况下，使得数组中某一元素的出现次数尽可能的多，并同时使这个元素尽可能的小。

请你找出出现次数最多的元素的出现次数以及此时这个元素的最小值。

输入：

输入数组长度 n(1 <= n <= 10^5)，最多能操作次数 k(1 <= k <= 10^5)，和一个包含 n 个数的数组，数组中第 i 个元素的值为 ai(0 <= ai <= 10^4)。

输出：

输出操作后出现次数最多的元素的出现次数以及此时这个元素的最小值。

解题方法：模拟法

变量概述

- int n : 有 n 个数
- int k : 最多操作 k 次
- int[] a : 存储 n 个数的数组
- 只能对 a[i] 进行一种操作 : +1

首先理解题意：经过 k 次操作后能出现次数最多的元素 , 以及这个元素的最小值；

示例说明：

- 1 2 2 3 3 4

两个 3 要变成 4 就得操作 2 次

两个 2 要变成 4 就得操作 4 次

- 因为 ai 的大小被限定在 [0, 10^4], 所以可以使用一个数组 arr, 来统计每个数字出现的次数。
- 遍历数组 arr, 对于任意不等于 0 的数 arr[i], 逆向遍历前面所有 `* 不为 0 的 ***arr[j]**`, 并对每个不为 0 的 arr[j], `**k-=arr[j] * (i-j)**`, 直到 k 不够减或没前面没数字了。
- 如果是 k 不够减 , 则得在判断 k 能够再减去多少个 arr[j], **因为之前的公式是计算减去所有的 arr[j] 的**
- 之所以要不为 0 的 , 是因为压根没有这个数 , 无法对其进行 + 操作

解题步骤

1. 定义变量

- int[] arr : 用来存储每个数字出现的次数 , 容量为 10001
- int max : 用来存储最大次数 , 初始化为 1

- int maxValue：用来存储最大次数所对应的最小值，随着最大次数的更新而更新，初始化为 a[0]
- int k1：用来备份 k 值，避免 k 值计算后不见了
- int temp：用来统计当前最大数字出现的次数，初始化为 arr[i]

2. 遍历数组 arr, 找出出现最多的次数以及对应的最小值 .

- 对于每个 arr[j] 如果 arr[j] 为 0, 则直接跳过 . 不是不加，而是没法加
- 如果 arr[j] * (i−j) <= k, 证明当前的操作次数能够将 **j** 全部变成 **i**, 则使用 **temp** 加上所有的 **j**
- 如果 arr[j] * (i−j) > k, 则证明**当前次数 k 无法将全部 j 变成 i**，开始尝试**将部分 j 转成 i**
  - 使用 k/(i−j) 计算还能够将多少个 j 转成 i，然后进行转换
  - 判断 temp 和最大值是否大于最大次数 max, 如果大于，则更新 max 和 maxValue
- **注意：**当 arr[j] 顺利遍历到 arr[0] 时退出循环，因为我们的比较最大值操作是在循环内进行的，所以此时，可能无法对 max 进行更新，因此需要在手动判断一次 temp 和 max 的大小

3. 返回数组

时间复杂度：O（n^2）

空间复杂度：O（n）

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：调整数组

算法笔试模拟题精解之“最优分组”

简介：一组数字，在不改变顺序的情况下，每相邻两个数字之间的差值是确定的。所以可以对这组数字进行排序后遍历的方法。

题目描述

等级：容易

知识点：尺取法

查看题目：最优分组

给出一个长度为 n 的数组和一个数字 k ，你需要在数组中选出一些数字，这些数字两两之间的差值不能超过 k。你需要判断最多能够挑出的数字个数。(n 在 [1, 100000] 之间，k 和数组中的数字在 [1，1000000000] 之间)

输入数组长度 n；选出的两数字最大差值 k；和一个长度为 n 的数组。

输出最多能够选出的数字个数。

示例 1

输入：

5

5

[13,4,6,9,20]

输出：

3

解题方法：排序后遍历

很多同学在思考这道题的过程中，碰到最大的难点就是，一个数字可能同时属于多个分组，而这个分组的大小，可能是从 2~n 任何一个数字。这样的组合方式，如果用暴力的方法，挨个遍历的话，是非常恐怖的。

比如，如果这道题提供了一个长 100 的数组用例，那么选出一组数可能的组合方式，就有 2^100 ≈ 1024^10 ≈ 10^30 种。

所以我们发现，拖累解题过程最直接的点，在于题目给的数据结构不合理。对一个无序的，可能很长的数组，怎么可能短时间内遍历所有的“取一组数”的可能性呢?

于是，我们可以考虑，如何优化这道题提供的数据。对数组，在不违反题目要求的情况下，最简单的优化方式就是排序。排序的时间复杂度很低，好的排序算法的空间复杂度为常数级，对本题影响很小。所以不妨**假设对于一组有序的数字**，再去考虑这道题，会不会发现简单了不少?

当面对一组数字时，他们中最大的值和最小的值的差值如果不超过 k，那么其中任何两个数字的差值，都一定不超过 k。而有序数组，恰恰可以帮我们快速找到任何一组数字中的**最大的值和最小的值**，我们可以用一对指针，用右指针指向的值，与左指针指向的值做差，从而一次性判断左右指针之间的值组成的一组数，是否满足上面的条件。

当右指针与左指针的差值超过 k 时，我们可以尝试移动左指针，重新满足上面的条件。

当右指针与左指针的差值不超过 k 时，我们可以尝试移动右指针，寻找所求值的最大值。

试着应用上面的思想，最佳方案应该可以达到：最坏情况下两个指针各遍历一次数组，就可以得到正确答案了。

时间复杂度：O(nlogn)

看完是不是有了新思路了呢，快来试一下吧：查看题目：最优分组 。

算法笔试模拟题精解之“破译密码”

简介：可以使用尺取法来解题；设当前区间为 [L, R]。初始 L = R = 0；使用尺取法需要判断什么时候调整 L 和 R。

题目描述

题目等级：中等

知识点：尺取法

查看题目：破译密码

给你一个长度为 n 的序列，元素标号 1−n。问能够找到多少对不同的（L,R）（1 <= L <= R），使得在子序列 [L,R] 内存在出现频率不低于 K 的元素?

输入序列大小 n（1 <= n <= 10^4）、出现频率 k（1 <= k <= n）和一个包含 n 个整数的数组，第 i 个整数表示序列的第 i 个元素为 ai(1 <= ai <= 10^9)。

输出满足条件的子序列个数。

示例 1

输入：

4

2

[1 ,2 ,1 ,2]

输出：

3

注意

三个子序列分别为 [1,3],[1,4],[2,4]

解题思路：尺取法

与 57 超级区间那道题是类似的方法。

设当前区间为 [L, R]。初始 L = R = 0；使用尺取法需要判断什么时候调整 L 和 R。

情况 1： 假设对于某个区间 [L1, R1] 满足题目要求。则保持 L1 不变，依次增加 R1 直到 n 为止的所有区间都满足题目的要求。

情况 2： 假设对于某个区间 [L2，R2] 不满足题目要求。则需要保持 L2 不动，增加 R2 才可能满足满足题目要求。

情况 3： 是情况 2 拓展。如果 [L2，n] 不满足题目要求，则任何 i>L2，区间 [i, n] 都不满足题目要求。

使用一个 2*n 的数组 numb 来记录当前区间 [L, R] 内每个元素的出现次数。numbL 和 numbR 表示区间 [L, R] 对应的 numb 数组范围。因为每个数都是按照进入区间 [L, R] 的顺序离开区间 [L, R] 的，相当于一个队列，所以 numb 数组可以用 numbL 和 numbR 来表示当前区间 [L, R] 内的元素，而不用考虑中间某个数不在区间 [L, R] 内的情况。

计算过程

1. 初始 L = R = 0；sum = 0; 用来计算满足条件的区间个数
2. 判断区间 [L, R] 的情况，满足情况 1，则 sum++; R++。满足情况 2，L++。满足情况 3，结束计算。
3. R 每次加 1，都要给 numb 数组中对应的数出现次数 +1。当是一个新数时，要使 numbR++；

4. L 每次减 1，都要给 numb 数组中对应的数出现次数 −1。当减为 0 时，要使 numbL++

对于情况 1，因为 L 不变时，后面的所有 R 都满足条件，所以可以修改为 sum+=n−R+1；L++。

时间复杂度 O(n^2) 修改之前最差为 O(n^2)

空间复杂度 O(2*n) numb 数组的空间

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：破译密码

二、数据结构

2.1 图

算法笔试模拟题精解之“变换的密钥”

简介：根据题意，只要根据给出的初始关系，计算出对于每个字母可以变成的字母，就可以直接判断 s1 能不能转化成 s2 了。对于求幂操作有可以加快计算的方法，叫做矩阵快速幂。这里用求数的次幂举例。

题目描述

题目等级：中等

知识点：广度优先搜索 /BFS、Floyed 最短路、图

查看题目：变换的密钥

Tom 最开始有一个密钥 s1，s1 是长度为 n 的由小写字母组成的字符串。Jerry 也有一个长度为 n 的由小写字母组成的密钥 s2。

现在有 m 组关系，每组关系由两个数字 [u,v] 构成 (1<=u,v<=26)，表示 26 个字母表中的第 u 个小写字母可以直接转换为第 v 个小写字母。

假设 u=1，v=2, 那么说明字母 'a' 可以直接转换为字母 'b'。现在 Tom 对于 s1 的每个字母使用无数次转换，请判断 s1 能否转换为 s2。

输入内容为四部分，第一部分为 n(1<=n<=100000)，代表字符串的长度。接下

来两部分分别是长度为 n 的 s1,s2。

第四部分为 m（1 <=m <=325），代表关系个数。接下来是 m 组 [u,v] 关系。

如果 s1 能变成 s2, 则输出 YES, 否则输出 NO。

示例 1

输入：

6

"aabbcc"

"cdbcad"

4

[[1,3],[3,1],[1,4],[2,3]]

输出：

"YES"

注意

可以转换多次，比如 a 可以转换为 b，而 b 可以转换为 c，则 a 可以转换为 c。

样例：aabbcc->cabbcc->cdbbcc->cdbccc->cdbcac->cdbcaa->cdbcad

解题思路一：一般思路

根据题意，只要根据给出的初始关系，计算出对于每个字母可以变成的字母，就可以直接判断 s1 能不能转化成 s2 了。

例如样例中计算过后

```
a -> a,c,d
b -> b,c
c -> a,c,d
d -> d
```

对于这个结果，可以使用一个 26*26 的二维数组 change 来保存。每一行表示原始字母，每一列表示可以变换成的字母。1 表示可以变化，0 表示不能变换。

计算过程：

1. 初始化，根据给出的关系填充数组
2. 记得数组对角线要填为 1，表示每个字母可以不进行变换，是自己本身。
3. 计算经过无数次变换后的数组。

对于第三步可以直接进行多次 for 循环，因为数组不大，一般不会超时。

方法二：矩阵快速幂

经过 0-2 次变换后可以达到的字母的结果相当于 change*change，0-3 次对应的相当于 change*change*change，0-n 次对应的就是 change^n。这是一个矩阵求幂的过程。

对于求幂操作有可以加快计算的方法，叫做矩阵快速幂。这里用求数的次幂举例。

假设要求 x^13。直接的方法是进行 12 次相乘。

我们可以观察到 13 = 8 + 4 + 1= 2^3 + 2^2 + 1

所以 x^13 = x^8 + x^4 + x= (x^4)^2 + (x^2)^2 + 0*(x)^2 + x

从右向左，每一项都是前一项的平方。这样原来需要 12 次乘法的操作变成了 3 次惩罚，3 次加法（第二项系数为 0，不用加）。

与求数的次幂类似，矩阵也可以用相同的方法加速计算。对于这道题本身，因为矩阵中非 0 的结果表示可以进行变换，所以相乘时没有必要算出具体的值，只需要判

断结果是否非 0。

因为只有 26 个字母，所以只需要算到 26 次幂就可以了，也可以计算 32 次幂来去掉加法的部分。

时间复杂度 O(5*26*26*26) 5 是求 32 次幂需要的矩阵乘法次数。26*26*26 是一次乘法需要的时间复杂度。

空间复杂度 O(2*26*26) 一个二维数组保存当前结果，一个保存相乘后的结果。

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：变换的密钥

2.2 搜索

算法笔试模拟题精解之“2 的幂次方数”

简介：对于本题，因为要从数组中删除数字。删除分为两步，一是定位（查找），二是删除（将出现次数减 1）。

题目描述

等级：中等

知识点：数学、枚举

查看题目：2 的幂次方数

Tom 是一个十分喜欢数学的人，尤其喜欢 2 的幂次方的数字。现有 n（1<=n<=150000）个数字，对于其中的每一个数字 ai(1<=i 输入数字个数 n 和 n 个数字；

输出一个数字，表示最终需要删除的数字的个数。

示例 1

输入：

3

[1,2,3]

输出：

1

解题方法：位运算

对于本题，因为要从数组中删除数字。删除分为两步，一是定位（查找），二是删除（将出现次数减 1）。所以首先需要考虑的难题是，如何快速定位要查找的数字，并记录下数字出现的次数？ for 循环的线性查找，递归的二分查找（对有序数组），建立哈希表，都是可选的方式。既然追求最佳解法，你不妨先试试将题目提供的数据结构转为哈希表?

之后的第二个难点，就是如何得出“与某个数相加为 2 的幂次方数”的数字了。我们知道，用二进制表示时，一个 2 的幂次方的正整数，譬如 2，4，8，16...，只有最高位为 1，其余位都是 0，譬如 b1，b10，b100，b1000... 所以，对每个数字，只要**用位元算找到它的最高位 1 的位置**，就可以确定比它大的 **2 的幂次方数中最小的数字**了。

本题最后的陷阱，在于正确理解**“与这个数相加为 2 的幂次方数”**的条件。举个例子来说，对于数字 1，它不仅与 1 相加为 2 的幂次方数，与 3，7，15... 相加后，结果都是 2 的幂次方数。很多同学想到位运算的时候，可能忽略了这个条件，而只考虑了**比数字大的 2 的幂次方数中最小的数字**

时间复杂度：O(31n)（考虑到本题 Integer 正整数所占的二进制位数

空间复杂度：O(n)

看完之后是不是有了答题思路了呢，快来练练手吧：2 的幂次方数

算法笔试模拟题精解之“能量半径”

简介：题目的含义就是找到距离原点最近的第 k 个点，并求它的半径。这个题的关键在于排序算法。使用最简单的冒泡排序，时间复杂度 O(n^2)；使用快速排序等好一点的排序算法，时间复杂度 O(nlog(n))；也可以使用 java 中 Arrays 类中的 sort 函数。

题目描述

题目等级：中等

知识点：二分查找、树状数组

查看题目：能量半径

codancer 来到了一个能量平面上的中心，坐标为 (0,0)，接下来巫师 Tom 会在 q 个坐标上放置能量点，每个能量点的能量值为 1，为了打败哥斯拉，他需要至少 k 点的能量，因此他想确定一个最小的整数半径 r 使得 codancer 能够从这个圆心为 (0,0)，半径为 r 的圆形区域内得到至少 k 个能量值，请你帮他确定最小的整数半径 r。

输入包含三个部分，第一个输入能量点的个数 q（1<=q<=100000），第二个输入必须获得的最小能量值 k(0<=k<=q)，第三个是 q 个坐标 (xi,yi)，(−100000<=xi<=100000, −100000<=yi<=100000 并且 xi 和 yi 都是整数)。

输出一个正整数 r，表示最小的半径。

示例 1

输入：

4

3

[[1,1],[−1,1],[−1,−1],[2,3]]

输出：

2

注意

当半径为 2 的时候可以收集到 1,1,[−1,−1] 处的能量。

解题思路：排序算法

题目的含义就是找到距离原点最近的第 k 个点，并求它的半径。

计算过程：

1. 计算每个点到原点的距离，为了减少计算量可以只计算到平方。使用 long[n] 这样的数组来保存。使用 long 类型是为了防止平方之后结果超出 int 类型范围。
2. 对距离进行从小到大排序
3. 取排序后的第 k 个数值开根号，转化为 int 类型并加 1。加 1 是因为开根号后可能为小数，转化成 int 类型会直接舍去小数部分，导致结果比实际小一些。

这个题的关键在于排序算法。

使用最简单的冒泡排序，时间复杂度 O(n^2)；

使用快速排序等好一点的排序算法，时间复杂度 O(nlog(n))；

也可以使用 java 中 Arrays 类中的 sort 函数。

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：能量半径

算法笔试模拟题精解之“苹果收获程序”

简介：因为每次下落时，苹果树每一层的节点都会往下掉一层。由此可以想到，如果苹果树某一层的节点的数目为奇数时，这一层的节点的苹果掉落到第一层时，由于一个节点只能存储一个二进制位的原因，只会剩下一个苹果。而如果苹果树某一层的节点数目为偶数，这一层的节点的苹果掉落到第一层时，剩下的苹果数目为 0。

题目描述

题目等级：容易

知识点：深度优先搜索 /DFS

查看题目：苹果收获程序

Alice 和 Bob 在春天的时候种下了一棵苹果树，为了吃到苹果，他们每天都会去给苹果树浇水。一眨眼到了苹果成熟的时候，但是他们却因为平时照顾苹果树太累了，没有更多的精力去收获苹果。身为程序猿的 Bob 灵机一动，写了一个自动收获苹果的程序。

这个程序把苹果树简化成了一个拥有 n 个节点，根节点为 1 的树，每个节点有 1 个苹果，苹果会在程序的作用下同时往根节点的方向掉落。

但是这个程序有一个致命的 Bug：每当有两个苹果同时掉落到同一个节点的时候，这两个苹果会在 Bug 的作用下消失。

每当 1 苹果落到根节点 1 的时候，Bob 就可以收获 1 个苹果。

Bob 想要预测他们最后可以通过程序收获多少个苹果。

输入内容有两行。

第一行有一个正整数 n(2<=n<=10^5)，表示树有 n 个节点。

第二行有 n−1 个数 p2,p3,...,pn，(1<=pi 滚落到节点 2 上面的苹果会因为 bug 而消失的只剩一个。)

输出一个数字，表示 Bob 最后能收获的苹果数量。

示例 1

输入:

5

[1,2,2,2]

输出:

3

注意

1. 苹果的掉落速度是相等的，即每次都会掉落到与当前节点直接相连的节点上。
2. 只要有苹果出现在根节点 1 上面就会被收获。

解题思路

苹果收获程序在正常情况下，有多个苹果落到同一节点时，应该会是一个相加的情况。结合这个 BUG 的情况，可以猜想，这个程序的 BUG 也许是因为这棵树每个节点只能存储一个二进制位导致的，在这种情况下出现的 BUG 和题目中的相符。

因为每次下落时，苹果树每一层的节点都会往下掉一层。由此可以想到，如果苹果树某一层的节点的数目为奇数时，这一层的节点的苹果掉落到第一层时，由于一个节点只能存储一个二进制位的原因，只会剩下一个苹果。而如果苹果树某一层的节点数目为偶数，这一层的节点的苹果掉落到第一层时，剩下的苹果数目为 0。

因此可以统计每一层有多少个节点来解决这个问题，根节点 1 是第一层，掉落到一层的节点是第二层，以此类推，通过判断一个节点会掉落到的层数来判断该结点当前是第几层。遍历数组，用一个数组 count 记录每一层拥有的节点数，遍历数组，计算出一个节点所在的层之后，在 count 数组中将该层拥有的节点数加一。最后判断哪些层的节点数为奇数，这些层每层可以收获一个苹果，累加后即可得到答案。

提交以后，发现还有一些测试用例无法通过，通过分析以后发现，还需要注意一个问题。在遍历数组计算每个节点所在的层时，需要注意，如果数组中的数字表示这个节点会掉落到自身节点的位置上，也就是这个节点的苹果不会往下层掉，会永远留在这个节点，因此在统计每层的节点数时，这个节点不该被统计，而掉落到这个节点的其他节点的苹果也永远不会掉落到第一层，因此这些节点也不能被统计，不属于任何层，不被统计进 count 数组。

时间复杂度：O(n)

空间复杂度：O(n)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：苹果收获程序

算法笔试模拟题精解之“恐怖的辐射”

简介：因为 N M 和最大辐射值都不大，所以可以直接模拟辐射扩散的实际情况，最后判断是否有小于等于 7 的位置。

题目描述

题目等级：困难

知识点：广度优先搜索 /BFS

查看题目：恐怖的辐射

一天，Codancer 突然发现自己身处一个奇怪的世界，他发现世界是一个 N*M 的矩形，在矩形的某些位置分布着一些恐怖的辐射源，每个辐射源都有相应的辐射等级，经过他的分析，这些辐射源总共有五个等级，分别命名为 A-E 级，A 级辐射等级为 15，B 级 14，C 级 12，D 级 7，E 级则没有辐射。

奇怪的是，这些辐射源的辐射是沿着曼哈顿距离进行传播的且随曼哈顿距离递增辐射等级递减，并且，他会绕过其他辐射源，辐射源的等级不可叠加，这意味着如果某位置同时处于两个辐射源的影响范围，则他受到的辐射为辐射等级最高的那个。

他知道当辐射等级小于等于 7 时，他受到的辐射将不会威胁到他的生命。因此，他想要知道，这个世界是否存在一个位置不会威胁到他的生命。

每组数据的第一行和第二行分别为 N,M，接下来为 N*M 的矩阵，'0' 代表空的位置，A-E 代表辐射源。（1<=N,M<=100）

输出为 T 行，每行输出 "Yes" 或 "No"。

示例 1

输入：

2

3

["0A0","00E"]

输出：

"No"

解题思路

因为 N M 和最大辐射值都不大，所以可以直接模拟辐射扩散的实际情况，最后判断是否有小于等于 7 的位置。

首先把输入的字符串转化为二维数组，每个位置的值为初始的辐射值。同时还要记录辐射源的位置，因为辐射源的辐射值不会被更高辐射覆盖，即使辐射小于等于 7 也不能生存。

因为小于等于 7 的辐射就不会致命，而且辐射值最高的辐射源只有 15，所以最多只需要遍历 8 次，也就是说辐射值最高的辐射源 7 格之外就是安全得了。

遍历时每一格的修改规则。如果是辐射源，则不修改。如果不是，修改规则如下。

用下图的 a 位置举例，设 T(a) 为 a 位置辐射的当前值。

需要对比 T(a) T(b)−1 T(c)−1 T(d)−1 T(e)−1 的值，选其中最大的作为 a 位置的新值。减 1，是因为辐射扩散一次减少 1。

最后遍历整个地图，判断是否有小于等于 7 的位置。

时间复杂度 O(9*n*m) 第一遍生成初始状态的数组，之后模拟辐射扩散

空间复杂度 O(2*m*n)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目

算法笔试模拟题精解之“树的拆分”

简介：如果我们删除了一条边，那么一定将其分成了两棵树，其中一棵树一定为原树的子树，那么我们就可以利用 DFS 序将树遍历一遍得出每个点的时间戳。

题目描述

等级：困难

知识点：深度优先搜索 /DFS、树状数组

查看题目：树的拆分

给你一个有 n 个节点的树 , 节点标号 1-n。每个节点有一个权值 w[i]，一棵树的价值为树上所有不同的权值的数量。

现在你可以删除一条边，问删除一条边后形成的两棵新树的价值之和最大为多少。

第一行输入一个 n，表示一共有 n 个节点（1 <= n <= 10^5）。

第二行输入 n-1 个数，表示第 i+1 个节点和第 e[i] 个节点有一条边相连（1 <= e[i] < i+1）

第三行输入 n 个数，第 i 个数表示第 i 个节点的权值 wi。

输出一个数字，表示删除一条边后两棵树最大的价值和

示例 1

输入：

3

[1,1]

[1,2,2]

输出：

3

解题思路：DFS

如果我们删除了一条边，那么一定将其分成了两棵树，其中一棵树一定为原树的子树，那么我们就可以利用 DFS 序将树遍历一遍得出每个点的时间戳。

DFS 序即为每个点的在一次 DFS 中的遍历顺序，从定义可以知道，子树上的每个节点的 DFS 序一定连续，因此我们可以求出每棵子树所对应的区间，剩下的节点即为另一棵树。

此时我们就可以利用树状数组求每个区间里面不同的数的个数，由于一个子树区间可能会在数组中间，会将剩下的数字分为左右两部分，因此我们需要将原权值数组扩充一倍，然后进行计算。

需要注意的是，现在这个权值数组不是输入的权值数组，而是根据 DFS 序进行重新排序的数组。

是不是有思路了呢，点击链接立刻答题：>>118. 树的拆分

算法笔试模拟题精解之“Password”

简介：可以用枚举法解决这个问题。

题目描述

题目等级：容易

知识点：字符串、枚举

查看题目：Password

Tom 在期末考完试以后学习了 Python 语言，他发现 Python 语言确实是简洁又强大，在他学完字符串以后，他写了一个随机生成密码的 python 文件。

原理是这样的，输入一个字符串 s，然后系统会随机**将这个 s 重新任意排序**，然后又生成两个字符串 s1 和 s2，并拼接起来最终生成随机密码 str = s1 + s + s2。

现在 Tom 有一个问题要问你，每次给你两个字符串，一个是 s(表示输入的字符串)，一个是 str(表示产生的随机字符串)，问这两个字符串是否符合上述要求的原理? 如果符合输出 YES，否则输出 NO。1<=s,str<=100

输入两个字符串，一个是 s，表示输入的字符串，一个是 str，表示产生的随机字符串 (1<=s,str<=100)

输出内容为一行字符串，如果符合题意中的原理输出 "YES"，否则输出 "NO"。

示例 1

输入：

"abc"

"xxxcabyyy"

输出：

"YES"

解题方法：枚举法

解题过程中用到的参数列出：

char[] s1：密码字符串的字符数组格式

char[] str1：随机生成字符串的数组格式

int[] S：密码字符串中每个字符出现的个数。(i：0-25)

int[] temp：S 数组的备份，用于恢复 S（使用 clone 方法获得）

count：表示**成功匹配到的密码字符的个数**

计算过程：

1. 将提供的字符串转换为字符数组 s1 和 str1
2. 用一个长度为 26 的 int 型数组对密码中各个字符出现的次数进行统计。
3. 从第一个字符开始，对之后的挨个序列进行判断

（1）如果 **s1** 中有这个字符，则在 **S** 中将这个字符的个数 **减 1**（减了之后判断该字符个数是否小于 0），同时将 **count+1**

若减完后 S 中对应字符个数小于 0，则表示该序列与 **s1** 不匹配，则**将 count 重置为 0**，并使用 **temp** 对 **S** 进行重置，然后直接跳到下一个字符进行判断。

（2）如果从某个字符 **(str1[i])** 开始之后的一段序列的值全部为密码字符序列中的值（即 **count 的值 = s1 中字符的个数**），则代表这个就是密码，可以直接输出 "YES" 了

（3）如果对整个 **str1** 遍历一遍后仍找不到与 **s1** 匹配的序列，则表示 str1 中没有密码，则输出 "NO"

特殊情况：如果随机字符串 **str 的长度小于**密码字符串 **s 的长度**，则肯定无法匹配成功，这个时候就直接返回 "NO"

时间复杂度：O(n^2)

空间复杂度：O(n)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：Password

算法笔试模拟题精解之“神奇数字在哪里”

简介：注意读题，本题要求的数字中“只”含有 1，2，3 三个数字。

题目描述

等级：中等

知识点：搜索、递归

查看题目：神奇数字在哪里

请计算出 1-n(1<=n<=1000000000) 中有多少个数字中，只存在 '1','2','3' 这三个数字，且这三个数字至少都要出现 1 次。

输入一个数 n。

输出 1-n 中符合要求的数字的个数。

示例 1

输入：

232

输出：

4

解题方法：

注意读题，本题要求的数字中“只”含有 1，2，3 三个数字

方法 1：暴力搜索

对本题，最简单的想法是针对每个数字，转为字符串，然后判断数字中是否只同时包含 1，2，3 三个字符。鉴于本题输入是 int，n 最大可以达到 2^31−1 ≈ 2*10^9，一个这么长的 for 循环，时间复杂度还是比较高的，不是最优解法。

时间复杂度：O(n)

方法 2：深度优先搜索

寻找这种神奇数字，暴力法可以说是从数字中“尝试”某个数字，是否符合我们的标准，这个过程好比线性搜索。但换个思路想，我们也可以尝试用“构建”的方式，构建出所有符合“神奇数字”标准的数字，并挨个计数，完成这个过程。

这个构建数字和搜索的过程，可以写一个递归做深度搜索实现：从 0 开始，给一个较短的数字，不断让把的其它位数字移动 1 位，并把多出来的个位数设为 1，2，3，并在搜索的范围超过 n 时，及时停下搜索。

时间复杂度：O(logn)（或者 O(k)，k 表示数字的位数）

方法 3：预热 + 查找

用过方法二后，我们发现，在限定了 n 是 int 类型的情况下，其实这种只含有 1，2，3 的数字的数量并不多。甚至可以在静态初始化方法里，提前计算好所有这样的数字，并按大小排列好。然后对输入的每个 n，直接在准备好的数组中二分查找，找到比它小的数字的数量，就可以返回结果了。

看完之后是不是有了答题思路了呢，快来练练手吧：神奇数字在哪里

算法笔试模拟题精解之“神奇的棋子”

简介：因为我们每选择一次，消失的数对左右都有影响，而且也会被挑选的顺序影响，所以需要去考虑每个棋子的贡献。

题目描述

等级：中等

知识点：搜索

查看题目：神奇的棋子

Tom 有一天在玩棋子，这些棋子都不是普通的棋子。现在有 n 个棋子排成一排(2<=n<=18)，每个棋子都有它们自己的权值 ai(0<=ai<=1e9)，现在 Tom 每次选择连续的三枚棋子，然后中间的那枚棋子会消失，而它左右两边的棋子会分别加上它的权值，问最后只剩下两枚棋子的时候，这两枚棋子的权值的最小的和为多少?

输入棋子个数 n 和一个数组 a，a[i] 表示每个棋子的权值。

输出一个数，表示最终剩下的两枚棋子的权值的最小的和。

示例 1

输入：

4

[1,2,3,4]

输出：

17

解题思路描述

因为我们每选择一次，消失的数对左右都有影响，而且也会被挑选的顺序影响，所以需要去考虑每个棋子的贡献。

枚举一段区间最后选的棋子，然后将其分为两段区间，设该段区间的左端点最终的贡献为 x，右端点最终的贡献为 y，那么在只剩最后选的数，左端点，右端点三个点时，我们选择最后的那个点，那么最后那个数最终的贡献就是 x+y 次了，然后根据这个思路已知分治下取就好了。

dfs(l, r, x, y) 表示左端点、右端点、左端点的贡献、右端点的贡献的区间合并成两个数的最小和。

时间复杂度最坏是 O(n^3*2^n)。

看完之后是不是有了答题思路了呢，快来练练手吧：神奇的棋子

2.3 树

算法笔试模拟题精解之“全奇数组”

简介：从题意及示例可以知道，应该从大到小进行操作。当除 2 后，需要快速查找是否有相等的其他数，这个需求可以使用 HashSet 代替。

题目描述

题目等级：中等

知识点：堆、贪心、哈希

查看题目：全奇数组

codancer 现在有 n 个正整数 a[1],a[2]…a[n]，Tom 告诉 codancer 他可以进行下列操作，选择某个偶数 x，把这 n 个数中全部等于 x 的数字除 2，Tom 想知道把这 n 个数字全部变成奇数最少需要几次这样的操作?

输入一个正整数 n(1<=n<=100000)，代表有 n 个正整数，接下来输入这 n 个正整数。

输出 codancer 把这 n 个数字全部变成奇数的最少次数。

示例 1

输入：

6

[40,6,40,3,20,1]

输出：

4

注意

1.x=40, 数组变为

[20,6,20,3,20,1]

2.x=20，数组变为

[10,6,10,3,10,1]

3.x=10, 数组变为

[5,6,5,3,5,1]

4.x=6，数组变为

[5,3,5,3,5,1]

因此最少需要 4 次

解题思路

从题意及示例可以知道，应该从大到小进行操作。当除 2 后，需要快速查找是否有相等的其他数，这个需求可以使用 HashSet 代替。

因此，先将 n 个中的偶数入 HashSet，再对 HashSet 中元素从大到小排序，依次遍历；

每个元素除 2 后从 HashSet 查找，存在则移除，计数 +1，直到该数变成奇数。

最坏情况下，除 2 过程没有重复数字

时间复杂度：O(n+n*n/2)

空间复杂度：O(n)

看完之后是不是有了想法了呢，题目直达链接：查看题目：全奇数组

算法笔试模拟题精解之“Codancer 的旅行”

简介：根据样例数据 从 1 到 2 的花费为 1，从 1 到 3 的花费为 2，从 2 到 3 的花费为 1，花费都小于 3，因此总共有三种方案。

题目描述

题目等级：困难

知识点：二分查找 / 并查集 / 贪心

查看题目：Codancer 的旅行

期末考试终于结束啦，Codancer 开始了他的旅行。

现在整个地图上由 n 个城市，这些城市之间有 n−1 条道路相连，每条道路都有一个距离，并且保证整个图是连通的，即这个地图可以看作是一棵树。

现在假设 Codancer 要从城市 A 到城市 B，那么他的路费就是从 A−B 的路径上边权最大的边的权值 wmaxx 元。

现在 Codancer 有 k 元，他想知道他能选择那些 (A,B) 并且 A<B 使得 codancer 能够到达。

第一行输入两个正整数 n 和 k，代表城市的个数和 Codancer 现有的资金数目，接下来 n−1 行每行三个数 u,v,w, 代表城市 u 和城市 v 之间有一条长度为 w 的道路。(1<=n<=100000,1<=k<=1000,1<=u,v<=n,1<=w<=1000)

输出 codancer 可选择的方案数。

示例 1

输入：

3

3

[[1,2,1],[2,3,1]]

输出：

3

注意

Codancer 可以选择从：

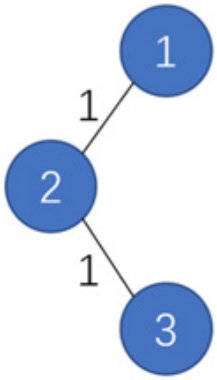
1->2

1->3

2->3

解题方法一

根据样例数据 从 1 到 2 的花费为 1，从 1 到 3 的花费为 2，从 2 到 3 的花费为 1，花费都小于 3，因此总共有三种方案。



首先按照边权将这些边从小到大排序，使用并查集记录当前连通块内的最大的边权。

当 u 和 v 连接起来的时候，假设此时的联通块大小为 cnt, 则答案应该加上 cnt*(cnt−1)/2，同时应该减去之前 u 和 v 单独为连通块的方案数。

令 ans[i] 为最大边权为第 i 条时构成的连通块的方案数，那么对于 k，二分查找最大的小于等于 k 的下标 id，最终答案就是 ans[id]。

时间复杂度为：O(nlog(n))

解题方法二

虽然题中给出的是一个树形结构，但是解题时和树的关系不是很大。

题中指出从城市 A 到 B 的路费是从 A−B 的路径上边权最大的边的权值 wmaxx 元。可以理解成对于权值大于现有资金数目 k 的边，codancer 不能通过，其他边可以任意通过。所以原始的树形结构被不能通过的边分割成了多个子区域。每个子区域没的任意两个城市是可以互相到达的。每个子区域内方案数为 C(n, 2)，n 为子区域内城市个数。

对于子区域的计算可以考虑从底向上合并的形式。创建一个 1*n 的数组 a，数组的每个位置代表一个城市，每个位置的内容代表这个城市所在的子区域。初始化时 a[i] = i。遍历时，选择可以通过的边，把两个边对应的城市所属子区域合并（修改成同样的数字）即可。

时间复杂度 O(2n) 第一遍判断每个城市属于哪个子区域。第二遍计算每个子区域城市个数，并用 C(n, 2) 求和。

空间复杂度 O(n)

看完之后是不是有了答题思路了呢，快来练练手吧：Codancer 的旅行

算法笔试模拟题精解之“Codancer 的求和”

简介：对前缀和和后缀和分别建立线段树，O(log(n)) 求解 s[i] 即可。

题目描述

等级：困难

知识点：线段树

查看题目：Codancer 的求和

现在 Codancer 有两个长度均为 n 的数组 a 数组和 b 数组，其中对于 a 中的每个元素 a[i] 都有 (−10^6<=a[i]<=10^6)，对于 b 中的每个元素 b[i] 都有 (0<=b[i]<=n)，对于每个 i(1<=i<=n)，我们定义区间 [L,R] 是合法的只有 [L,R] 满足下面的条件：

1. 1<=L<=R<=n
2. 0<=i−L,R−i<=b[i]

现在对于每个 i,Codancer 想找到一组合法区间使得 (a[L]+a[L+1]+...a[R]) 最大，并将这个最大值记为 s[i]。

现在请计算 (s[1]+s[2]+s[3]+...+s[n])。

输入包含一个 n(1<=n<=10^5) 和两个数组 b,a。(b 在前面，a 在后面)

输出 s 数组的和。

示例 1

输入：

5

[1,2,3,4,4]

[−5,1,2,3,−4]

输出：

16

注意：

s[1]=−4

s[2]=6

s[3]=6

s[4]=6

s[5]=2

因此最终答案为 16

解题思路

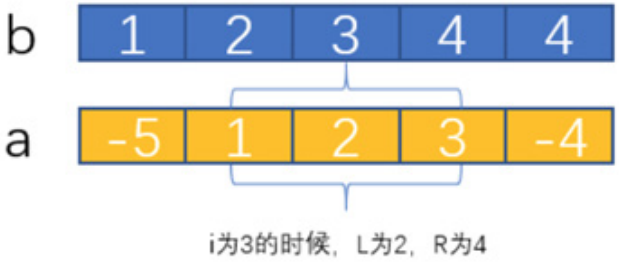
S[1]=a[1]+a[2]=−4，此时 S[1] 最大

S[2]=a[2]+a[3]+a[4]=6, 此时 S[2] 最大

同理可得

S[3]=a[2]+a[3]+a[4]=6, S[4]=a[2]+a[3]+a[4], S[5]=a[2]+a[3]+a[4]+a[5]=2

因此答案为 S[1]+S[2]+S[3]+S[4]+S[5]=16。



考虑对 a 数组做前缀和数组 pre, 即 pre[i] 是 (a[1]+a[2]+...+a[i]) 的值，同时对 a 做后缀和数组 sa, 即 sa[i] 是 (a[i]+a[i+1]+...a[n]) 的值，那么 s[i] 可以由两部分组成，左边的 a[L]+...+a[i]，右边的 a[i]+...+a[R]。

左边其实就是找到一个 L，使得 a[L]+...+a[i] 尽可能的大，右边同理 , 因此左边其实就是 max(sa[max(1,i−k)]...sa[i])−sa[i+1]。

右边是 max(pre[i]...pre[min(i+k,n)])。

对前缀和和后缀和分别建立线段树，O(log(n)) 求解 s[i] 即可，答案会爆 int, 因此需要用 long long 存储。

看完之后是不是有了答题思路了呢，快来练练手吧：Codancer 的求和

算法笔试模拟题精解之“找出二叉搜索树的第 2 大的数”

简介：这是一个关于二叉搜索树的知识点。对于二叉搜索树，若它的左子树不空，则左子树上所有结点的值均小于它的根结点的值；若它的右子树不空，则右子树上所有结点的值均大于它的根结点的值。

题目描述

等级：容易

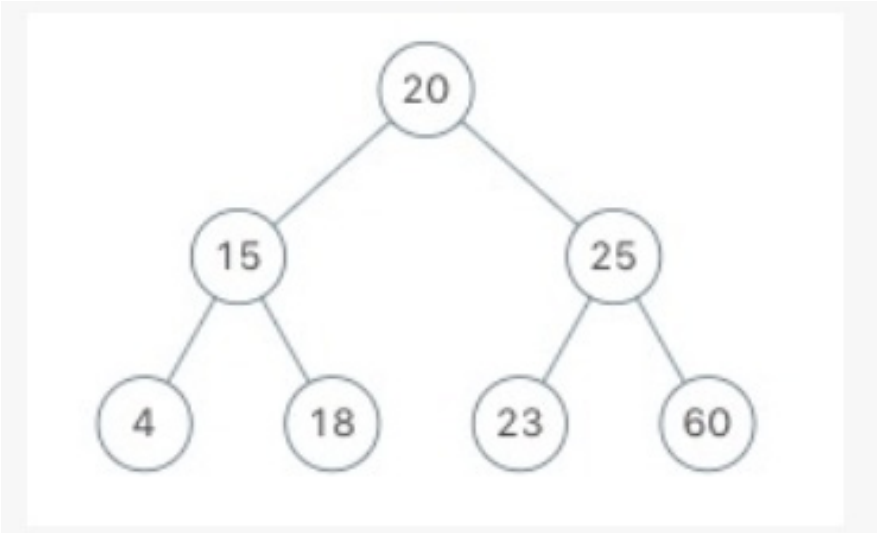
知识点：树

查看题目：找出二叉搜索树的第 2 大的数

给定一个二叉搜索树，找出其第二大的数。

示例 1

比如二叉搜索树如下



那么第二大的值是 25

注意

对于二叉搜索树，若它的左子树不空，则左子树上所有结点的值均小于它的根结点的值；若它的右子树不空，则右子树上所有结点的值均大于它的根结点的值

解题思路

这是一个关于二叉搜索树的知识点。

对于二叉搜索树，若它的左子树不空，则左子树上所有结点的值均小于它的根结点的值；若它的右子树不空，则右子树上所有结点的值均大于它的根结点的值。

因为题中没有说这是一棵平衡的树，所以某些节点可能只有一棵子树。

对于树中最大的数是很容易找到的，一直选择右子树直到右子树为空就可以了。

对于第 2 大的数，存在两种情况。一种是最大数的节点存在左子树，一种是不存在左子树。

第一种情况下。根据二叉搜索树的性质，可以知道 25 的左子树中的所有值都比

25 小，也都比 25 的父节点（20）大。所以第二大的数应该出现在 25 的左子树中。寻找 25 左子树中的最大值即可。

第二种情况下。第二大的数就是最大数节点（60）的父节点（25）. 因为 25 的父节点和左子树都比 25 小。而右子树中只有最大数一个值。

对于特殊情况，根节点没有右子树。可以归类到情况 1 中，根节点为最大的树。如果也没有左子树的话，那就是样例有问题了，因为这样树中只有一个数。

时间复杂度 O(n) 因为树不是平衡的，所以最差的情况下，从根节点开始都只有右子树，需要完整遍历整棵树。

空间复杂度 O(1) 只需要保存当前遍历到的节点即可。

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：找出二叉搜索树的第 2 大的数

2.4 线型

算法笔试模拟题精解之“最大矩形面积”

简介：根据题意，想要组成面积最大的矩形，需要有最大的长与宽，并且组成长与宽的木棍都需要有 2 根，因此，只要选择最大的两组木棍即可组成最大的矩形。

题目描述

题目等级：容易

知识点：数组

查看题目：最大矩形面积

有 n 根木棍 (4<=n<=1e5)，它们的长度分别为 a1,a2,a3...an(1<=ai<=1e9)，现在请你从中挑选出 4 根木棍来组成一个矩形，问这个矩形的最大面积是多少？

输入木棍数 n 和 n 个木棍长度

输出能组成的矩阵的最大面积

示例 1

输入：

6

[1,1,2,2,3,3]

输出：

6

解题思路

根据题意，想要组成面积最大的矩形，需要有最大的长与宽，并且组成长与宽的木棍都需要有 2 根，因此，只要选择最大的两组木棍即可组成最大的矩形。

先对数组从大到小排序，编译寻找最大的木棍，然后遍历数组。找到两个连续的相同数字，记录下这个位置，以这两个数字组成矩形的两条长。接下来从刚才记录的位置接着往下找，再找出两个连续的相同数字，以此组成矩形的两条宽。将找出的矩形的长与宽相乘，即可得到矩形的最大面积。

时间复杂度：O(n)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：最大矩形面积

算法笔试模拟题精解之“最活跃的数”

简介：根据题意，最终需要将 n 个点连通并达到最大边权，而边权为两个点的点权之和的一半，所以一个点加入连通图的最大边权就是和点权最大的点连通。

题目描述

题目等级：简单

知识点：数组

查看题目：最活跃的数

现有一个包含 n 个整数的序列 (1<=n<=1e5)，n 个数分别是 a1,a2,a3...an (0<=ai<=1e5)，现在对于每个 ai 都有 3 种操作，一种是使 ai+1，一种是使 ai−1，还有一种是不变，问在对这 n 个数操作完后，出现次数最多的数的出现次数是多少。

输入序列中整数个数 n，和 n 个整数 [a1,a2,⋯,an]

输出一个数，表示在操作过后的出现次数最多的数的出现次数

示例 1

输入：

8

[3,2,1,5,3,4,9,5]

输出：

5

解题方法：贪心

根据题意，最终需要将 n 个点连通并达到最大边权，而边权为两个点的点权之和的一半，所以一个点加入连通图的最大边权就是和点权最大的点连通。

因此想要得到最大边权和，只要所有点都与点权最大的点相连即可。

实现过程中，首先求出最大的点权，然后计算出其他点与这个权值最大的点的边权之和即可。

时间复杂度：O(n)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：最活跃的数

算法笔试模拟题精解之“非递减序列”

简介：非递减序列的定义为，序列中任意两个相邻的数，后一个数大于等于前面的数。

题目描述

题目等级：容易

知识点：数组

查看题目：非递减序列

给了 n 个数 (1<=n<=100000)，分别为 a1,a2,a3...an(1<=ai<=1000000000)，对于每一个 ai，要么不变，要么让它减 1，问能否使这个序列变为非递减序列，如果可以输出 "YES"，否则输出 "NO"。

输入序列中数字的个数 n，和 n 个数，表示每个 ai 的值

输出一行字符串，如果可以变为非递减序列输出 "YES"，否则输出 "NO"

示例 1

输入：

5

[1,1,2,1,5]

输出：

"YES"

解题方法：

非递减序列的定义为，序列中任意两个相邻的数，后一个数大于等于前面的数。

遍历数组，比较所有相邻的数字，如果发现当前数字大于后一个的数字，则当前数字需要 −1，并做一个标记，表示这个数字已经减过 1 了，不能再减 1 了。同时如果一个数字减一之后，这个数字前面的数字如果比这个数字大了，前面太大的数字也需要跟着减一。

每次做 −1 操作时，都需要检查标记，若发现在 −1 操作之前，数字已经做过标记，表明该数字无法再 −1，序列无法变成非递减序列，返回 NO。如果照上述操作整个数组都没有问题，则表明序列可以变成非递减序列，返回 YES。

时间复杂度：O(n^2)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：非递减序列

算法笔试模拟题精解之“Tom 跳方格”

简介：根据题意，此题需要求出最长非递增序列的长度。可以设置一个 count 值来记录 Tom 每次连续跳的方格数，设置一个 max 值记录连续跳的最大长度。

题目描述

题目等级：容易

知识点：数组

查看题目：Tom 跳方格

现在有 n 个方格 (1<=n<=1e5)，每个方格都有不同的高度 h1,h2,h3...hn (1<=hi<=1e9)，Tom 最喜欢跳方格了，刚开始他可以任意选一个方格作为起点，只要他右边的方格没有当前的方格的高度高，他就会不断的往右边的方格去跳，请帮助 Tom 计算一下他最多能跳多少个方格？

输入方格总数 n(1<=n<=1e5)，和 n 个数 h1,h2,h3...hn 表示每个方格的高度

输出 Tom 能连续跳的最大长度

示例 1

输入：

5

[5,4,3,2,1]

输出：

4

解题方法

根据题意，此题需要求出最长非递增序列的长度。

可以设置一个 count 值来记录 Tom 每次连续跳的方格数，设置一个 max 值记录连续跳的最大长度。

遍历数组，当右边的数字小于等于当前数字时，count 值加一，继续下一个方格。当右边的数字大于当前数字时，连续跳方格被打断，统计此次连续跳的方格数 count 是否大于 max ，若大于则用 count 替换 max。然后继续下一轮统计，并把 count 重新值 0。遍历完数组后，max 值即为 Tom 能连续跳的最大长度。

时间复杂度：O(n)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：Tom 跳方格

算法笔试模拟题精解之“复杂的字符串”

简介：这是一道极其简单的入门级题目，我们来梳理一下题目的结构：两个字符串变相同，只有唯一的一种操作方法就是删除左侧的字符。

题目描述

等级：容易

知识点：字符串

查看题目：复杂的字符串

现在有两个字符串 s1 和 s2（长度不超过 200000），Tom 是一个有强迫症的人，他想要把这两个字符串变的相同，但是每次只能删除其中一个字符串的最左端的字符，问最少需要经过多少次操作才能使这两个字符串变的相同。

输入内容为两个，第一个为字符串 s1，第二个为字符串 s2。

输出一个数字，表示最小的操作次数。

示例 1

输入：

"dadc"

"dddc"

输出：

4

题解描述

这是一道极其简单的入门级题目，我们来梳理一下题目的结构：两个字符串变相同 , 只有唯一的一种操作方法就是删除左侧的字符。

给出两个字符串 s1 和 s2（注意题目并没有说明两个字符串是等长的）。

当题目开始变换时，是这样的：

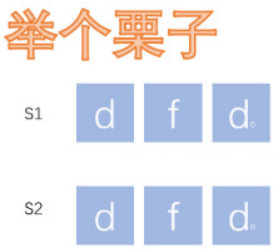
（插入为 gif 动图使用 web 模式进行浏览）

我们很容易可以看到字符串左端是变动的而字符串右端的字符是不会发生改变的，

所以当满足字符相等的条件时，最右端的字符顺序是不会改变的，

当字符串长度不等时会优先裁剪多出来的部分，然后再判断字符串长度，保证了字符串长度的相等，然后调整字符串大小每次都砍去头一个字符，直到剩余字符串相等。

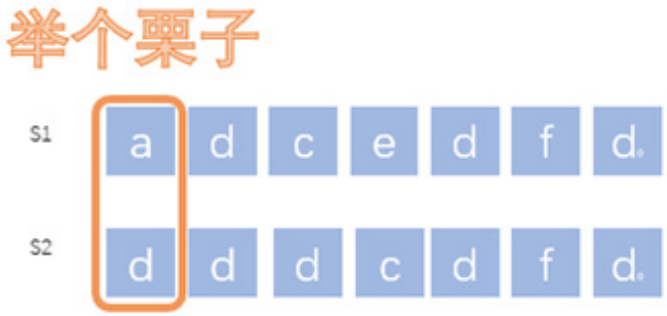
动态版 gif

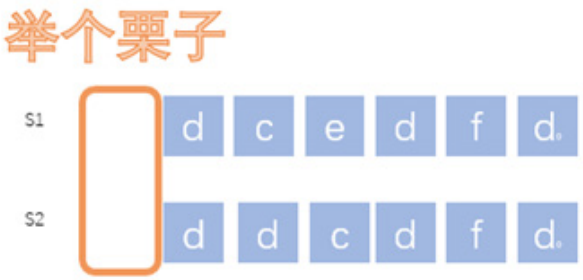


按照之前最朴素的描述我们会先裁剪字符串长度不等的部分，

得到两个长度相等的字符串再进行处理。

由于可活动调节范围是字符串的左侧 所以我们判断字符串不等后直接砍掉每个字符串左侧的第一个字符判断新生成的字符串是否相等。





然后依次删除直到剩余字符串相等为止。

最终得到相等字符串 返回删减次数即可。

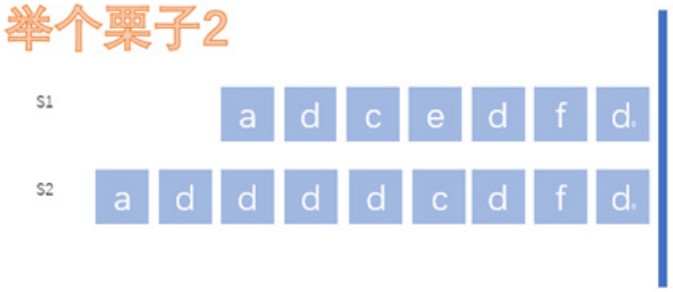
删除最多 n 次，每次删除后都需要遍历匹配剩余部分 最多判断 n*n 次

所以**复杂度 为 n*n**

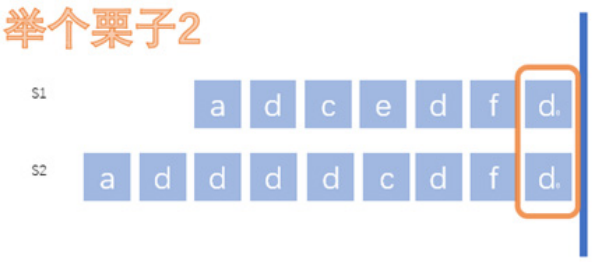
这样虽然能够得到答案但是明显过于朴素，我们很容易的就能想到换一个角度进行求解。

动态版 gif

将字符串从右侧进行对齐。

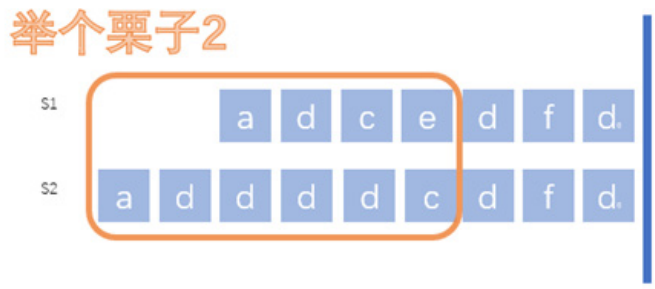


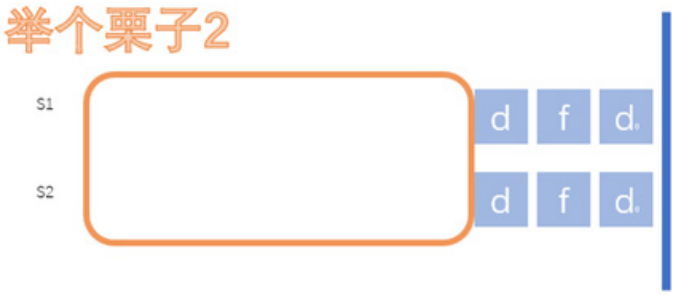
由于右侧的字符串不变，所以最终答案剩下的字符串部分是右侧连续相同的子串 , 所以我们很自然地想到从末尾开始进行匹配。



然后依次向前推进当不满足相等条件时可以停止，推进的字符串即为剩余的相等部分。

我们需要砍掉的就是其他的前面的字符。





这样就一次遍历最短的字符串就能得到最终结果。

复杂度为 1

这道题题目很简单，相信大部分人也都会做，但是题目所体现的转换思考的角度，是我们每个人在日常生活中都需要去发现的。

是不是有思路了呢，点击链接立刻答题：复杂的字符串

算法笔试模拟题精解之“神秘消失”

简介：本题可以通过观察规律得出解题思路，根据题意，两种颜色的书相邻摆放，就都会消失。可以得知只要两种颜色的书同时存在，就会处于不稳定的状态，总会有书消失，因此到最后的时候必然只能剩下一种颜色的书。

题目描述

题目等级：简单

知识点：字符串

查看题目：神秘消失

在书架上摆着一些书，这些书只有两种颜色，要么是黄色，要么是蓝色，突然某一天这些书被施了魔法，如果一本黄色和一本蓝色的书挨着，这两本书就会消失不见，然后右边的书会往左边移动，直到和左边的书挨着，如果这两本颜色不同，这两本书又会神秘消失。

现在给你一个只包含 A 和 B 的字符串 s(1<=|s|<=100000)，其中 A 表示黄色的书，B 表示蓝色的书，问这 n 本书中最多会消失多少本书？

输入一个字符串 s，s 中 A 表示黄色的书，B 表示蓝色的书；

输出最多会消失多少本书。

示例 1

输入：

"AABB"

输出：

4

解题方法

本题可以通过观察规律得出解题思路，根据题意，两种颜色的书相邻摆放，就都会消失。可以得知只要两种颜色的书同时存在，就会处于不稳定的状态，总会有书消失，因此到最后的时候必然只能剩下一种颜色的书。

分析到这里，答案呼之欲出了。两种书中，数量较少的书最终会全部消失，消失的时候又会带走一本不同颜色的书。所以消失的书的数量为：数量较少的书的数量*2。

只要把字符串转成字符数组，然后分别统计字符 A 和字符 B 的数量即可。

时间复杂度：O(n)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：神秘消失

三、计算

算法笔试模拟题精解之“最后的胜者”

简介：根据题意，可知魔法师在攻击别的魔法师时，自身不会消耗魔法值，只有被攻击时，魔法值才会有损失，损失的魔法值等于攻击者的魔法值。本题要求的就是，任意攻击的情况下，剩下最后一名魔法师的最小魔法值，可以这样解决。

题目描述

题目等级：简单

知识点：字符串

查看题目：最后的胜者

现在有 n 个魔法师 (2<=n<=100000)，这 n 个魔法师都有自己的魔法值 ai(1<=ai<=1000000000)，他们为了证明自己是最强的魔法师便开始了争夺战，任意一个魔法师都可以对其他的魔法师发起攻击，每次攻击，被攻击的魔法师损失掉的魔法值是攻击者当前的魔法值，当魔法值小于等于 0 的时候淘汰出局，问最后只剩下一名魔法师时，他的魔法值最少是多少。

输入魔法师数 n，和 n 个数，表示每个魔法师的初始魔法值

输出一个数，在任意的对决中，最后只剩下来一名魔法师的最小的魔法值

示例 1

输入：

4

[2,8,6,20]

输出：

2

解题方法

根据题意，可知魔法师在攻击别的魔法师时，自身不会消耗魔法值，只有被攻击时，魔法值才会有损失，损失的魔法值等于攻击者的魔法值。本题要求的就是，任意攻击的情况下，剩下最后一名魔法师的最小魔法值，可以这样解决。

先给各个魔法师按照魔法值从大到小排序，设置 min 的初始值为当前最小的魔法值，用这个最小的魔法值攻击其他魔法师，直到有其他魔法师的魔法值小于 min，这时 min 的值更新为此时的最小魔法值。然后继续前述攻击方法，直到 min 为 1（最小魔法值不会小于 1），或者只剩下最后一名魔法师，此时剩下的魔法师的魔法值即为最小魔法值。

时间复杂度：O(n)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：最后的胜者

算法笔试模拟题精解之“简单题?”

简介：二分法的核心在于，每次操作都让问题的复杂度减小为原来的一半。

题目描述

题目等级：容易

知识点：数学

查看题目：简单题?

有一个 1~n（1<=n<=100）的集合，现在可以让你在集合中选择任意多个数去减去一个正整数 x(x 是任意数)，问最少需要操作多少次可以使这个集合中的数都变为 0？

输入一个数 n(1<=100)，表示集合中的数据数量

输出最少的操作数

示例 1

输入：

2

输出：

2

解题方法：二分法（减治法）

思路如下：二分法的核心在于，每次操作都让问题的复杂度减小为原来的一半。

若有一个 1~n 的集合，可以让集合中大于 n/2 的数同时减去 n/2，减完后发现所有的数会变成一个 1~n/2 的集合，也就是一次操作后整个问题的复杂度变为了原来的一半。继续同样的操作，直至问题的复杂度为 0，统计这个过程中二分操作的次数即可得出最少的操作数。

时间复杂度：O(log_2 n)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：简单题?

算法笔试模拟题精解之“朋友一生一起走”

简介：根据题意可以得出，在不考虑数字范围的情况下，相加等于 k 的数总共有 k/2 对（如果 k 为偶数，应为 k/2−1 对，此处以 k/2 为例）。也就是说，如果 n 的值大于 k 的值，那么 k 的所有数对都符合条件，即 1−n 中一共有 k/2 对好朋友。

题目描述

题目等级：容易

知识点：数学

查看题目：朋友一生一起走

Tom 想从 n 个数 (1<=n<=1e14) 中把“好朋友”挑出来，“好朋友”的定义是相加等于 k(1<=k<=1e14)，其中 (1,2) 和 (2,1) 算一对，请你帮 Tom 计算一下 1−n 中一共有多少对好朋友?

输入数字总数 n 和“好朋友”数的和 k；

输出 1−n 中的好朋友有多少对。

示例 1

输入：

3

5

输出：

1

解题方法

根据题意可以得出，在不考虑数字范围的情况下，相加等于 k 的数总共有 k/2 对（如果 k 为偶数，应为 k/2−1 对，此处以 k/2 为例）。也就是说，如果 n 的值大于 k 的值，那么 k 的所有数对都符合条件，即 1−n 中一共有 k/2 对好朋友。

如果 n 的值小于等于 k，那么 k 的有些数对会超出 n 的范围，需要舍弃。

根据 n 的范围限制，可以计算得出，需要舍弃掉的对数为（k−n−1），即此时一共有 k/2 −（k−n−1）对好朋友，若计算出此数为负值，即好朋友的对数为 0。

时间复杂度：O(1)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：朋友一生一起走

算法笔试模拟题精解之“正三角塔”

简介：这是一个数学问题，将三角塔多写出几层后就可以发现规律。

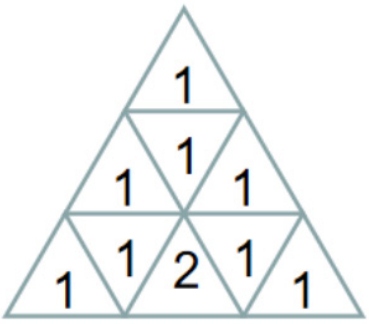
题目描述

题目等级：中等

知识点：数学

查看题目：正三角塔

一个正三角形塔，按以下规则叠 n 层，最高层（第一层）的一个三角形值为 1，接下来对于第 i 层的每个三角形，若是正三角形（尖朝上），则它等于同一层与它相邻的两个三角形值的和（若是没有两个相邻的则值为 1）；若是倒三角，则它等于第 i−1 层与它相邻的一个三角形的值。



问第 n 层第 m 个三角形的值为多少（答案对 10^9+7 取余）？

输入整数 n，表示第 n 层；和整数 m，表示第 m 个三角形（1<=n<=10^5, 1<=m<=n*2−1）

输出第 n 层从左到右第 m 个三角形的值。

示例 1

输入：

3

3

输出：

2

解题思路

这是一个**数学问题**，将三角塔多写出几层后就可以发现规律。

每一行都是两组组合数，正三角与倒三角分别为一组组合数。

对于第 k 层，

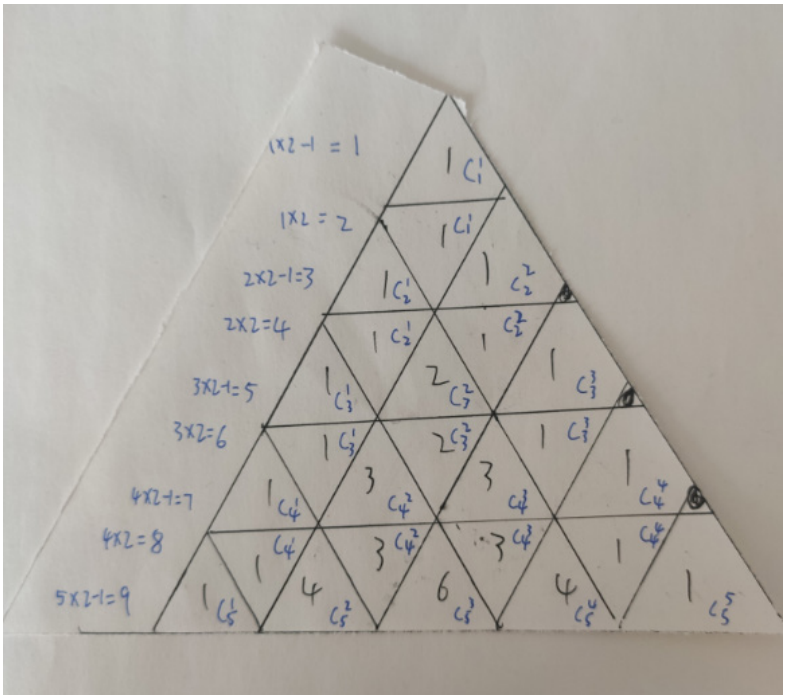
正三角的值依次为 C(k, 1) 到 C(k, k)。

倒三角的值依次为 C(k−1, 1) 到 C(k−1, k−1)。

根据题中给出的 m 值，可以判断是正三角还是倒三角，也可以判断是第几个位置。

时间复杂度 与计算组合数的方法有关

空间复杂度 与计算组合数的方法有关



看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：正三角塔

算法笔试模拟题精解之“组队难题”

简介：根据题意，本题需要找出符合 a ⊕ b>max(a,b) 条件的队伍，如果使用两重循环，两两判断学生是否符合条件，会超出时间限制。

题目描述

题目等级：容易

知识点：位运算

查看题目：组队难题

H 大学迎来了一年一度的羽毛球双打比赛，小伙伴们都很积极地报了名。

但是为了达到 1 ⊕ 1>1（注：a ⊕ b>max(a,b)）的效果，学校给每位同学进行了实力认证，每位同学都获得了一个能力值。在组队的时候，组成队伍的两位同学的能力值 a,b 必须满足条件：a ⊕ b>max(a,b)。

校长想要统计一共可以组成多少不同的队伍，请你帮助校长计算出来。

输入学生数 n(2<=n<=10^5) 和 n 个正整数 ai(1<=ai<=10^9)，表示每位同学的能力值。

输出一个整数，表示一共可以组成的队伍总数

示例 1

输入：

5

[1,2,3,4,5]

输出：

6

注意

1、如果两个队伍至少有一个队员不相同，这两个队伍就是不同的。

2、每位同学可以同时出现在不同的队伍中。

解题思路

根据题意，本题需要找出符合 a ⊕ b>max(a,b) 条件的队伍，如果使用两重循环，两两判断学生是否符合条件，会超出时间限制。

本题的约束条件 a ⊕ b>max(a,b) 中，假设 a < b，b = 1001（该数字为二进制形式），从右往左数，可以发现数字 b 的第二位和第三位为 0，若 a 想要满足约束条件和 b 组队，数字 a 的最高位必须为数字 b 值为 0 的对应的位，即第二位和第三位，若 a 等于 100 或 10，均可以满足条件和 b 组队。

理解了上一步之后，可以统计每个数字中 0 出现位置，比如第三位是 0 的数字在数组中有多少。用 HashMap 进行存储，以 0 的位置为 key，对应位置为 0 的数字的数量为 value。

统计完所有 0 出现的次数后，遍历数组，计算每个数字的二进制的位数，在 HashMap 中取出对应的 value。在遍历数组的过程中，对所有取出的 value 值求和即可得到可以组成的队伍总数。

时间复杂度：O(n)

空间复杂度：O(n)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：组队难题

算法笔试模拟题精解之“2n 合体”

简介：关键在于理解题意，题中的含义是这样的，每次挑两对 2n 合并成 m，然后判断有多少个子序列，之后再挑选不同的 2n。所以示例中的答案是 4 种，而不是 1 种。

题目描述

等级：容易

知识点：数学

查看题目：2n 合体

有一个只包含小写字母 n 和 t 的字符串 s(1<=|s|<=1000000)，其中如果有两个 n 相邻，那么它们可以合并成一个 m，现在需要你去求这个字符串中，含有多少个 'mtm' 这样的子序列。

输入一行字符串 s；

输出这个字符串中所包含的 mtm 子序列的个数。

示例 1

输入：

"nnntnnn"

输出：

4

解题思路

两个位置容易出现理解问题。

1. 子序列。子序列不要求一定是连续的。例如 abcdef 的子序列可以是 abc，也可以是 ace。
2. 含有多少个’mtm’这样的子序列。这里不是说先判断如何合并，然后在合并后的字符串上判断 mtm 子序列有多少个。

题中的含义是这样的（感觉题干确实没说清楚），每次挑两对 2n 合并成 m，然后判断有多少个子序列，之后再挑选不同的 2n。所以示例中的答案是 4 种，而不是 1 种。

有了上面的解释，理解题干就没有问题了。

求解思路：对于每一个 t 进行单独考虑，计算这个 t 的左侧和右侧分别可以合成出多少种不同的 m，这两个数的乘积就是对于这个 t 来说的 mtm 子序列的个数。

因为只有相邻的 n 才可以合成 m，所以 t 将原来的字符串分成了一系列 n 的串。

例如：nnntnnnntnntnnn

被分成了 nnn nnnn nn nnn

没有段包含的 m 个数分别为 2 3 1 2

对于每个 t 来说的 mtm 子序列就分别是 2*(3+1+2)、(2+3)*(1+2)、(2+3+1)*2；所以结果为 12+15+12 = 39

时间复杂度为 O(2*n) 第一次遍历算出每一段 m 个数，第二次遍历求和；

空间复杂度 O(n) 保存每一段 m 个数

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：2n 合体

算法笔试模拟题精解之“平行线”

简介：任意两点确定一条直线。判断两条直线是否平行就是比较两条直线的斜率。这道题对于由不同的点确定的但是重合在一起的直线应该也判断为一组平行线。此题使用简单的冒泡排序可能会超时，可以考虑更快的快速排序，归并排序或者直接使用 java 的排序函数。

题目描述

等级：容易

知识点：数学

查看题目：平行线

为了管理动物园不听话的大猩猩们，动物管理员决定去远方的动物之城找一些平行线。

当他逛到一个神奇的店铺时，他发现了一副黑色的图，上面依稀可见一些白色的点。

动物管理员询问店铺老板这幅画是什么，老板说：“天机不可泄露”。动物管理员仔细端详了一会这幅画后，他发现其中的一些白点可以连成两条平行线。

动物管理员问这幅画多少钱，老板说：“原价 ¥99999999999，但现在你只要计算出来这里面有几对平行线，就可以打折，有几对平行线就价值多少钱”。请你计算出动物管理员最少需要支付多少钱?

输入一个整数 n, 表示总共有 n 个点（1 <= n <= 1000）; 和一个含有 n 组数据 (xi, yi) 的数组，(xi, yi) 表示二维平面上一个点 (1 <= xi,yi <= 1000)，且每个点均不重复。

输出 n 个点能够找出几对平行线。(答案不超过 int)

示例 1

输入:

6

[[0,0],[1,0],[1,1],[3,1],[3,3],[5,4]]

输出:

10

解题思路:排序问题

任意两点确定一条直线。判断两条直线是否平行就是比较两条直线的斜率。这道题对于由不同的点确定的但是重合在一起的直线应该也判断为一组平行线。

一共有 n 个点,就有 n*(n-1)/2 条直线。对于点 i 和 j 确定的直线斜率为 (yi-yj)/(xi-xj).

使用一个数组保存斜率,然后排序。

排序过后相同斜率的直线就在连续的位置了。

假设斜率为 k 的直线有 a 条,对应着有 a*(a-1)/2 组平行线(任意两条相互平行)。

遍历一次排序后的数组就可以得到结果。

使用简单的冒泡排序可能会超时,考虑更快的快速排序,归并排序或者直接使用 java 的排序函数。

时间空间复杂度与排序算法有关。

看完之后是不是有了想法了呢,快来练练手吧 >> 查看题目

算法笔试模拟题精解之“叠叠高”

简介：题目中问的是可以组成多少种不同高度的塔。首先计算每种高度的塔至少需要多少张卡片，然后判断多出来的卡片数与至少需要的卡片数之间的关系。

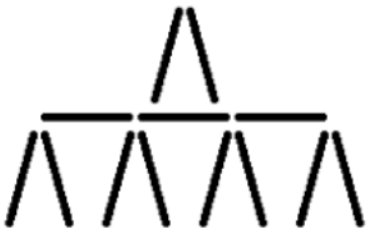
题目描述

等级：容易

知识点：数论

查看题目：叠叠高

A 同学买了 n 张卡片，他想用卡片来叠成一个塔，每层塔由数量不同的房间组成，且高层的房间数量必须少于低层的房间数量，房间可看做两张倾斜的卡片组成，两张卡片上面头部相接触。特殊的，每层的所有房间必须相邻，且高层的房间需要建在天花板上。



当 n=13 的时候只有以上两种方法去叠，但只能叠高度为 2 的塔。

相邻两个房间之间用一张卡片作为天花板相连。现在他想知道 n 张卡片全部使用的情况下他一共可以叠成多少种不同高度的塔。

输入一个整数 n(1 <= n <= 10^9)，表示卡片数量；

输出一个数字，可以叠成的高度的种类数。

示例 1

输入：

13

输出：

1

解题思路：数学题

题目中问的是可以组成多少种不同高度的塔。

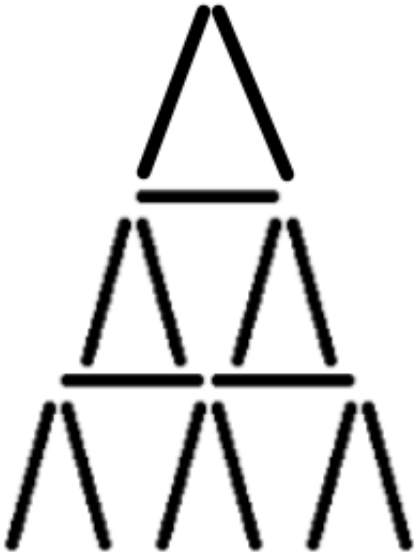
首先计算每种高度的塔至少需要多少张卡片，然后判断多出来的卡片数与至少需要的卡片数之间的关系。

观察下图三层的塔。从上向下。

第一层一个房间，第二层两个房间，……，第 i 层 i 个房间。

一个房间需要 2 张卡片，两个房间需要 5 张卡牌（每个房间 2 张 + 天花板 1 张），……，i 个房间需要 2**i+i−1 = 3**i−1 张。

这样每一层的卡片数都可以计算出来了，前 k 层的卡片数求和即可。记前 k 层最少需要 f(k) 张卡片



接下来观察多余卡片与房间的关系。下图中圈出来的房间位置从 1 层放到了 2 层。因此，对于多出来的卡片，我们可以都放到第一层处理（因为题目只问有多少种不同高度，而不是多少种不同的搭建方法）。

每多出来一间房，需要 3 张卡片（包括天花板）。所以多余的卡片为 3 的倍数，则这个高度的塔可以搭建。

计算过程

1. 从第一层开始，计算 f(i)，直到 f(i) 大于 K。

2. 判断每个 f(i) 与 k 的差值是否为 3 的倍数，是，则总数加 1。

时间空间复杂度都很小。f(i) 不需要一个数组来保存，只需要算出当前层的直接判断是否为 3 的倍数即可。

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目

算法笔试模拟题精解之“公平”

简介：可以先求出两个小朋友初始的糖的数量差 diff，如果 diif 为 0，则发糖次数为 0。

题目描述

题目等级：容易

知识点：数学

查看题目：公平

到了万圣节，Tom 要给小朋友们发糖，现在有两个小朋友，他们手里分别有 x 个糖和 y 个糖 (1<=x,y<=1e9)，但是糖少的小朋友就会不开心，Tom 想让他们两个的糖一样多。

Tom 的操作是这样的，第一次给他们其中一个小朋友发一个糖，第二次给他们其中一个小朋友两个糖，第三次给他们其中一个小朋友发三个糖，以此类推，问至少要多少次这两个小朋友的糖会变的一样多?

输入两个数字，输入 x 和 y，表示两个小朋友刚开始所拥有的糖数。

输出 Tom 要发多少次使得两个小朋友的糖一样多。

示例 1

输入：

[1,4]

输出：

2

解题思路

可以先求出两个小朋友初始的糖的数量差 diff，如果 diif 为 0，则发糖次数为 0。

如果 diff 不为 0，则需要先计算如果连续给一个小朋友发糖，至少需要发几次才可以使小朋友的糖一样多或实现反超，记这个次数为 n，发 n 次糖总共发的糖的数量为 mount = (n*n+n)/2。

如果 mount – diff 为偶数，则最后发糖次数为 n。若为奇数，发糖次数为 n+2。

时间复杂度：O(n)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：公平

算法笔试模拟题精解之“Tom 的手工课”

简介：对于每个方案来说，最少需要 n/2 个彩带，最多要 n−1 个彩带，然后我们分别对其进行计算贡献。

题目描述

等级：中等

知识点：数学、计数

查看题目：Tom 的手工课

一天 Tom 在上手工课，老师给他们每个人发了一个白色的纸条，上面有 n 个方格 (2<=n<=1e6)。

然后又给他们每个人发了 n−1 个彩带，一个彩带可以粘到两个相邻的方格上。现在老师让他们把 n 个方格都粘上彩带 (可以不用完 n−1 个彩带，一个方格上可以重复粘彩带)。

Tom 是一个热爱数学的人，他想知道所有的方案中，一共用了多少次彩带 (所有的方案所用的彩带的总和)。(答案对 1e9+7 取模)

输入一个数 n 表示方格的个数。

输出一个数表示最终方案数，答案对 1e9+7 取模。

示例 1

输入：

3

输出：

4

解题方法：

对于每个方案来说，最少需要 n/2 个彩带，最多要 n−1 个彩带，然后我们分别对其进行计算贡献。

操作最多 i 次的方案数是 f[i]，恰好 i 次的方案就是 f[i]−f[i−1]，而 f[i]=C(n−1−I, i−1)。

具体含义：可以看作是每次可以选择 +1,+2 , 求构成 n−2 的方案数 , 我们先默认都 +1, 剩下就是选择 +0,+1 了 , **只要总共的 i−1 次操作中有 n−1−i 个选择了 +1, 就一定可以达到目标了**。

看完之后是不是有了答题思路了呢，快来练练手吧：Tom 的手工课

算法笔试模拟题精解之“填数问题”

简介：“对于任意一个编号为 i 的格子，编号大于 i 的格子上的数都大于等于 i 号格子上的数”可以转化为 m 个格子上的数是单调不减的。

题目描述

等级：中等

知识点：数论、数学

查看题目：填数问题

有 m 个格子，编号为 1,2…m，每个格子可以填 1,2...n 中的任意一个数。定义这 m 个格子上的数都是“好的”，仅当对于任意一个编号为 i 的格子，编号大于 i 的格子上的数都大于等于 i 号格子上的数。求有多少个填数方案，满足这 m 个格子中填的数是“好的”，答案对 P 取模。

三行分别输入三个整数，m、n、P，分别表示有 m 个格子、填入的最大数字 n 和模 P。(保证 1<=n,m<=1e18,2<=P<=1e5 且 P 是质数)

输出一个整数表示答案对 P 取模的结果。

示例 1

输入：

2

2

11

输出：

3

解题方法：

“对于任意一个编号为 i 的格子，编号大于 i 的格子上的数都大于等于 i 号格子上的数”可以转化为 m 个格子上的数是单调不减的。令 cnt_i 表示 i 这个数在 m 个格子中的出现次数，可以发现一组 $\sum cnti = m(i \in [1,n])$ 唯一对应着一个单调不减的填数方案。

由插板法可知，n 个非负整数和为 m 的方案数为 (n+m−1,n−1)。所以答案就是 (n+m−1,n−1)，n,m 很大时需要用卢卡斯定理。

复杂度 O(logp n)

看完之后是不是有了答题思路了呢，快来练练手吧：填数问题

算法笔试模拟题精解之“Jerry 的考验”

简介：本题的关键在于理解题意：所谓挑选 n 个字符变成 a 和 b 两个字符串，是指在原字符串中抽出 n 个字符，这些字符的的顺序保持不变，剩下字符的顺序也保持不变，由此组成 a 和 b 两个字符串。

题目描述

等级：中等

知识点：搜索、字符串、位运算

查看题目：Jerry 的考验

有一天 Jerry 给 Tom 出了一道题来考验他。Jerry 给了 Tom 一个长度为 2*n 的只包含小写字母的字符串，让 Tom 将这个字符串任意挑选字符，将其分成两个等长的字符串 a 和 b(对于一个 si 不能同时被选到 a 和 b 中)，然后 a 要和 reverse(b) 相同 (a 和反转后的 b 相同)，问这样的方案数有多少？ Tom 有些为难，所以请你来帮帮他吧。

输入一个正整数 n，和一个长度为 2*n 的字符串

输出方案数。

示例 1

输入：

2

"abba"

输出：

4

解题思路描述

本题的关键在于理解题意：所谓挑选 n 个字符变成 a 和 b 两个字符串，是指**在原字符串中抽出 n 个字符，这些字符的的顺序保持不变，剩下字符的顺序也保持不变**，由此组成 a 和 b 两个字符串。

例如 "abcdef"，挑选第 2、3、5 个字符，则分成 "bce" 和 "adf" 两个串。

接下来是整理的思路解析：**整体框架是 dfs，枚举每个字符属于 a 还是属于 b，搜索过程中需要利用 a 和 b 的对称性做加速处理，否则会超时。**

比方说

xccccddcccxdd

从左往右枚举 a 字符串的构成，如果令第一个 x 属于 a，根据对称性，倒数第三个字符 x 一定是属于 b；如此推导出末尾的 dd 一定属于 a，中间位置的 dd 一定属于 b，而且是 b 的头两个字符；然后左边 ccc 一定 a，右边 ccc 一定是 b，由此得出 1 种方案。令第一个 x 属于 b 也可以用同样的方式得到 1 种方案。

用这个思路直接写代码不太好写，可以通过枚举二进制，固定左半边的选择情况，然后对于每一个 case，通过 dfs 搜索右半边有多少种合法组合，搜索过程中利用对称性进行剪枝。

对于字符全部相同 case 如 "aaaaaaaa"，因为过程中无法剪枝，会退化成 2^(2*n)。对于这种 case，答案就是 C(2n,n) ，预判一下直接返回即可。

看完之后是不是有了答题思路了呢，快来练练手吧：Jerry 的考验

算法笔试模拟题精解之“超车”

简介：只要超过了一辆车，就算发生了超车，按照此思路批量判断超车，提高内循环的效率。

题目描述

等级：容易

知识点：模拟

查看题目：超车

一天某地在举行公路自行车比赛，一共有 n 名选手参加 (2<=n<=100000)，在比赛的过程中，有一段路程观众是看不见的，现在给你了选手进入这段路程的顺序编号，又给了选手出这段路程的顺序编号（顺序按照先进先出的原则，编号为 1-n），请你计算一下，有多少名选手在这段公路上完成了超车（出路段后超过了进路段前在自己前面的选手）。

第一行输入一个数 n，第二行输入 n 个数，表示选手进入路段的顺序，第三行输入 n 个数，表示选手出路段时的顺序输出一个数 s，表示有 s 名选手在该路段中超车

示例 1

输入：

5

[4,2,1,3,5]

[2,4,1,5,3]

输出：

2

注意

输入样例表示 4 第一个进，2 第二个进 .... 然后 2 第一个出，4 第二个出 ...

解题思路详解

方法 1：暴力解法（不能通过）

根据题意，超车意味着对于入洞前序列中的每辆车 x 及 x 前面的车集合 {a,b,c,d...}，如果在出洞序列中，发现 x 前面的某车，出洞时不在 {a,b,c,d...} 集合中，就说明 x 超过了这辆不在 {a,b,c,d...} 集合中的车。而且这个超车行为与其他车无关，也就是说，即使上面个的清空中 x 的排名后撤了，被其它车超过了，它也依然被视为发生了超车。**“只要超过了一辆车，就算发生了超车”**。

根据上面思路，首先需要遍历入洞序列，对每辆车 x 遍历，同时用一个列表记录下入洞序列中 x 前面的车。对这样的 x 和列表，再去出洞序列中从尾部开始遍历，直到发现 x 停止。如果遍历出洞序列的过程中，发现某车存在于列表中，说明这辆车入洞前在 x 的前面且出洞后在 x 的后面，发生了超车，此时令结果加 1。

如果简单应用这种代码，不加优化，会有嵌套循环，时间复杂度很高，不能通过。

时间复杂度： O(n^2)

空间复杂度： O(1)

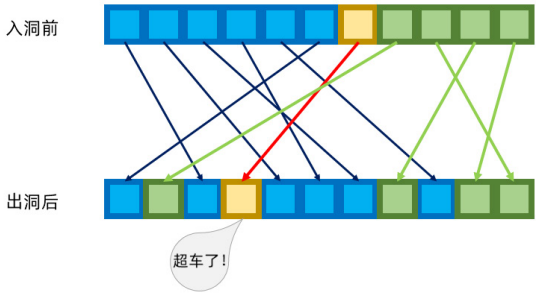
方法 2：批量判断超车，提高内循环的效率

上面的思路中，耗时最多的部分，就是内循环**在出洞序列中，判断“入洞序列在 x 前面车的集合”，有没有出现在 x 的后面**这个步骤。如果直接查找，由于出洞序列没有顺序，不可以使用二分查找，只能从挨个线性搜索出洞序列数组，时间复杂度非常高。所以我们先从这里切入，尝试优化。

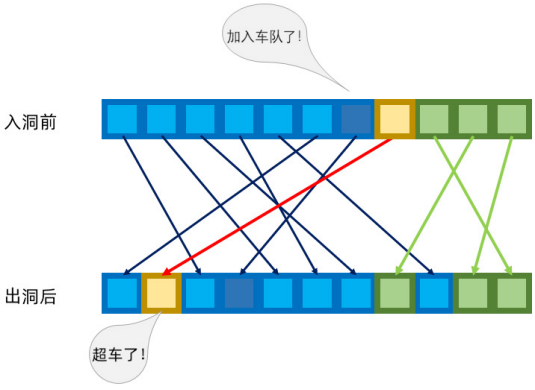
根据上面的定义，我们想，既然 x 只要超过入洞序列中，x 前面的任何一辆车就

算超车，而且题目不要求我们求出超过了哪辆车，也不要求求出超过了几辆车，那么我们可不可以将超过一辆车的行为，视为 **x 超过了入洞序列 x 前面所有车组成的车队**的行为呢?

首先我们需要用哈希表（或者表示入洞出洞顺序的数组），记录下每辆车出洞和入洞排名的对应关系；**之后从 0（或者说 1）开始遍历入洞排名**，找到入洞序列中排名 i−1 及 i−1 之前的车组成的**“车队”，在出洞序列中覆盖的范围**，你不需要记录这个范围中具体覆盖了那些车，只需要记录这个范围的第一辆车 a 和最后一辆车 z。最后，判断 i 车在出洞时的位置 x，是否超过了这个车队的最后一辆车的位置 z。如果超过了，说明发生了超车，如果未超过，说明可以把 i 车也加入到车队中，将车队的最后一辆车位置更新为 x。然后继续这个遍历过程，就可以得到结果：



上图中的黄色车，如果要判断黄色车有没有超车，只要记录下蓝色车组成的“车队”，在出洞序列中的范围，有没有覆盖到黄色车的出洞位置，就可以判断了。
在判断过后，黄色车相当于“加入”了蓝色车队，使得我们可以继续判断下一辆车，有没有超过这个“车队”。



这种解法利用了哈希表查找速度，编写合理的话，时间复杂度会比较低。

时间复杂度：O(n)

空间复杂度：O(2n)

看完之后是不是有了答题思路了呢，快来练练手吧：超车

算法笔试模拟题精解之“坏掉的时钟”

简介：本题关键在于理解题意。

题目描述

等级：容易

知识点：模拟

查看题目：坏掉的时钟

B 同学有一个时钟，能够显示 1-d，初始值为 1。这个时钟每天显示的数字加一，特殊的，当某天显示的值为 d 时，第二天就会显示 1。

但是每个月的时间并不总是 d 天，因此 B 同学就要通过手动调整使得显示的时间正确，每次手调都可以使显示数字加一。

现在给你 n 个月每月的天数，请你计算一下若是让时钟每天显示的数字都是正确的，他这 n 个月一共需要调多少次时钟。

输入月份数 n（1 <= n <= 10^5）、时钟的最大显示时间 d（1 <= d <= 10^4）和一个包含 n 个数的数组，第 i 个数表示第 i 个月有 ai 天（1 <= ai <= d）

输出使时钟正常显示一共要调整的次数。

示例 1

输入：

3

5

[3, 4, 3]

输出：

3

解题方法：模拟

本题关键在于理解题意：

题干的含义是，在除去最后一个月后，其余每个月的最后一天的 24 点时，时钟上的逻辑时间会超过那个月的最大天数，同时实际时间变为下一个月的第 1 天。此时逻辑时间和实际时间有差别，需要调整时钟，让逻辑时间重新回到 1，使其符合实际时间。

如题中例子所指：

```
3
5
[3, 4, 3]
```

- 第 1 个月第 1 天，时钟实际值 1，符合；
- 第 1 个月第 2 天，时钟实际值 2，符合；
- 第 1 个月第 3 天，时钟实际值 3，符合；
- 第 2 个月第 1 天，时钟实际值 4，不符合，对时钟进行 2 次加 1 操作，时钟实际值变为 1，符合；
- ……
- 第 3 个月第 1 天，时钟实际值 5，不符合，对时钟进行 1 次加 1 操作，时钟实际值变为 1，符合；
- ……
- 第 3 个月第 3 天，时钟实际值 3，符合；
- 结束，共需要调整 3 次。

时间复杂度：O(n)

空间复杂度：O(1)

趁热打铁，快来练练手吧：查看题目：坏掉的时钟

算法笔试模拟题精解之“期末考试”

简介：根据题意，需要计算被抄的期望人数。那么首先要计算每个人被抄袭的概率。

题目描述

题目等级：容易

知识点：概率

查看题目：期末考试

期末考试到了，n 名标号 1-n 的同学坐成一排参加考试。考完试后，为了防止混乱，监考老师决定依次让第 n 个人的卷子传给第 n-1 个人，第 n-1 个人的卷子传给第 n-2 个人 ... 第 2 个人的卷子传给第 1 个人，这样老师就能够直接收到所有人的卷子了。但是同学们经过了多年的考试，都练就了一身抄答案的好本领。再传卷子的过程中，第 i 个人都有 ai 概率抄到第 i-1 个人或者 bi 概率抄到第 i+1 个人。特殊的，a0 与 bn 均为 0。

但是每个人在抄他人的同时又不想被别人抄，被抓到了也得受罚。因此现在他们需要计算一下没有被抄到卷子的期望人数，以决定此次考试是否要大家一起作弊。

入参有四个，第一个是一个整数 n，表示总人数。

第二个是一个整数 m，表示被抓人数的上限。(1<= m <= n <= 10^5)。

第三个输入 n 个数，表示 a1-an。

第四个输入 n 个数，表示 b1-bn。

若此次被抄的期望人数不超过 m，输出 "Yes"，否则输出“No”。(不包括引号)

示例 1

输入：

3

1

[0.000,0.500,0.500]

[0.500,0.500,0.000]

输出：

"No"

解题方法

根据题意，需要计算被抄的期望人数。那么首先要计算每个人被抄袭的概率。

第 i 个人可能被 i−1 抄，也有可能被 i+1 抄。被 i−1 抄的概率是 pre = b[i−1]，被 i+1 抄的概率是 next = a[i+1]。(若 i−1 或者 i+1 不存在，则被其抄的概率为 0)

在计算 i 被抄袭的概率时，可以先计算 i 不被抄袭的概率，再用 1 减去不被抄袭的概率即为被抄袭的概率。被抄袭的概率为：1−(1−pre)*(1−next)。

将每个同学被抄袭的概率相加，即可得到被抄袭的期望人数。

时间复杂度：O(n)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：期末考试

算法笔试模拟题精解之“滑雪比赛”

简介：可以分别计算出每个朋友滑到终点所需要的时间，比较得出其中的最小值即为比赛结束时间。要注意的一点是，如果在到达终点时刚好需要休息，那这个休息的时间是不用计算的。

题目描述

题目等级：容易

知识点：模拟

查看题目：滑雪比赛

Jerry 造完滑雪场之后又修建了一条长度为 M 的滑雪赛道，他邀请了一些朋友来他的滑雪场进行滑雪比赛，第一名将会 Jerry 版滑雪套装!

Jerry 一共邀请了 N 个朋友来他的滑雪场参加滑雪比赛，由于他的朋友都不是专业的滑雪运动员，所以每滑一会儿雪都要停下来休息一会儿补充体力，也就是说 Jerry 的第 i 个朋友在滑雪滑了 ti 秒之后，需要停下来休息 si 秒来补充体力，在休息期间位置不会变化。他们的滑雪速度都是 a m/s，只有当第一个人滑倒终点的时候才会比赛结束。

请问比赛进行了多久之后会结束?

前三行分别输入三个整数 N,M,A 表示 Jerry 的朋友数量，赛道的长度，Jerry 的朋友滑雪速度 (2 ≤ N ≤ 1000)，(1 ≤ M ≤ 10000)，(1 ≤ A ≤ 100) ,M 是 A 的倍数

接下来 N 行，每行两个整数 ti 和 si (1 ≤ ti ≤ 100)，(1 ≤ si ≤ 100)

输出一个整数，表示比赛开始后经过多少秒结束

示例 1

输入：

2

100

1

[[10,5],[5,10]]

输出：

145

注意

第一个人，滑雪 9 次，休息 9 次，再滑 1 次就达到终点，比赛就结束了，一共经历了 10 9+5 9+10=145 s

解题思路

可以分别计算出每个朋友滑到终点所需要的时间，比较得出其中的最小值即为比赛结束时间。

要注意的一点是，如果在到达终点时刚好需要休息，那这个休息的时间是不用计算的。

计算朋友滑到终点所需要的时间可以分为两部分，一部分是完整周期（一个完整周期等于朋友滑一次雪加休息一次），一部分是周期外的剩余时间。完整周期外的时间单独计算，这部分时间全部为滑雪时间。

将这两部分时间加起来就可以得到一个朋友滑到终点所需要的时间。比较朋友花费的时间，得出的最小时间即为答案。

时间复杂度：O(n)

空间复杂度：O(1)

看完之后是不是有了想法了呢，快来练练手吧 >> 查看题目：滑雪比赛

免费刷题神器
海量题目等你来挑战

阿里云开发者“藏经阁”
海量免费电子书下载