NOIP 模拟赛

2023 年 10 月 25 日

题目概况

题目名称	回文	快速排序	混乱邪恶	校门外歪脖树上的鸽子
题目类型	传统型	传统型	传统型	传统型
目录	palin	qsort	chaoticevil	pigeons
可执行文件名	palin	qsort	chaoticevil	pigeons
输入文件名	palin.in	qsort.in	chaoticevil.in	pigeons.in
输出文件名	palin.out	qsort.out	chaoticevil.out	pigeons.out
每个测试点时限	1s	3s	3s	6s
内存限制	512MB	512MB	512MB	512MB
测试点数目	20	25	25	25
每测试点分值	5	4	4	4
评测方式	全文比较	全文比较	Special Judge	全文比较

提交源程序文件名

对于 C++ 语言	palin.cpp	qsort.cpp	chaoticevil.cpp	pigeons.cpp

编译选项

对于 C++ 语言	-O2 -lm

注意事项

1. **建立题目文件夹，** 文件名（程序名和输入输出文件名）必须使用英文小写,。
2. C/C++ 中函数 main() 的返回值类型必须是 int，程序正常结束时的返回值必须是 0。
3. **因评测机较慢，后 3 题开的时限较长。**
4. 全文比较如无特殊说明，一律为过滤行末空格和文末回车。

回文（palin）

【题目描述】

给定一个 n 行 m 列的只包含小写字母的矩阵 A，请求出从 $(1,1)$ 到 (n,m) 只向下或向右走，且路径上的所有字符按照顺序排列可以构成一个回文串的路径条数。

由于答案可能很大，请输出答案在模 993244853 意义下的结果。

【输入格式】

从文件 ***palin.in*** 中读入数据。

第一行是两个正整数 n,m。

接下来 n 行，每行是一个长为 m 的字符串，其中只包含英文小写字母，描述矩阵 A 的内容。

【输出格式】

输出到 ***palin.out*** 中。

输出一行一个非负整数，表示满足条件的路径数模 993244853 后的值。

【输入样例 1】

见选手目录下的 ***palin/palin1.in***。

```
3 4
noip
ffff
pion
```

【输出样例 1】

见选手目录下的 ***palin/palin1.ans***。

```
2
```

【样例解释 1】

满足条件的路径为 $(1,1) \to (2,1) \to (2,2) \to (2,3) \to (2,4) \to (3,4)$ 和 $(1,1) \to (1,2) \to (2,2) \to (2,3) \to (3,3) \to (3,4)$。

【输入样例 2】

见选手目录下的 ***palin/palin2.in***。

```
4 5
wwwww
wwwww
wwwww
wwwwa
```

【输出样例 2】

见选手目录下的 ***palin/palin2.ans***。

```
0
```

【样例解释 2】

由于左上角和右下角的字符不同，任何路径上的字符都不可能构成回文串。

【输入样例 3】

见选手目录下的 ***palin/palin3.in***。

```
10 12
abbcbdbababa
bcccdcdccccb
bccccccccca
ccccdcdcdcdb
bdcdcccccccd
dccccccdcdb
bdcdcdcdcccc
acccccccccb
bccccdcdcccb
abababdbcbba
```

【输出样例 3】

见选手目录下的 ***palin/palin3.ans***。

```
20046
```

【样例 4】

见选手目录下的 ***palin/palin4.in*** 和 ***palin/palin4.ans***。

【数据规模与约定】

对于 20% 的数据，$1 \leq n, m \leq 10$。
对于 35% 的数据，$1 \leq n, m \leq 20$。
对于 50% 的数据，$1 \leq n, m \leq 80$。
对于另外 15% 的数据，$1 \leq n \leq 80$。
对于另外 15% 的数据，保证对任意 $i > 1, j < m$ 有 $A_{i,j} = A_{i-1,j+1}$。
对于 100% 的数据，$1 \leq n, m \leq 500$，输入的矩阵仅包含小写英文字母。

快速排序（qsort）

【题目描述】

小 A 在 E++ 课上学会了快速排序。

这天的模拟赛上，第一题是一道排序的板子。他写了一个快速排序的代码，很快通过了样例并交了上去。

然而，比赛结束前的两分钟，他意识到自己的快排忘记随机化了，并且他来不及修改了。

出成绩后，小 A 发现自己爆零了。

“按理说至少有暴力分啊，为什么会爆零啊……”

查看了一下测试数据，他发现数据损坏了，一些数字被替换为了 nan。

E++ 定义，nan 与 nan、nan 与任何整数、任何整数与 nan 比较的结果均为假。即：（以下为伪代码）

```
boolean operator < (Int x, Int y)
    if isnan(x) or isnan(y)
        then return false
        else return x<y
```

小 A 的快排代码实现是：

（以下为伪代码，选手目录下的 ***qsort/qsort_example.cpp*** 中有此伪代码的 C++ 实现，供选手参考使用。）

```
quicksort (Int[] A, int L, int R)
    if L >= R
        then return
    Int Left <- A[L]
    Int[] B
    int Mid = L
    int Bid = 0
    for int i L+1 to R by 1 do
        if A[i] < Left
            then A[Mid] <- A[i]
            Mid <- Mid + 1
            else B[Bid] <- A[i]
            Bid <- Bid + 1
    end
    for int i 0 to Bid - 1 by 1 do
        A[Mid + i + 1] <- B[i]
        A[Mid] <- Left
    quicksort (A, L, Mid - 1)
    quicksort (A, Mid + 1, R)
```

小 A 修好了数据，重测后得到了暴力分。但是他想知道自己的暴力在原来损坏的数据上的输出。

【输入格式】

从文件 ***qsort.in*** 中读入数据。

输入包含多组数据。

输入的第一行是一个正整数 T，表示数据组数。

接下来是 T 组数据的描述。

对于每组数据的描述，第一行是一个正整数 n，代表要排序的数的个数，接下来一行是 n 个字符串，其要么是 $[1, 10^9]$ 内的正整数，要么是字符串 `nan`。

【输出格式】

输出到 ***qsort.out*** 中。

对于每组数据，输出一行，包含 n 个字符串，每个字符串是一个正整数或 `nan`，代表小 A 的程序排序后的结果。

【输入样例 1】

见选手目录下的 ***qsort/qsort1.in***。

由于此测试点中第三组数据过长，此处略去。

```
2
4
2 4 1 3
7
nan 2 4 nan 1 nan 3
```

【输出样例 1】

见选手目录下的 ***qsort/qsort1.ans***。

由于此测试点中第三组数据过长，此处略去。

```
1 2 3 4
nan 1 2 3 4 nan nan
```

【样例 2】

见选手目录下的 ***qsort/qsort2.in*** 和 ***qsort/qsort2.ans***。

【样例 3】

见选手目录下的 ***qsort/qsort3.in*** 和 ***qsort/qsort3.ans***。

【样例 4】

见选手目录下的 ***qsort/qsort4.in*** 和 ***qsort/qsort4.ans***。

【数据规模与约定】

对于 12% 的数据，$1 \leq n \leq 10$。

对于 24% 的数据，$1 \leq n \leq 2000$。

对于另外 12% 的数据，输入中不包含 `nan`。

对于另外 24% 的数据，输入的每一个字符串有 50% 概率为 `nan`，有 50% 概率为一个 $[1, 10^9]$ 内均匀随机的正整数。

对于另外 12% 的数据，每组数据中的字符串只包含至多一种正整数。

对于 100% 的数据，$1 \leq T \leq 10$，$1 \leq n \leq 5 \times 10^5$，各组测试数据的 n 之和不超过 10^6。

混乱邪恶（chaoticevil）

【题目描述】

namespace_std这天打开了一本古老的算法书，上面描述了一个经典的 NP 问题：

给定一个集合，其元素为 $a_1, a_2, \ldots, a_n$ ，其中 $a_1 + a_2 + \ldots + a_n$ 为偶数，你需要找到一组 $c_1, c_2, \ldots, c_n \in \{-1, 1\}$，使得 $\sum a_i c_i = 0$。

namespace_std 发现这是一个很困难的问题，因此他想加一些限制。他限制 $\{a_1, a_2, \ldots, a_n\}$ 是 $\{1, 2, \ldots, m\}$ 的一个子集。

他发现这还是一个很困难的问题，因此他又限制 $\lfloor \frac{2m}{3} \rfloor < n \leq m$。

这个问题还是很困难，但他作为一名混乱邪恶的出题人，已经不想再弱化这个问题了……

因此，这个问题就交给你了。

你需要判断这个问题是否有解，如果有解则还要输出一组合法的 c_i。

【输入格式】

从文件 ***chaoticevil.in*** 读入数据。

输入两行。

第一行是两个正整数 n, m。

接下来一行是 n 个正整数 $a_1, a_2, \ldots, a_n$。

保证 $\{a_1, a_2, \ldots, a_n\}$ 是 $\{1, 2, \ldots, m\}$ 的一个子集，$a_1 + a_2 + \ldots + a_n$ 为偶数，且 $\lfloor \frac{2m}{3} \rfloor < n \leq m$。

【输出格式】

输出到 ***chaoticevil.out*** 中。

如果问题有解，请输出一行 `NP-Hard solved`，并在接下来一行输出 n 个 1 或 -1 ，表示你找到的解中的 c_i。如果有多种可行的 c_i，你只需要输出任意一种。

否则只需要输出一行 `Chaotic evil`。

【输入样例 1】

见选手目录下的 ***chaoticevil/chaoticevil1.in***。

```
6 8
1 6 2 5 4 8
```

【输出样例 1】

见选手目录下的 ***chaoticevil/chaoticevil1.ans***。

```
NP-Hard solved
1 -1 -1 -1 1 1
```

【样例解释 1】

$1-6-2-5+4+8=0$，因此 $c=\{1,-1,-1,-1,1,1\}$ 是一组合法的方案。

同理，$c=\{-1,1,1,1,-1,-1\}$ 也是一组合法的方案。

【样例 2】

见选手目录下的 ***chaoticevil/chaoticevil2.in*** 和 ***chaoticevil/chaoticevil2.ans***。

请注意，该样例的答案**并不唯一**。

【数据规模与约定】

对于 12% 的数据，$m\leq 20$。

对于 28% 的数据，$m\leq 300$。

对于 44% 的数据，$m\leq 1000$。

对于 56% 的数据，$m\leq 10^5$。

对于另外 8% 的数据，$n=m$。

对于另外 16% 的数据，保证 $n\geq m-5$。

对于 100% 的数据，$3\leq n\leq m\leq 10^6$，保证 $\{a_1,a_2,\ldots,a_n\}$ 是 $\{1,2,\ldots,m\}$ 的一个子集，$a_1+a_2+\ldots+a_n$ 为偶数，且 $\lfloor\frac{2m}{3}\rfloor<n\leq m$。

校门外歪脖树上的鸽子（pigeons）

【题目描述】

小 A 的校门口有一棵树。

这是一棵二叉树，由于长得歪歪扭扭，被人们称为歪脖树。

我们可以将歪脖树视为一棵二叉树，其中每个非叶节点恰有两个儿子，每个节点代表一个区间，并满足：

一个非叶子节点代表的区间是左儿子的区间和右儿子的区间的并，而 dfs 序第 i 小的叶子节点代表的是区间 $[i,i]$。

容易发现,对于任意一个区间 $[l,r]$,你可以用类似线段树的方式选取出若干个节点 $p_1,p_2,\ldots,p_k$，使得：

1. k 最小。

2. 这些节点代表的区间的并恰好是 $[l,r]$，且这些区间两两不交。

树上有一群鸽子，其中每个节点上恰好有一只。由于鸽子们经常听学校里的 OIer 讨论算法，已经学会了线段树。

小 A 偶然发现了这一点，便决定训练一下这群鸽子。但是由于鸽子非常咕，它们既懒得 pushup，也懒得 pushdown。但它们无意间学会了标记永久化，因此它们学会了在每个节点上维护一个标记。

小 A 很好奇现在让这群鸽子计算区间加区间求和会算出什么。具体地，他会尝试让鸽子完成两种操作：

1. 修改：给定区间 $[l,r]$ 和整数 d，对二叉树上按照上述方式选出的所有节点 p，执行 $s_p \leftarrow s_p + d \times (r_p - l_p + 1)$，其中 l_p 和 r_p 分别表示 p 点所代表区间的左端点和右端点。

2. 查询：给定区间 $[l,r]$，对二叉树上按照上述方式选出的所有节点 p 求 s_p 的和。

然而鸽子比他想象的要懒很多，并不愿意执行全部的操作。因此这个任务就交给你了。

【输入格式】

从文件 ***pigeons.in*** 中读入数据。

输入的第一行是两个正整数 n,m，分别表示二叉树的叶子节点个数和操作总个数。

接下来 $n-1$ 行，第 i 行是两个正整数，表示编号为 $n+i$ 的节点的两个儿子编号。

保证输入形成一棵二叉树，且树根的编号为 $1,2,\ldots,2n-1$ 这 $2n-1$ 个数中在这 $2n-2$ 个数中未出现的那个数（形式化地说，存在唯一的一个点 p，不存在任何点的儿子包含 p，那么 p 为树根）。

保证二叉树的所有叶子编号为 $1\sim n$，且若按照中序遍历访问这棵树，$1,2,\ldots,n$ 将依次被访问到。

接下来 m 行，每行包含三到四个正整数，表示一次操作。

第 i 行的第一个整数 op_i 表示此次操作的类型：

如果 $op_i=1$，则表示这是一个修改操作，接下来有三个正整数 l_i,r_i,d_i，代表一次修改，具体意义见【题目描述】。

如果 $op_i = 2$，则表示这是一个查询操作，接下来有两个正整数 l_i, r_i，代表一次查询，具体意义见【题目描述】。

【输出格式】

输出到 ***pigeons.out*** 中。

对于每一个 $op_i = 2$，你需要输出一行一个非负整数，代表此查询操作的答案。

【输入样例 1】

见选手目录下的 ***pigeons/pigeons1.in***。

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

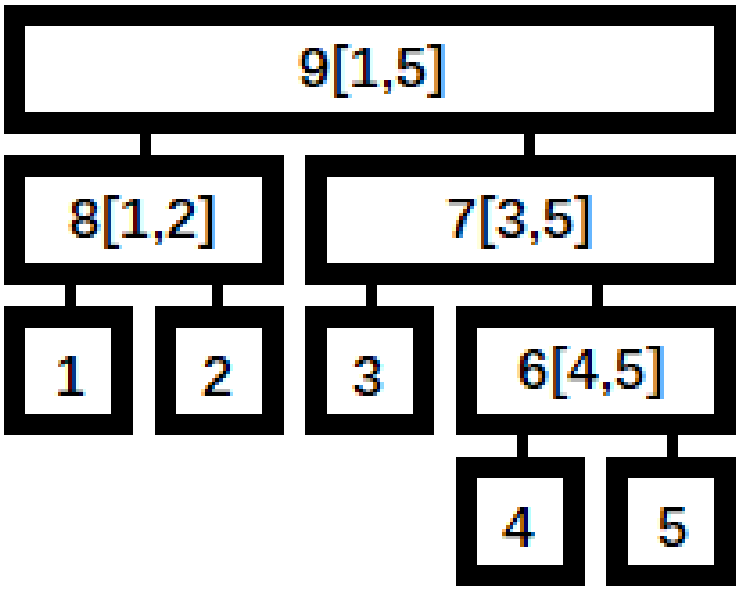
【输出样例 1】

见选手目录下的 ***pigeons/pigeons1.ans***。

```
0
5
3
9
```

【样例解释 1】

此样例对应的二叉树形态如下图所示：



以下是具体的操作内容。
操作 1：将 s_9 加上 5。
操作 2：询问 s_2 和 s_3 的和，回答 0。
操作 3：询问 s_9 的值，回答 5。
操作 4：将 s_2 加上 3，将 s_7 加上 9。
操作 5：询问 s_2、s_3 和 s_4 的和，回答 3。
操作 6：询问 s_5 的值，回答 9。

【样例 2】

见选手目录下的 ***pigeons/pigeons2.in*** 和 ***pigeons/pigeons2.ans***。
该样例满足测试点 7 的性质。

【样例 3】

见选手目录下的 ***pigeons/pigeons3.in*** 和 ***pigeons/pigeons3.ans***。
该样例满足测试点 17 的性质。

【样例 4】

见选手目录下的 ***pigeons/pigeons4.in*** 和 ***pigeons/pigeons4.ans***。
该样例满足测试点 22 的性质。

【数据规模与约定】

对于 100% 的数据，$2 \leq n \leq 2 \times 10^5$，$1 \leq m \leq 2 \times 10^5$，$1 \leq l \leq r \leq n$，$1 \leq d \leq 10^8$。
具体的数据规模与约定见下表。

测试点编号	n	m	特殊约定
1	≤ 20	≤ 20	无
2			
3	≤ 100	≤ 100	树高为 n
4			无
5	$\leq 10^3$	$\leq 10^3$	树高为 n
6			所有查询在修改之后
7			无
8			
9		$\leq 2 \times 10^5$	所有查询在修改之后
10			对于所有修改和查询，$l = 1$ 或 $r = n$
11			无
12			
13	$\leq 2 \times 10^5$	$\leq 10^3$	树高为 n
14			对于所有修改和查询，$l = 1$ 或 $r = n$
15			无
16			
17	$\leq 5 \times 10^4$	$\leq 5 \times 10^4$	树高为 n
18			所有查询在修改之后
19			对于所有修改和查询，$l = 1$ 或 $r = n$
20			无
21	$\leq 2 \times 10^5$	$\leq 2 \times 10^5$	树高为 n
22			所有查询在修改之后
23			对于所有修改和查询，$l = 1$ 或 $r = n$
24			无
25			