2016 元旦欢乐赛
解题报告

Gromah

Contents

1 Fibnacci 解题报告

1.1 20 分算法

直接暴力进行操作就可以了。

时间复杂度 $O(nm)$，可以拿下前 2 个测试点。

1.2 40 分算法

其实我并不会，只是出在这里看有没有什么奇怪的算法来做这部分数据点 ...

1.3 60 分算法

其实我并不会，只是出在这里看有没有什么奇怪的算法来做这部分数据点 ...

1.4 100 分算法

考虑用线段树处理这个问题。首先，我们定义伪斐波那契数列：

- $G_0 = a, G_1 = b$
- $G_n = G_{n-1} + G_{n-2} (n \geq 2)$

考虑如果要把两个伪斐波那契数列各项相加，那么我们实际上就只需要把两个序列中的 a 和 b 分别相加就可以了，然后这个新的伪斐波那契数列的每一项就会等于之前的两个序列的对应项的和。

那么我们就可以给每一个区间维护这样的 a, b 标记，然后我们只需要快速求出 G_k 以及数列的前 k 项和，那么这个问题就可以解决了。因为 $k \leq n$，所以我们就可以直接预处理出 n 个转移矩阵。然后每当要求值的时候，我们就可以直接进行一次矩阵乘法就可以了。不过这里就会有一个比较大的常数。~~要不然我怎么会开 2 秒。~~

时间复杂度 $O(n \log n)$，可以拿下全部测试点。

2 Monster 解题报告

2.1 20 分算法

考虑动态规划。设 $Dp[x][u]$ 表示 x 秒后，点 u 上的“大年兽”的数量。对于每一个状态都可以枚举 v，如果 $dis(u,v)=1$ 或者 $u=v$ 则进行转移，即：$Dp[x+1][v]+=Dp[x+1][u]$。

时间复杂度 $O(t4^n)$，可以拿下前 2 个测试点。

2.2 40 分算法

对于 20 分算法的动态规划，我们考虑优化：因为每次转移的转移矩阵都是一样的，于是可以很自然地想到用矩阵乘法加速递推。

时间复杂度 $O(8^n \log t)$，结合之前的算法可以拿下前 6 个测试点。

2.3 60 分算法

对于 20 分算法的动态规划，我们考虑优化：首先有 $Dp[x+1][u]+=Dp[x][u]$，然后枚举 $i\in[0,n)$，令 $Dp[x+1][u\oplus 2^i]+=Dp[x][u]$。答案即为 $Dp[t]$。

时间复杂度 $O(tn2^n)$，结合之前的算法可以拿下前 6 个测试点。

2.4 100 分算法

对于 60 分算法的动态规划，我们考虑优化：这个转移过程看起来就比较特别，实际上就相当于是 $Dp[x+1]=Dp[x]\oplus A$，其中 $\oplus$ 表示异或卷积，而对于 A 的话，只有下标为 0 或者 2^i 的元素为 1，其他的元素都是 0。那么这样就可以用快速幂加速运算了。

时间复杂度 $O((n+\log t)\times 2^n)$，可以拿下全部测试点。

2.5 Extra

异或卷积是一个这样的东西：我们定义：$C=A\oplus B$，则对于 C 中的每一个元素 C_i 有：

$$C_i=\sum_{j=0}^{2^n-1} A[j]\times B[i\oplus j]$$

这个东西的快速算法和 FFT 差不多，只不过式子可能有些不同，具体的我就不详细讲了。

丢链跑：http://picks.logdown.com/posts/179290-fast-walsh-hadamard-transform

3 Star 解题报告

3.1 20 分算法

枚举每一个直线集合，然后看这个集合内的所有直线是否都经过同一个点，这样做可以保证一个点只会被计算一次。具体地，我们可以求出这个集合内任意两条直线的交点，然后看其他直线是否都经过这个点就可以了。

时间复杂度 $O(n \times 2^n)$，可以拿下前 2 个测试点。

3.2 40 分算法

其实我并不会，只是出在这里看有没有什么奇怪的算法来做这部分数据点 ...

3.3 60 分算法

我们其实可以直接枚举每一个交点，然后再看有多少条其他的直线经过这个点就可以了。

注意去重。

时间复杂度 $O(n^3)$，可以拿下前 6 个测试点。

3.4 100 分算法

我们同样可以求出所有交点，其实我们只要对于同一个点的出现次数做个统计就可以知道有多少条直线共这个点了。所以我们就可以给交点排个序然后扫一遍就可以了。

注意精度。标程是用双关键字 Hash 来代替实数运算的。

时间复杂度 $O(n \log n)$，可以拿下全部测试点。