Codeforces Round #278 解题报告

大连市第二十四中学 于纪平

目 录

0 概要

这一场Codeforces的题目是我和陶润洲同学出的，在此来分享一下详细的中文解题报告。

所有的题目可以在http://codeforces.com/contests/487,488找到。

1 2A - Giga Tower

1.1 题目大意

给出一个整数a，求一个最小的正整数b，使得$a+b$的十进制表示当中包含数字8。

$-10^9 \le a \le 10^9$。

1.2 算法

考你会不会编程，从小到大枚举b检验就可以了。

注意，枚举到10是不够的，因为当$a=-8$的时候答案是16。

2 2B - Candy Boxes

2.1 题目大意

有4个正整数，其中n个已知。求一种给剩下$4-n$个数赋值的方案，使得这4个数的平均数、中位数和极差都相等，或输出无解。

$0 \le n \le 4, 1 \le a_i \le 500$。

2.2 算法1

用最基本的数学推导可知，任何合法的方案一定是$\{x, y, 4x-y, 3x\}$的形式。

那么，从1到500枚举x，从x到$2x$枚举y，依次检验是否能与已知的n个数匹配。

2.3 算法2

对于$n = 0, 1, 4$的情况，直接依次特判掉。

这时，至多有两个未知数，从1到1500枚举每个未知数就可以了。

3 A - Fight the Monster

3.1 题目大意

勇士打怪兽，双方分别拥有生命值、攻击力、防御力三种属性。每回合，双方会互相造成伤害，伤害值为自己的攻击力减去对方的防御力。一旦生命值减为0或更低就算被打败。给出双方的初始属性和勇士购买每种单位属性值的价格，问至少花费多少钱能打败怪兽而自己不被打败。

输入的所有数都是100以内的正整数。

3.2 算法

显然，如果勇士的攻击力至少为怪兽的防御力与生命值之和，就没必要继续买攻击力；如果勇士的防御力至少为怪兽的攻击力，就没必要继续买防御力。所以，最终的攻击力不会超过200，而防御力不会超过100。

枚举最终的攻击力和防御力，就可以算出战斗的时长，以及消耗的生命值，进而算出总价格。取最优解就可以了。

消耗的生命值也可以二分后模拟计算。

4 B - Strip

4.1 题目大意

把长度为n的数组切成若干个连续的段，要求每一段的长度至少为l，并且极差不超过s，问最少切的段数。

$n, l \le 10^5, -10^9 \le a_i \le 10^9, s \le 10^9$。

4.2 算法

考虑动态规划：设$dp[i]$表示数组的1..i前缀最少切成的段数，那么$dp[i] = \min_{j=left(i)-1}^{i-l} dp[j] + 1$，其中$left(i)$表示一个以$i$结尾的段的最靠前的可能的开头位置。

我们可以二分计算$left(i)$的值，二分后转化为了RMQ问题。计算出$left(i)$以后，算这个dp式子也是一个RMQ问题。

使用线段树或者ST表处理RMQ，总的时间复杂度为$\Theta(n \log^2 n)$或$\Theta(n \log n)$。

这道题也可以使用单调队列将时间复杂度优化至$\Theta(n)$。

5 C - Prefix Product Sequence

5.1 题目大意

给出n，求一个$[0, 1, \cdots, n-1]$的排列，使得这个排列的前缀积在模n意义下也是一个$[0, 1, \cdots, n-1]$的排列，或输出无解。

$n \le 10^5$。

5.2 算法1

首先特判掉$n = 1$。

显然，0应该放在最后，否则前缀积就会有某个后缀都是0。所以，前缀积序列的倒数第二个元素一定是$(n-1)! \mod n$，且这个数不能与最后的0相等。

考虑一个大于4的合数n，将它分解成$n = pq(2 \le p \le q < n)$。

如果$p \neq q$，那么显然$pq|(n-1)!$，从而$(n-1)! \mod n = 0$。

如果$p = q$，那么$2p < n$，从而$2pq|(n-1)!$，$(n-1)! \mod n = 0$。

所以对于大于4的合数n总是无解的。对于$n = 4$我们可以特判输出$\{1, 3, 2, 0\}$。现在只剩下n为质数的情况。

先考虑这个问题的前缀和版本。即求一个$[0, 1, \cdots, n-1]$的排列，使得这个排列的前缀和在模n意义下也是一个$[0, 1, \cdots, n-1]$的排列。

显然，0应该放在最前面，否则前缀和就会有两个相同的数。对于n为奇数的情况，整个排列的总和是$n(n+1)/2$，这是n的倍数。这要求前缀和的第一位和最后一位都是0，所以n只能是1或偶数。

偶数的情况是很好构造的，例如$n = 8$，我们可以构造$[0, 1, 6, 3, 4, 5, 2, 7]$，它的前缀和是$[0, 1, 7, 2, 6, 3, 5, 4]$。

现在回到前缀积的问题。既然n是质数，我们可以找到它的一个原根g。只要我们解决了规模为$n-1$的前缀和问题，那么把这个解作为指数，g作为底数求幂，最后再补一个0就是前缀积问题的答案了。

5.3 算法2

上述算法需要用到原根，比较复杂。

还是特判掉$n = 1$和$n = 4$。考虑n是质数的情况，我们大胆猜测：前缀和序列是$[1, 2, \cdots, n-1, 0]$，即原序列的第0位是1，第i位是$(i+1)/i$（模n意义下）。

现在证明原序列的元素两两不同：假设$(i+1)/i$与$(j+1)/j$相同，则$(i+1)j = (j+1)i$，即$i = j$。

所以只需要按照这个输出一遍就可以了。

6 D - Conveyor Belts

6.1 题目大意

有一个$n \times m$的矩形地图，每个格子写着”<>ˆ”之一的字符。每次如果从一个格子出发，就要顺着箭头方向走，直到走出地图，或者死循环。

现有q次询问，每次可以修改一个位置的字符，或者查询如果从某个位置出发的目标点（或输出死循环）。

$n \leq 10^5, m \leq 10, q \leq 10^5$，修改次数$p \leq 10000$。

6.2 算法1

给地图水平分块，对于每个格子处理出走出当前块的目标点。

对于每次询问用这个预处理结果加速模拟就好了，对于每次修改暴力重构当前块。

设块大小为S，那么每次修改的时间是$O(S)$，每次查询的时间是$O(nm/S)$。总时间$O(pS + qnm/S)$。

取$S = O(\sqrt{qnm/p})$，总时间$O(\sqrt{pqnm})$。

6.3 算法2

用线段树维护行，每个区间记录区间的最后一行的m个点的走出这个区间的目标点。两个区间可以$O(m)$合并（用类似两个置换相乘的方法），叶子可以$O(m)$计算。

每次修改的时间为$O(m \log n)$，每次查询的时间$O(\log n)$。

总时间$O(nm + q \log n + pm \log n)$。

6.4 算法3

考虑离线处理，对询问分块，每S个修改分一块。

对于每一块的开头，暴力处理出此时除了将要被这一块的修改影响的点的所有点的目的地。每个目的地可以是：地图边界外，死循环，或者是一个将要被修改的点（称为特殊点）。

对于每个查询，用这个预处理表加速模拟。每次对于每个特殊点只会经过一次（否则就是出现了死循环），每次查询的时间是$O(S)$。

所有预处理表的总时间是$O(nmp/S)$。总时间$O(qS + nmp/S)$。

取$S = O(\sqrt{nmp/q})$，总时间$O(\sqrt{pqnm})$。

注意这种算法并不依赖字符集是”<>”，也不依赖有一条边的长度很短的条件。对于这种算法，可以加入任意定义的字符而不影响复杂度，这是前两种算法所做不到的。

7 E - Tourists

7.1 题目大意

给出一个n点m边无向连通图，点上有权。

q次询问，每次可以修改一个点权，或者查询某两点间的所有简单路径的点权最小值的最小值。

$n, m, q \leq 10^5$。

7.2 算法0

指数时间的暴力就不说了。考虑这样的一个暴力：

设现在要从a走到b，我们枚举路径上权值最小的点c，现在的问题是确定是否存在一条从a到b的简单路径经过了c，这等价于，从c出发分别存在到a和b的两条不相交路径。

这个问题可以转化为网络流问题。用(u, v, w)表示从u到v，容量为c的有向弧。建立源S和汇T，连边$(S, c, 2), (a, T, 1), (b, T, 1)$；对于原图的每一条边$(u, v)$，连边$(u, v, 1), (v, u, 1)$。如果最大流是2，那么我们就从$c$找到了两条分别前往$a$和$b$的路径。

这样找到的两条路径可能会相交，所以还需要对除了a以外的点进行1的流量限制。这只需把每个点u拆成入点u_0和出点u_1，原来与u相连的边，根据方向分别改接到u_0或u_1上，然后连边$(u_0, u_1, 1)$就可以了。

这个算法每次询问的时间是$O(m)$，总时间$O(qm)$。

7.3 算法1

先考虑整个图都是点双连通图的情况。

引理1. *在点双连通图中，对于任意的点$a, b, c(a \neq b)$，总是存在从a到b经过c的简单路径。*

证明. 如果图的顶点数不超过2，结论显然正确。

对于至少有3个点的点双连通图，它一定也是边双连通图，所以删掉任何一个点或任何一条边，图依然是连通的。

考虑之前的网络流建图方式，欲证最大流=2，根据最大流最小割定理，只需证最小割=2，这等价于证不存在权值为1的割。而权值为1的边有以下三种：

对于$(a, T, 1), (b, T, 1)$这两条边，割掉以后图显然仍然连通。

对于形如$(u_0, u_1, 1)$的边，割掉这条边相当于将这个点的容量设为0，即相当于在原图上删掉这个点，根据之前的结论图依然连通。

对于形如$(u_1, v_0, 1)$的边，即使把这条边与$(v_1, u_0, 1)$一起割掉，也只等价于在原图上删掉这条边，图依然连通。

故不存在权值为1的割，所以最大流为2，证毕。

□

所以对于双连通图，我们只需要用数据结构维护一个（可重）集合，支持修改一个元素和查询最小元素，这是很容易做到的。

对于非双连通图，将它用Tarjan算法求出所有的割点和点双连通分量（以下简称“块”），把所有的割点和块抽象成顶点建一张新图，如果某个块包含了某个割点，就在它们之间连一条边，显然新图是一棵树。定义：

$$treenode(u) = \begin{cases} \text{树中}u\text{对应的顶点}, & u\text{是割点} \\ \text{树中}u\text{所在的块的顶点}, & u\text{不是割点} \end{cases}$$

引理2. *对于点$u, v(u \neq v)$，可能成为答案的点恰好是$treenode(u)$和$treenode(v)$的树上路径经过的所有点。*

证明. 先证明所有经过的点都可能成为答案：

对于路径上除了两个端点以外的点，证明是显然的；对于端点$treenode(u)$是割点的情况，证明也是显然的。

现在$treenode(u)$不是割点，设x是$treenode(u)$对应块的任何一个点：

如果u和v在同一个块中，根据引理1，存在从u到v经过x的路径；

如果u和v不在同一个块中，设树上路径的第二个点是割点c，根据引理1，存在从u到c经过x的路径，而这个路径显然不会与从c到v的路径相交。

然后证明没经过的点都不可能成为答案：

对于一个没经过的点x，它走到树上路径上必然会经过某个割点，故从u走到x之后不可能回到树上路径，即不可能走到v。

□

所以我们只需要维护树上路径点权最小值，每次询问直接查询，每次修改先把所在块的最小权值计算出来，再在树上修改这个块。但是如果一个割点被很多块包含，这种方法就会使每次操作的时间退化为线性。

一个比较简便的方法是：对于每个块只维护孩子割点，不维护父亲割点。这样对于割点的修改就只需要修改它的父亲块。然而，在查询树上路径时，如果发现LCA是块，就要把这个块的父亲割点也考虑到答案当中。

用轻重链剖分或Link-Cut Tree维护树上点权，总的时间复杂度为$O(n + m + q \log^2 n)$或$O(n + m + q \log n)$。