题目讲评

2024 菜鸟杯华中师范大学程序设计竞赛

华中师范大学 ACM 程序设计协会

2025-03-22

目录

比赛情况

本次共有 77 位选手有提交记录，其中 74 位选手至少通过了一题。

目录

find out the mvp

Harden5 正在和他的好朋友一起开黑，但是打的异常吃力，他很快意识到队伍里面有躺赢狗。虽然某个队友说“你那么在意那个评分干嘛，那个评分他是会将人的付出给异化掉的”，但是 **Harden5** 还是想找出躺赢狗。

他盒出了 n 个人的名字和他们的评分，并认为所有评分不高于 3.0 的人都是躺赢狗，并要求你把这些人的名单按照他们的评分从小到大输出。如果有多个人评分相同，优先输出名字字典序更小的。

题解

使用任意时间复杂度不高于$O(n^2)$的排序算法即可

目录

Sound name

小 C 最近在学习音乐的过程中接触了一点乐理知识，如上图所示，`1(do)2(re)3(mi)4(fa)5(sol)6(la)7(si)`

这 7 个音阶对应的音名依次为 `CDEFGAB`。

现在，每次输入两个由单个大写字母 a_i, b_i 表示的音名，请输出其中**音调较高的音**的音名。

题目包含多组测试数据。

题解

做一个字符变换，比大小的时候把 A 变成 H,B 变成 I,然后把字符比大小,输出的时候再转变回去即可

目录

good-numbers

N405 的众人在讨论什么数是世界上最好的，有人说是质数，有人说是完全数，这时候 Sadbo1 站出来说，我觉得好数是这样的:我们定义一个数 x 为好数，当且仅当存在正整数 $l < r$，使得$x = l + (l + 1) + ... + r$。

这时 Whitecarrot 发难了：“现在给你一个区间 $[L, R]$，你能快速回答这个区间里面有多少个数是好数吗？”

Sadbo1 觉得聪明的你肯定能回答这个问题，你需要回答 T 组询问。

题解

打表，我们枚举l,r计算区间和，然后从小到大排序，你会发现最终结果是神秘的 2 的幂次消失了。二分找包含了几个 2 的幂次即可。

目录

change-offer

在一个神秘的世界线, CCNUACM 的 n 位成员都得到了**一份** offer ,但俗话说别人的才是最好的，于是他们决定交换 offer ,他们会将自己的这**一份** offer **随机等可能**的给**除了自己以外的任意一人**。如果在交换结束后，**所有人**手中都有一份新的 offer ，那么这就是一次**能给大家带去笑容的交换**。

LogSingleDog 想知道一次交换**能给大家带去笑容**的概率为多少。

在发生交换以后有一些人可能会有多份新 offer，而有些人可能收不到 offer

题解

我们用发生了带给大家带去笑容的方案除以所有情况的方案，首先我们来考虑分母，所有情况的方案怎么计算呢。

题解

首先我们有 n 个成员，每个成员会等概率选择 n-1 中的一个成员发送 offer，因此根据乘法原理，总方案数为$(n-1)^n$

题解

分子如何计算呢，我们设a_i为第 i 个成员选择发送 offer 的成员编号，那么序列A为收到 offer 的成员编号。想要每个人都收到 offer，那么序列A应当是一个排列，同时每个人都不会给自己发 offer，因此序列A是一个错位排列，所以分子为长度为n的错排数

最终概率：n 的错排数$/(n-1)^n$

目录

putoutthefire

纵火犯想要烧光洛杉矶，Trump 为了关爱人民与国家，决定亲自灭火。但是由于年老体衰，他每次只能携带一桶水。

现在纵火犯在 n 个地方放了火，第 i 个着火的位置在 (x_i, y_i) 处。一桶水刚好可以灭一个地方的火。由于当地建设不发达，Trump 每次只能到 $y = h$ 的河流处取水（河流无限长），而 Trump 只能在格点上进行移动。无论他携带的桶中有没有水，他每移动一个单位长度所消耗的体力始终为一。

他当然想要消耗的体力最小，所以他要求首席大臣你来计算他灭完所有火消耗的最小体力。

题解

首先，Trump 要尽快到达河流。所以首先直接上下移动到达河流。

由于 Trump 一次只能抬一桶水，所以最优的策略中，我们应当在河流上左右移动。而对于每一个要扑灭的火源点(x_i, y_i)，我们应移动到对应的同一横坐标的位置(x_i, h)处，进行取水，上下移动，以达到最小距离。显然的，对于每个着火的位置，都会消耗$2 \cdot |y_i - h|$的体力。

此时我们就只需要考虑 Trump 如何在$y = h$这条河流上移动的路径尽量少，所以两种可能的方式是：

1. 到达最左边的火，然后从左往右依次扑灭
2. 到达最右边的火，然后从右往左依次扑灭

题解

然而，Trump 在扑灭最后一团火之后，不应该再移动。那么此处我们就不需要再返回河流。所以我们要再考虑 Trump 最后停下来的点，考虑以这个点作为结束点所能减少消耗的体力能有多少。

时间复杂度：$O(n)$

目录

treeplanting

小念和凉凉树在植树节这天约好了一起去吃好吃的，但是 ufowoqqqo 被他们放鸽子了很生气，他们要选出一个人去找 ufowoqqqo 道歉。

凉凉树给出了一棵有N 个节点的树，ufowoqqqo 会随机给出一个点$X(1 \leq X \leq N)$，小念和凉凉树轮流删去树上只有度为1的节点和这个节点的边，谁先把点X删掉就可以让对方去道歉，小念**先手**。

题解

首先，如果点 x 的度数为 1，那么先手必胜。

题解

我们假象一种极限情况，如果点 x 度数为 2,那么假设你先手，你是不会想把 x 的度数变为 1 的，如果你把 x 的度数变为 1，那么后手直接获胜。所以如果 x 的度数为 2 的时候，先手和后手都会删一些不会影响 x 度数的点，所以谁赢谁输只和 n 的奇偶性有关系。 n 为奇数后手胜，n 为偶数先手胜

目录

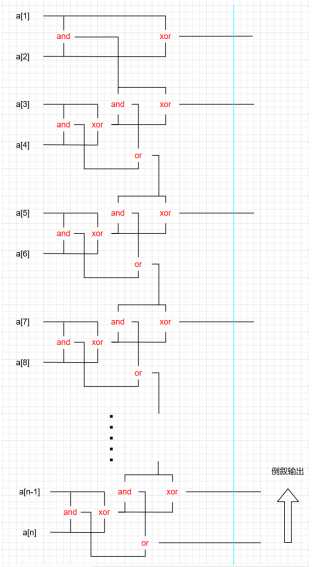
logic-gate

逻辑门（Logic Gates）是在集成电路（Integrated Circuit）上的基本组件。

由于 Sirius 没学过数字逻辑，当她看到这样一个电路图时她只能用 and、xor、or 标出每个逻辑门的种类，并在左边标出输入电路的n个电平a_i（a_i为 0 或 1）。请你帮她计算出这个复杂的电路会在输出口输出什么。

图中逻辑门用 and、xor、or 表示，在逻辑门上下的导线表示输入，右边的导线表示输出。a_i代表高低电平，0表示低电平，1表示高电平，黄点表示导线互相连接。and、xor、or 的运算法则与 C++中相同，例如 and 逻辑门中若输入低电平、高电平，则输出低电平，若输入高电平、高电平，则输出高电平。 蓝色线段右边的为输出口，低电平用0表示，高电平用1表示。输出顺序如图所示（从下往上倒叙输出）。

logic-gate



题解

看图写代码，大力模拟。我们每次循环处理 a[i],a[i+1]这两个逻辑变量，然后用 tmp 记录从之前的逻辑变量算出来，需要参与当前变量的 and 和 xor 的变量。根据电路图，计算输出和更新 tmp 即可。

目录

finalexam

VladmirZ 正在准备他的期末考试。不幸的是，他什么也没学过。

为了期末周复习，VladmirZ 将需要复习的知识点抽象成了一个 $n * m$ 的矩阵。VladmirZ 每次会选中一组 **L 型方格**，并让他非常厉害的队友教他。每次进行这个操作要花费代价 k，并且选过的方格无法再次选中。

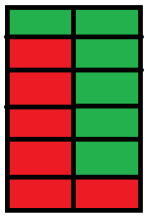
他想知道为了复习到至少 60 分（**学会矩阵中至少 60% 的方格，应当向上取整**），他至少要花费的代价是多少？特别的，如果 VladmirZ 必定挂科，请输出 -1。

> **L 型方格**：矩阵中的一组方格。其位置满足：存在且仅存在一个方格 P_0，使得其余的点均两两相邻的排布在与该点相同的行或列上，且方格 P_0 的四个相邻方格中恰有两个未被选中。

> **说人话**：顾名思义，方格形成一个 **L**，而不是一根棍，也不是一个点。

题解

注意到，当 $\min(n,m)=1$ 时，我们无法选取到任何一组 **L 型方格**，所以答案是 -1.
当 $\min(n,m)=2$ 时，对于 $2\le\max(n,m)\le 5$，我们可以选取任意一个角落来填充至少 60% 的方格，而当 $\max(n,m)>5$ 时，我们可以选择类似下面这种填充方式。



对于剩下的情况，我们贪心的考虑，每次选择最左下角的 **L 型方格**，这样做的花费是最少的，第 i 次填充的个数为 $n+m+1-2*i$，这构成了一个等差数列，于是我们可以二分等差数列的项数去计算出满足条件的最小操作次数，复杂度是 $\mathrm{O}(T\log n)$

目录

colorfulstring

每个魔法乐谱由若干音符构成，每个音符拥有一个颜色。定义乐谱的前 k 个音符为**和弦前缀**，其颜色由第 k 个音符决定。两个和弦前缀 s 和 t 的**差异度**为：

$$|s| + |t| - 2 \times lcp(s, t)$$

其中 $lcp(s, t)$ 是 s 与 t 的最长公共前缀的长度。

给定 n个魔法乐谱的具体内容，请帮助喜多计算**所有同色和弦前缀的差异度之和**。若两个前缀颜色相同，则它们的每一对组合均需计入总和。

形式化的说，你需要求出

$$\sum_{c \in C} \sum_{1 \le i < j \le |C|} \left(|s_i| + |s_j| - 2 \times lcp(s_i, s_j) \right)$$

colorfulstring

其中，C 是每个颜色组，$|C|$ 是该组中和弦前缀的数量，s_i 和 s_j 是该组中的和弦前缀。

fun fuct

fun fact: 英文题目名中的 string 既指字符串，又指琴弦

题解

出题人的做法：转化为树上问题来求解

做法一：

首先我们建字典树，把n个串全部都塞进去。字典树上的一个节点表示多个字符串的前缀，所以一个节点应当有一个颜色二元集合C_i（$<c,num>$表示颜色c的前缀有num个）表示表示i节点对应的多个字符串前缀的颜色集合。我们先只考虑单一一种颜色形成的树。所求为

$$2 \times (\sum_{u \in V} dep[u] \times (|V| - 1) - 2 \times \sum_{u,v \in V} dep[lca(u,v)])$$

题解

首先对于式子的前一部分我们是可以遍历每个串完成计算的，而后半部分，我们维护d[u][c]表示以节点 u 为根的子树中颜色为 c 的节点数量，计算 lca 位于 u 的颜色为 c 的那一部分的贡献，此时贡献的计算方式为首先考虑 d[u][c]和 d[son[u][1][c]之间产生的贡献

$$2 \times d[u][c] \times d[son[u][1][c] \times dep[u]$$

.

题解

之后我们考虑 son[u][2]引发的贡献，此时你会发现只要一个子孙在 son[u][2]中取，一个在 u 和 son[1]中取，所以遍历一遍所有儿子，维护之前的 d[u]和参与贡献的 son 的 d 之和就可以计算贡献。 我们利用树上启发式合并对于一个点 u 使用一个 $set < pair < int, int >>$维护出以 u 为根的子树中某种颜色出现了几次，然后直接暴力维护 set 合并的时候顺便维护后面那个式子的和。

做法二：如果你看懂了单一颜色的做法，那么我们不妨只让树变成单一颜色，利用虚树直接建出每种颜色对应的树，然后直接在树上求解即可。

做法三：利用点分治维护出每个节点 d，然后算贡献。

题解

做法?（由人类心智的荣耀（国内）选手提出）：首先处理所有前缀，把颜色相同的归为一类，然后对同一类的字符串做类似 SA 的排序，因为做出后缀数组以后，求解 lcp 可以转换为求解区间 min，那么对于后面那个式子就转换成了一个简单单调栈模型，类似一个序列求解$\sum_{\{l<r\}} \min_{\{l\le i\le r\}} a_i$，就可以计算出后面那个式子的贡献。

目录

number-string

Soubai 创造出了一个很大很大很大的数，大到把 **CCNUACM** 实验室的众人都吓到了，这时**小红**拿出了无穷多个 * 和 +，希望 **Soubai** 把这些运算符添加到这个数中间，使之变成一个表达式，满足表达式合法的同时让表达式的结果最小(表达式中不允许出现数字有前导0，注意数字0不算含有前导0)。

Soubai 想知道有多少种方法可以让表达式结果最小。

由于方案数可能很大，你需要输出方案数对 998244353 取模的结果。

题解

没 0 情况：

在思考这种情况时，我们首先应该发现添加符号永远优于不添加符号。

我们首先注意到对于一个连续的"1"子段，如果其两边有数字且不全是“2”，例如31112，我们可以发现这中间的四个空位有且只有一个+号并且其余全是*号时为最优。那么对于无“1”的子段且无连续“2”的子段，例如2332，我们可以发现，全部插入+号永远比不插入符号或插入*号更优。

那么对于两边不全是2长度为n的连续1子段的方案数是n+1。

接下来我们来考虑“2”这一特殊数字，因为2+2==2*2。我们先考虑连续“2”子段。那么我们可以发现只要*号不连续出现（如2*2*2+2），那么整个连续“2”子段的值就是最小的。此时这个连续“2”段内的方案数可以通过递推求得。

我们设f(n)为2的连续长度为n时的方案数。

如果连续2段的长度为n，最后一个符号如果选择*，那么最后第二个符号就只能选择+号，此时指定了最后两个符号且不影响前面的符号选择，那么此时的方案数等于f(n-2)，如果最后一个符号选择+，那么同理，此时方案数等于f(n-1)。那么f(n) = f(n-1)+f(n-2);

但是此时我们发现，连续”2“子段中间如果夹杂”1“，仍然具有类似只有”2“时的性质。类似21112（不

题解

被2包裹的1不算在内），我们可以发现全部取*号和取1个+号其余为*号的值是一样的。那么推广一下，类似211221112这样的情况，我们也可以发现全部取*号的21...12段不能相邻。那么这种情况也可以递推求得。

我们对每个两边为2中间只有1和2的子段，记录每两个2之间的1的个数，记为数组a，我们设计一个二维dp数组

dp[i][0]代表第i个间隔有一个+号，dp[i][0]代表第i个间隔全是*号。

初始化：

```
dp[0][0] = a[0];
dp[0][1] = 1;
```

状态转移：

```
dp[i][0] = (dp[i-1][0]+dp[i-1][1])*a[i];
dp[i][1] = dp[i-1][0];
```

那么最终的答案是不包在2中间的连续1子段的方案数之积乘上21...121..12子段的方案数之积。

题解

有 0 情况：

由于最小答案永远可以是0，所以不是所有位置都需要添加符号。采用dp写法。

s[i]为第i位数字

dp[i][0/1/2/3]

i从i-1推得

状态设计

dp[i][0]表示i位前面插入符号并且前缀结果为0

dp[i][1]表示i位前面插入符号并且前缀结果不为0

dp[i][2]表示i位前面不插入符号并且前缀结果为0

dp[i][3]表示i位前面不插入符号并且前缀结果不为0

初始化

```
dp[0][0]=0;
dp[0][1]=0;
dp[0][2]=0;
dp[0][3]=1;
```

题解

状态转移

那么考虑当前点是0字符

```
dp[i][0]=dp[i-1][0]*2+dp[i-1][2]*2+dp[i-1][3]+dp[i-1][1];
dp[i][1]=0;
dp[i][2]=dp[i-1][0]*(s[i-1]=='0'?0:1)+dp[i-1][2];
dp[i][3]=dp[i-1][1]+dp[i-1][3];
```

当前点不是0字符

```
dp[i][0]=dp[i-1][0]+dp[i-1][2];
dp[i][1]=dp[i-1][0]+dp[i-1][1]+dp[i-1][2]+dp[i-1][3];
dp[i][2]=dp[i-1][0]*(s[i-1]=='0'?0:1)+dp[i-1][2];
dp[i][3]=dp[i-1][1]+dp[i-1][3];
```

答案为`dp[n-1][0]+dp[n-1][2];`

目录

rain of Kunming

题目描述

今天 wqsing 发现了一颗杨梅树，共 n 个点

每个树节点生长着一个杨梅，结点 i 的杨梅重量为 w_i

wqsing 总共只能摘下总重量不超过 m 的杨梅

这时 sad 出现了，sad 向 wqsing 询问了 q 个问题

每次询问为：如果只能摘下从点 1 到点 x 的简单路径上的杨梅，并且不能摘下点 y 处的杨梅，总共有多少种摘下杨梅的方案

注意：不摘下任何杨梅也是一种方案；数据不保证点 y 一定在点 1 到点 x 的简单路径上。

方案数可能很大，请输出方案数模 998244353 之后的结果。

题解

先考虑 1 到 x 的简单路径中所有点都能选的情况

发现其实就是树上 01 背包

$dp[x][k]$ 表示 1 到 x 的路径上所有点都能选，且总权重为 k 的方案数

那么转移如下 $dp[x][k] = dp[fa][k] + (k \geq a[x];?;dp[fa][k - a[x]] : 0)$

题解

设 $f[x][k]$ 表示 1 到 x 的路径上不能选点 y，且总权重为 k 的方案数

那么可以发现如下关系 $dp[x][k] = f[x][k] + (k \geq a[y];?; f[x][k - a[y]] : 0)$ 也就是权重为 k 的总方案数分为不选点 y 和选点 y 两种情况

不选点 y 的方案数就是 $f[x][k]$

选点 y 的方案数就是 $f[x][k - a[y]]$

根据上面关系，可以得出 $f[x][k]$ 的转移式子 $f[x][k] = dp[x][k] - (k \geq a[y]? f[x][k - a[y]] : 0)$ 然后发现第一维度 x 其实不需要，也就是 $f[k] = dp[x][k] - (k \geq a[y];?; f[k - a[y]] : 0)$ 如此对于每个询问，我们通过预处理得到的 $dp[x]$ 计算出 f 即可

题解

由于数据不保证点 y 一定在 1 到 x 的路径上，导致我们需要 dfs 判断一下点 y 是否在 1 到 x 的路径上，

如此复杂度也就是 $O(n*m+q*(n+m))$

其实也可以离线处理答案，在 dfs 中用 vis 数组标记在 1 到 x 的路径上的点，然后直接判断并离线统计答案即可

复杂度也就是 $O(n*m+q*m)$

题解

其实 $dp[x][k]$ 的第一维也是可以压缩的

在离线处理了某个x 的所有询问的答案后，其实 $dp[x]$ 只会用于 $dp[y]$ 的转移

$$dp[x][k] = dp[fa][k] + (k \geq a[x]; ?; dp[fa][k - a[x]] : 0)$$

此式子其实与上面 f 数组的式子非常相似，其实事实上本质就是相同的

对于 $f[x][k]$ 表示 1 到 x 的路径上不能选点 x，且总权重为 k 的方案数，其实也就等价于 $dp[fa][k]$

$$ndp[k] = dp[k] + (k \geq a[x] ? dp[k - a[x]] : 0) \setminus dp = ndp$$

更进一步，其实如果按照 k 从小到大进行转移，就不用新开数组（类似于 01 背包优化第一维），便可以通过 $dp[fa]$ 得到 $dp[x]$

题解

转移如下：

```
for(int k = 0; k <= m; ++k) { dp[k] += (k >= a[x] ? dp[k - a[x]] : 0) }
```

同理，如果希望由 $dp[x]$ 得到 $dp[fa]$，可以如下转移

```
for(int k = 0; k <= m; ++k) { dp[k] -= (k >= a[x] ? dp[k - a[x]] : 0) }
```