

# 提高组3

中文题目名称	回文	快速排序	混乱邪恶	校门外歪脖树上的鸽子
英文题目名称	palin	qsort	chaoticevil	pigeons
每个测试点建议时限	1500	3000	4000	3000
每个测试点空间限制	128 M	128 M	128 M	512 M
测试点数目	20	25	25	25
每个测试点分值	5	4	4	4
比较方式	逐行比较	逐行比较	特判程序	逐行比较
浮点输出误差精度	-	-	-	-

## 注意：

- 英文题目名称即文件名，若文件名为 filename，则提交的文件为filename.pas/c/cpp，程序输入输出文件名分别为 filename.in filename.out。
- 建议时限仅供参考，具体按照评测机上标程运行时间的2 - 3倍设置。
- 建议将栈大小设为64m，并打开编译参数O2。

# 回文

## 题目限制

1500 ms 128 M

## 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

第一行输入两个正整数  $n, m$ 。（ $1 \leq n, m \leq 500$ ）  
之后  $n$  行，每行输入一个长为  $m$  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

输出一行一个非负整数，表示满足条件的路径数模 993244853 后的值。

## 数据范围

- 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， $1 \leq n, m \leq 80$  。

另有 15% 的数据，  $1 \leq n \leq 80$  。

另有 15% 的数据，保证对任意  $1 \leq i, j$

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。

输入样例

```
输入样例1:
3 4
noip
ffff
pion
输入样例2:
4 5
wwwww
wwwww
wwwww
wwwa
输入样例3:
10 12
abbcdbababa
bcccdcdcccb
bccccccccca
ccccdcdcdcb
bdcdccccccd
dcccccdcdcb
bdcdcdcdcccc
acccccccccb
bcccdcdcccb
abababdbcbba
```

输出样例

```
输出样例1:
2
输出样例2:
0
输出样例3:
20046
```

样例解释

样例1：满足条件的路径为  $(1^*, 1) \rightarrow (2, 1) \rightarrow (2, 2) \rightarrow (2, 3) \rightarrow (2, 4) \rightarrow (3, 4)$  和  $(1, 1) \rightarrow (1, 2) \rightarrow (2, 2) \rightarrow (2, 3) \rightarrow (3, 3) \rightarrow (3, 4)$ 。

样例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。

# 快速排序

题目限制

3000 ms 128 M

题目描述

namespace\_std 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，namespace\_std 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）

```
1  boolean operator < (Int x, Int y)
2      if isnan(x) or isnan(y)
3          then return false
4      else return x<y
5
```

namespace\_std 的快排代码实现是：（以下为伪代码）

```
1  quicksort (Int[] A, int L, int R)
2      if L >= R
3          then return
4      Int Left <- A[L]
5      Int[] B
6      int Mid = L
7      int Bid = 0
8      for int i L+1 to R by 1 do
9          if A[i] < Left
10             then A[Mid] <- A[i]
11             Mid <- Mid + 1
12          else B[Bid] <- A[i]
13             Bid <- Bid + 1
14      end
15      for int i 0 to Bid - 1 by 1 do
16          A[Mid + i + 1] <- B[i]
17          A[Mid] <- Left
18      quicksort (A, L, Mid - 1)
19      quicksort (A, Mid + 1, R)
20
```

namespace\_std 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

输入包含多组数据。第一行输入一个正整数 T，表示数据组数。

之后对于每组数据，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 n 个字符串，其要么是 [1, 1e9] 内的正整数，要么是字符串 nan。

其中  $1 \leq T \leq 10$ ,  $1 \leq n \leq 5e5$ ,  $\sum n \leq 1e6$ 。

## 输出格式

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

## 数据范围

对于 12% 的数据， $1 \leq n \leq 10$ 。

对于 24% 的数据， $1 \leq n \leq 2000$ 。

另有 12% 的数据，输入中不包含 `nan`。

另有 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。

另有 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  $10^6$ 。

## 输入样例

```
3
4
2 4 1 3
7
nan 2 4 nan 1 nan 3
22
1 nan 1 nan 4 nan nan nan nan 5 nan nan nan nan nan 1 nan 4 nan nan nan nan
```

## 输出样例

```
1 2 3 4
nan 1 2 3 4 nan nan
1 nan 1 nan 1 4 nan nan nan nan 4 5 nan nan nan nan nan nan nan nan nan
```

# 混乱邪恶

## 题目限制

4000 ms 128 M

## 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  $\{a[1], a[2], \dots, a[n]\}$ ，其中  $a[1] + a[2] + \dots + a[n]$  为偶数，你需要找到一组  $c[1], c[2], \dots, c[n] \in \{-1, 1\}$ ，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  $\{a[1], a[2], \dots, a[n]\}$  是  $\{1, 2, \dots, m\}$  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

第一行输入两个正整数  $n, m$ 。  
第二行输入  $n$  个正整数  $a[1], a[2], \dots, a[n]$ 。  
保证  $\{a[1], a[2], \dots, a[n]\}$  是  $\{1, 2, \dots, m\}$  的一个子集,  $a[1]+a[2]+\dots+a[n]$  为偶数, 且  $\lfloor 2m/3 \rfloor < n \leq m$ 。

## 输出格式

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"。

## 数据范围

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,  $m \leq 10^5$ 。  
另有 8% 的数据,  $n = m$ 。  
另有 16% 的数据, 保证  $n \leq m - 5$ 。  
对于 100% 的数据,  $3 \leq n \leq m \leq 10^6$ , 保证  $\{a[1], a[2], \dots, a[n]\}$  是  $\{1, 2, \dots, m\}$  的一个子集,  $a[1] + a[2] + \dots + a[n]$  为偶数, 且  $\lfloor \frac{2m}{3} \rfloor < n \leq m$ 。

## 输入样例

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

## 输出样例

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

## 样例解释

$1-6-2-5+4+8=0$ , 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。

# 校门外歪脖树上的鸽子

## 题目限制

3000 ms 512 M

## 题目描述

namespace\_std 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  $i$  有代表一个区间  $[l_i, r_i]$ , 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  $j$  小的叶子节点代表的是区间  $[j, j]$ 。

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  $p_1, p_2, \dots, p_k$ ，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 Oler 讨论算法，已经学会了线段树。

namespace\_std 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。

namespace\_std 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 l r d: 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  $s_p$  表示  $p$  点的权值，**假定所有点初始值为0**。

2 l r: 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

第一行输入两个正整数  $n, m$ ，分别表示二叉树的叶子节点个数和操作总个数。（ $2 \leq n \leq 2e5, 1 \leq m \leq 2e5$ ）

之后  $n-1$  行，其中第  $i$  行输入两个正整数，表示编号  $n+i$  的节点的两个儿子编号。保证输入形成一棵二叉树，且未出现为儿子的点为树根。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， $1, 2, \dots, n$  将依次被访问到。

之后  $m$  行，每行描述一次操作。

## 输出格式

对于每一个操作2，输出一行一个非负整数，表示此查询操作的答案。

## 数据范围

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。

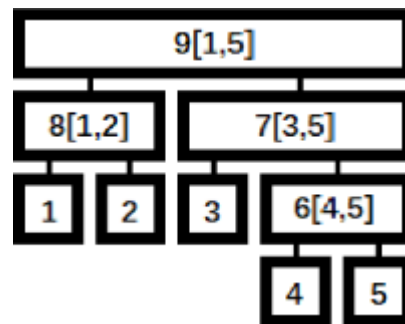
## 输入样例

```
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

0  
5  
3  
9

### 样例解释



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。