

2022年SDUTACM集训队暑假选拔赛题解

2022年SDUTACM集训队暑假选拔赛题解

比赛出题组

- [A] Ginger's number
- [B] Ginger's game
- [C] Ginger's sequence
- [D] Ginger's line
- [E] Ginger's coloring
- [F] Ginger's treasure
- [G] Ginger's password
- [H] Ginger's clone
- [I] Ginger's balance
- [J] Ginger's cow
- [K] Ginger's palindrome
- [L] Ginger's function

比赛出题组

- m_rd
- 波风——水门
- Yfan_
- xdx
- 凌乱之风
- Kininged
- wudima6
- Lj
- XY
- Rezarc
- Wraith_G
- drywwltu

[A] Ginger's number

@(m_rd)

题解

显然删除 x, y 的公因子是最大的，所以答案为 $\gcd(x, y)$

[参考代码](#)

[B] Ginger's game

@(波风——水门)

题解

首先考虑暴力，对于任意位置 $1 \leq i \leq n$ ，我们可以按照顺序依次遍历 $i, i - 1, i - 2, \dots, 1$

如果当前满足 $a_k \leq a_{k+1}$ 那么把 a_k 加入答案 sum , 如果当前满足 $a_k > a_{k+1}$ 那么我们可以把 a_k 设置成与 a_{k+1} 相等, 然后加入答案 sum 。显然我们以这样的贪心方式可以得到以 i 为起点的最优解。然后我们把每一个位置 i 所得到的 sum 求出一个最大值即为答案。

通过观察 $O(n^2)$ 做法, 我们不难发现, 对于任意位置 $1 \leq i \leq n$, 假设 j 是 i 左边第一个满足条件 $a_j \leq a_i$ 的下标, 然后此时我们可以得到位置 i 的最优解 dp_i , $dp_i = dp_j + (i - j) \times a_i$, 就是说 j 到 i 之间的数都会被改为 a_i , 然后类似于一个递归的过程去处理 dp_j 的状态, 又因为是从前往后的递推顺序, 这里也可以理解成有一点动态规划的感觉。

现在的问题就被转化成如何快速求出左边第一个满足条件 $a_j \leq a_i$ 的下标, 这样就可以用单调栈维护这个递推问题。

[参考代码](#)

[C] Ginger's sequence

@(Yfan_)

题解

我们从数量的角度考虑, 对于一个长度为 n 的数列, 它含有 $2^n - 1$ 个子序列(对于每一个数选或不选, 一共是 2^n 个, 再减去一个空集), 当 $n = 17$ 的时候, $2^n - 1 = 131071 \geq 10^5 \geq k$, 这时候根据鸽巢原理(抽屉原理)一定是有解的, 所以当 $n \geq 17$ 的时候答案一定是 **YES**, 当 $n \leq 17$ 的时候, 我们直接 2^n 爆搜出所有方案即可。

[参考代码](#)

[D] Ginger's line

@(xdx)

题解

$O(n^2)$ 暴力判交点或者开个桶存一下斜率即可。

[参考代码](#)

[E] Ginger's coloring

@(凌乱之风)

题解

首先对于一个排列 p , 如果看成环图的结构, 那么 i 就向 p_i 连一条无向边。所以对于任意一个排列就会产生若干个环, 连通性可以用并查集维护, 现在对每个点进行黑白染色, 题意转换为对于环中任意相邻两点颜色不能相同, 那么只有偶数元环才能够染色成二分图, 而每个偶环的方案数为 2, 设当前有 x 个环, 总答案就为 2^x , 当且仅当所有的环都为偶环时存在答案, 否则答案为 0

[参考代码](#)

[F] Ginger's treasure

@(Kininged)

题意: 从 n 个数中选出 k 个使其乘积最大, 输出最小字典序的方案。

下面给出一种可行的做法。

为了让乘积尽量大, 对于负数的选择, 应尽量选择偶数个。

当 k 为奇数时，必然不可能选奇数个负数，否则答案将为负数；那么正数必然选择奇数个，假设先选出来最大的正数，选择一个后， k 变为偶数。如果 k 为奇数且最大的数为正数，按照这个策略选择依然不会让答案变差，因为如果所有的数都为负数，容易想到的是选择 k 个相乘起来绝对值最小的数，所以最大的负数是必须选择的。

再考虑偶数的情况，这组数的组成必然是偶数个正数 + 偶数个负数，贪心地选取 $\frac{k}{2}$ 次，每次选出当前最大乘积的一对正数或者一对负数，可以保证乘积最大。

具体实现方法为把数值从小到大排序，使用双指针每次比较头和尾两对正数/负数，选择乘积最大的一组，重复 $\frac{k}{2}$ 次。

这样做就可以选出最大乘积。

对于输出字典序最小的方案，再次从上述贪心策略中考虑。对于每次选取的一对正数/负数：如果选择正数，由于正数是从数组尾开始选，为了保证字典序最小，相同的数值我们要选编号尽量靠前的；选择负数的情况同理。这样能保证对于相同的数值，肯定能优先选到小的编号。

需要注意的是，如果最终得到的最大乘积为 0，选择一个数值为 0 且编号最小的，剩余 $k - 1$ 个数选最前面的即可。

[参考代码](#)

[G] Ginger's password

@(wudima6)

题目要求除第一位外每一位都不比前一位字典序小，所以最终密码串一定是 `abccccddddddeffff.....z` 这种形式，当然也有某些字母不存在的情况。

所以问题就转化为：

cnt_i 为第 i 个字母的个数，求满足 $\sum_{i=1}^{26} cnt_i = k$ 的个数。

对于本题就是 $now_i \leq b_i \leq a_i$ ，求满足 $\sum_{i=1}^{26} b_i = k$ 的个数， now_i 为给出的串中的字符个数。

显然可以得到 dp 方程 $dp_{i,j} = \sum_{z=cnt_i}^{\min(a_i,j)} dp_{i-1,j-z}$ ，答案为 $dp_{26,k}$

本题也可以转化为背包，可参考[洛谷P1077摆花](#)题解。

[dp代码](#)

[背包代码](#)

此题也可以用多项式，对 $now_i \leq b_i \leq a_i$ 做一步转化，得到 $0 \leq b_i \leq a_i - now_i$ ，求满足 $\sum_{i=1}^{26} b_i = k - n$ 的个数， now_i 为给出的串中的字符个数，构造生成函数

$$F(x) = \prod_{i=1}^{26} \sum_{j=0}^{b_i} x^j$$

因此直接用 NTT 求 $[x^{n-k}]F(x)$

[参考代码](#)

[H] Ginger's clone

@(LJ)

题解

两人同时出发，转向的条件是当前方向的下一个格子超出矩阵范围或者已经被人走过，记录方向，只需要在更新坐标前判断下一个格子是否合法即可，若有一个无法继续走，说明两人都已到达各自的终点。

[参考代码](#)

[I] Ginger's balance

@(XY)

题解

假设现在有 3 个物品，两个分别放天平两端，若天平平衡，那么劣质品就是没放在天平上的那个，若不平衡，则是较轻的那个，若有 3 堆物品也是按照这种方案处理。那么在天平一次能装无限多个的情况下， k 次称量保证能称出 3^k 个物品。

现题目要求天平托盘只能装有限个，那么在天平能按三分的方法处理，即个数 $\leq 3n$ 之前，需要把物品数量减少，即每次称量 $2n$ 个，次数为 $\frac{m - 3n + 2n - 1}{2n} = \frac{m - n - 1}{2n}$ ，再加 $2n - 1$ 是为了处理无法整除的情况。在这样操作以后，剩下的物品就可以按照三分的方法处理。注意，只有 1 个物品时输出 0

[参考代码](#)

[J] Ginger's cow

@(Rezarc)

题解

二分图板子题。建边时用 map 映射一下将字符串映射成数字的点，建图后跑二分图匹配即可。

[参考代码](#)

[K] Ginger's palindrome

@(Wraith_G)

当要像题意中一样将拼接回文串，那么拼接字符串时 就可以把已经回文的地方直接删去，留下剩下的接着凑回文。

凑回文时，会发现每次剩下的部分要么是空，要么是字符串的前缀或者后缀。

通过这个性质，可以把字符串的所有前后缀都处理出来

再把所有前后缀与原始字符串进行匹配，这也就是连边的过程

所有前后缀加上放的位置的状态看成一个点标号，最终到回文就是终点，使用字符串看成边，每个字符串一开始使用的状态看成起点，就变成了多起点多终点的最短路问题，建立一个超级原点和超级终点，通过 Dijkstra 求最短路。若最后怎么拼接都无法出现回文，就启动第二方案，若字符串中字符为奇数的字符个数为 ≤ 1 ，那么通过调整顺序就可以构造出回文，否则就再拼上当前字符串一次，价值为原来的两倍，最后取个最小值即可。

[参考代码](#)

[L] Ginger's function

@(drywwltu)

题解

简单签到，由于 $n \leq 11$ 所以可以考虑搜索，对于 $(1 - x^{\lfloor \frac{f_i}{2} \rfloor} - x^{f_i})$ 发现他只有三项，所以只有复杂度最大只有 3^{11} 对于前面的系数我们可以开 map 存起来（也可以开 2×10^7 的数组）。

[参考代码](#)