2026 全国青少年信息学奥林匹克竞赛

CCF NOI 2026

第二试

时间：2026 年 7 月 22 日 08:00 ～ 13:00

题目名称	中位数	木棉	彩虹树
题目类型	传统型	传统型	传统型
目录	median	kapok	rainbow
每个测试点时限	1.0 秒	2.0 秒	1.0 秒
内存限制	1024 MiB	1024 MiB	1024 MiB
测试点数目	20	25	25
测试点是否等分	是	是	是
预测试点数目	20	25	25

提交程序源文件名

对于 C++ 语言	median.cpp	kapok.cpp	rainbow.cpp

编译选项

对于 C++ 语言	-O2 -std=c++14 -static

注意事项（请仔细阅读）

1. 赛后正式测试时将以选手留在题目目录下的程序源文件为准。对程序源文件以外文件的修改不会影响评测结果。
2. 选手提交的程序源文件大小不得超过 100 KiB。
3. 程序可使用的栈空间内存限制与题目的内存限制一致。
4. 禁止在提交的程序源文件中改变编译器参数（如使用 `#pragma` 命令），禁止使用系统结构相关指令（如内联汇编）或其他可能造成不公平的方法。
5. 因违反上述规定而出现的问题，申诉时一律不予受理。
6. 选手可使用快捷启动页面中的工具 selfEval 进行自测。选手需将待测程序的源文件置于相应题目目录下。每次自测时可选择全部或部分题目进行自测。注意：自测有次数限制，且自测结果仅供选手测试参考，不作为最终正式成绩。

中位数（median）

【题目描述】

对于大小为 m 的可重集 $S = \{x_0, x_1, \ldots, x_{m-1}\}$，设其所有元素从大到小排序后的结果为 $y_0 \geq y_1 \geq \cdots \geq y_{m-1}$。定义其中位数 $\mathrm{Median}(S)$ 为其中第 $\lceil m/2 \rceil$ 大的数，即 $\mathrm{Median}(S) = y_{\lceil m/2 \rceil - 1}$。**注意：本题中的中位数定义与常规定义可能有所不同。**

给定长度为 n 的序列 $[a_0, a_1, \ldots, a_{n-1}]$，以及一个正整数 k $(k \leq n)$。定义一个划分如下：选择一个长度为 $k-1$ 的递增下标序列 $0 < b_1 < \cdots < b_{k-1} < n$，即可将原序列划分为 k 个段，对应的下标区间依次为 $[0, b_1), [b_1, b_2), \ldots, [b_{k-1}, n)$。

对于一个划分，定义该划分的平衡度如下：对于划分出的每个段，计算其中所有元素形成的可重集的中位数，则这 k 个中位数形成的可重集的中位数即为该划分的平衡度。形式化地，对于划分 $b_1, \ldots, b_{k-1}$，记 $b_0 = 0$，$b_k = n$，设第 i $(0 \leq i < k)$ 个段中所有元素形成的可重集的中位数为 $c_i = \mathrm{Median}(\{a_{b_i}, a_{b_i+1}, \ldots, a_{b_{i+1}-1}\})$，则该划分的平衡度为 $\mathrm{Median}(\{c_0, \ldots, c_{k-1}\})$。

请求出所有划分中平衡度的最大值。

【实现细节】

选手不需要，也不应该实现 `main` 函数。

选手需要确保提交的程序源文件包含头文件 `median.h`，即在程序开头加入以下代码：

```
#include "median.h"
```

选手需要在提交的程序源文件 `median.cpp` 中实现以下两个函数：

```
void init(int c, int t);
```

- c, t 分别表示测试点编号与测试数据组数。$c = 0$ 表示该测试点为样例。
- 对于每个测试点，该函数会在程序开始运行时被评测程序调用恰好一次。

```
int median(int n, int k, std::vector<int> a);
```

- n, k, a 分别表示序列长度、划分的段数与给定的序列。
- 该函数需要返回平衡度的最大值。
- 对于每个测试点，该函数会被评测程序调用恰好 t 次。

本试题目录下的 `template_median.cpp` 是提供的示例代码，选手可参考并实现自己的代码。

【测试程序方式】

选手可以在本题目录下使用如下命令编译得到可执行文件：

```
g++ grader.cpp median.cpp -o median -O2 -std=c++14 -static
```

对于编译得到的可执行文件 median：

- 可执行文件将从标准输入读入以下格式的数据：
  - 第一行包含两个非负整数 c, t。
  - 接下来依次为每组测试数据，对于每组测试数据：
    - 第一行包含两个正整数 n, k。
    - 第二行包含 n 个正整数 $a_0, a_1, \ldots, a_{n-1}$。
- 可执行文件将输出以下格式的数据至标准输出：
  - 对于每组测试数据，输出一行一个正整数，表示平衡度的最大值。

【样例 1 输入】

```
0 2
10 4
6 5 1 9 2 3 10 7 4 8
10 5
5 7 3 10 8 2 9 1 6 4
```

【样例 1 输出】

```
9
8
```

【样例 1 解释】

对于第一组测试数据，一种平衡度最大的划分为 $b_1 = 3$，$b_2 = 5$，$b_3 = 7$，其将原序列划分为 4 个段 $[6, 5, 1], [9, 2], [3, 10], [7, 4, 8]$，段中所有元素的中位数分别为 $5, 9, 10, 7$，因此该划分的平衡度为 $\text{Median}(\{5, 9, 10, 7\}) = 9$。

对于第二组测试数据，一种平衡度最大的划分为 $b_1 = 2$，$b_2 = 4$，$b_3 = 6$，$b_4 = 8$，其将原序列划分为 5 个段 $[5, 7], [3, 10], [8, 2], [9, 1], [6, 4]$，段中所有元素的中位数分别为 $7, 10, 8, 9, 6$，因此该划分的平衡度为 $\text{Median}(\{7, 10, 8, 9, 6\}) = 8$。

【样例 2】

见选手目录下的 ***median/median2.in*** 与 ***median/median2.ans***。

该样例满足测试点 6 的约束条件。

【样例 3】

见选手目录下的 ***median/median3.in*** 与 ***median/median3.ans***。
该样例满足测试点 9 的约束条件。

【样例 4】

见选手目录下的 ***median/median4.in*** 与 ***median/median4.ans***。
该样例满足测试点 14 的约束条件。

【样例 5】

见选手目录下的 ***median/median5.in*** 与 ***median/median5.ans***。
该样例满足测试点 18 ∼ 20 的约束条件。

【数据范围】

设 N 为单个测试点内所有测试数据的 n 的和。对于所有测试数据，均有：

- $1 \le t \le 20$；
- $5 \le n \le 10^6$，$2 \le k \le n$，$N \le 10^6$；
- 对于所有 $0 \le i < n$，均有 $1 \le a_i \le n$。

测试点编号	$N \le$	$n \le$	k	特殊性质
$1, 2$	40	20	$\le n$	无
$3 \sim 5$	800	80		A
6				无
$7, 8$	$8,000$	800		A
9				无
10	2×10^5	2×10^5	$= 2$	
11			$= 3$	
$12, 13$			$= 5$	
14			≤ 10	
15			$\equiv 0 \pmod 2$	
$16, 17$	10^6	10^6	> 5	
$18 \sim 20$			$\le n$	

特殊性质 A：对于所有 $0 \le i < n$，均有 $a_i \le 2$。

木棉（kapok）

【题目描述】

故园中的那几棵木棉，仍旧生长在小 N 逐渐朦胧的记忆中。小 N 对故园的记忆，可以用一个长度为 n 的序列 $[a_0, a_1, \ldots, a_{n-1}]$ 表示。

故园中的每棵木棉，都形如一棵结点有标号的无根树。小 N 对一棵木棉的印象，可以用她的故园记忆的一个区间 $[l, r)$ 描述：

- 这棵树的结点数目为 $k = r - l + 2$，结点编号为 $0 \sim k - 1$；
- $[\min(a_l, k-1), \min(a_{l+1}, k-1), \ldots, \min(a_{r-1}, k-1)]$ 是这棵树的 Prüfer 序列，其中 Prüfer 序列的定义详见【提示】一节。

在回忆往事时，小 N 也向你提出了 m 次询问。其中第 i $(0 \le i < m)$ 次询问为：

- 在故园记忆的区间 $[l_i, r_i)$ 所对应的木棉上，结点 x_i, y_i 是否相邻？

【实现细节】

选手不需要，也不应该实现 `main` 函数。

选手需要确保提交的程序源文件包含头文件 `kapok.h`，即在程序开头加入以下代码：

```
#include "kapok.h"
```

选手需要在提交的程序源文件 `kapok.cpp` 中实现以下函数：

```
std::vector<bool> kapok(int c, int n, int m,
  std::vector<int> a, std::vector<int> l, std::vector<int>
  r, std::vector<int> x, std::vector<int> y);
```

- c, n, m 分别表示测试点编号、故园记忆序列的长度、询问次数，$c = 0$ 表示该测试点为样例。
- a 表示故园记忆序列。
- l, r 分别表示每次询问所给定的区间的两个端点。
- x, y 分别表示每次询问所给定的两个结点的编号。
- 该函数需要返回一个长度恰好为 m 的序列 $f_0, f_1, \ldots, f_{m-1}$，其中 f_i $(0 \le i < m)$ 表示第 i 次询问的结果。
- 对于每个测试点，该函数会被评测程序调用恰好一次。

本试题目录下的 `template_kapok.cpp` 是提供的示例代码，选手可参考并实现自己的代码。

【测试程序方式】

选手可以在本题目录下使用如下命令编译得到可执行文件：

```
g++ grader.cpp kapok.cpp -o kapok -O2 -std=c++14 -static
```

对于编译得到的可执行文件 kapok：

- 可执行文件将从标准输入读入以下格式的数据：
  - 第一行包含三个非负整数 c, n, m。
  - 第二行包含 n 个非负整数 $a_0, a_1, \ldots, a_{n-1}$。
  - 第 $i+3$ $(0 \le i < m)$ 行包含四个非负整数 l_i, r_i, x_i, y_i。
- 可执行文件将输出以下格式的数据至标准输出：
  - 第 $i+1$ $(0 \le i < m)$ 行包含一个非负整数，其中 0 表示 f_i 为 false，1 表示 f_i 为 true。

【样例 1 输入】

```
0 8 3
2 0 2 6 0 7 2 2
0 3 0 2
3 5 1 2
0 0 0 1
```

【样例 1 输出】

```
1
0
1
```

【样例 1 解释】

- 区间 $[0,3)$ 对应的树有 5 个结点，Prüfer 序列为 $[2,0,2]$，边集为 $\{(1,2),(0,3),(0,2),(2,4)\}$，因此结点 $0,2$ 相邻。
- 区间 $[3,5)$ 对应的树有 4 个结点，Prüfer 序列为 $[3,0]$，边集为 $\{(1,3),(0,2),(0,3)\}$，因此结点 $1,2$ 不相邻。
- 区间 $[0,0)$ 对应的树有 2 个结点，Prüfer 序列为空，唯一一条边为 $(0,1)$，因此结点 $0,1$ 相邻。

【样例 2】

见选手目录下的 ***kapok/kapok2.in*** 与 ***kapok/kapok2.ans***。
该样例满足测试点 $3 \sim 5$ 的约束条件。

【样例 3】

见选手目录下的 ***kapok/kapok3.in*** 与 ***kapok/kapok3.ans***。
该样例满足测试点 $3 \sim 5$ 的约束条件。

【数据范围】

对于所有测试数据，均有：

- $1 \le n, m \le 2 \times 10^5$；
- 对于所有 $0 \le i < n$，均有 $0 \le a_i < n + 2$；
- 对于所有 $0 \le i < m$，均有 $0 \le l_i \le r_i \le n$ 且 $0 \le x_i, y_i < r_i - l_i + 2$。

测试点编号	$n, m \le$	特殊性质
$1, 2$	500	无
$3 \sim 5$	$5,000$	
$6, 7$	2×10^5	对于所有 $0 \le i < n$，均有 $a_i < 10$
$8, 9$		对于所有 $0 \le i < n$，均有 $a_i < 10^3$
$10, 11$		对于所有 $0 \le i < m$，均有 $x_i, y_i < 10$
$12, 13$		对于所有 $0 \le i < m$，均有 $x_i, y_i < 10^3$
$14 \sim 16$		A
$17 \sim 19$		对于所有 $0 \le i < m$，均有 $r_i = n$
$20 \sim 22$	10^5	无
$23 \sim 25$	2×10^5	

- 特殊性质 A：对于所有 $0 \le i < m$，在给定 l_i, r_i 后，x_i, y_i 均从所有满足限制的有序点对中独立均匀随机生成。

【提示】

对于一棵结点编号为 $0 \sim c - 1$ $(c \ge 2)$ 的无根树 T：

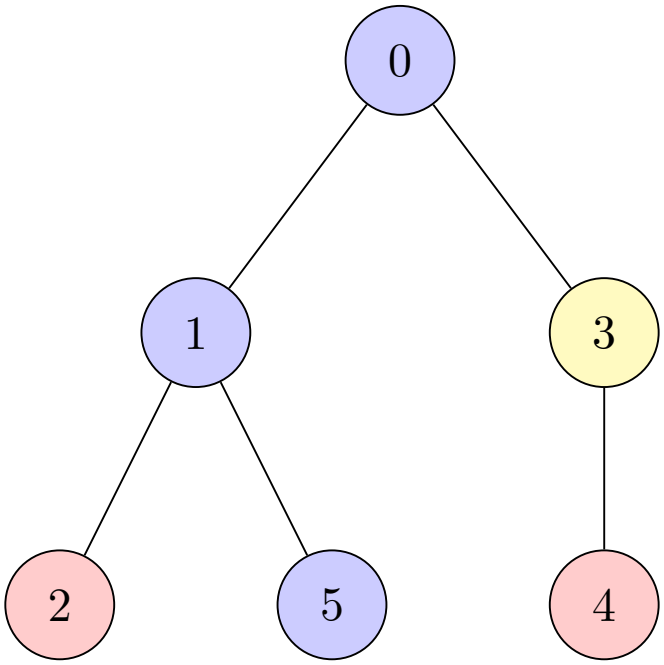
- 依次执行 $c - 2$ 次操作，其中第 i $(0 \le i < c - 2)$ 次操作如下：
  - 找出当前编号最小的叶子 v_i；
  - 记录其唯一相邻点的编号 p_i；
  - 然后从 T 中删去 v_i 及其唯一邻边。
- 如此得到的长度为 $c - 2$ 的序列 $[p_0, p_1, \ldots, p_{c-3}]$，即为 T 的 Prüfer 序列。

彩虹树（rainbow）

【题目描述】

传说在夜之国中，有一棵包含 n 个结点的彩虹树。彩虹树是一棵有根树，结点编号为 $0 \sim n-1$，其中结点 0 是彩虹树的树根，结点 i $(1 \le i < n)$ 的父亲结点为 f_i。

彩虹树上的每个结点都能显现出任意颜色。颜色共有无穷多种，但彩虹树的**缤纷度**只由每棵子树中出现过的**不同颜色的数量**决定，而与具体颜色无关。具体地，设以结点 i $(0 \le i < n)$ 为根的子树中共有 c_i 种不同的颜色的结点，则彩虹树的缤纷度可以用序列 $[c_0, c_1, \ldots, c_{n-1}]$ 表示。例如，在下图中，结点 $0, 1, 5$ 为蓝色，结点 $2, 4$ 为红色，结点 3 为黄色，因此彩虹树的缤纷度为 $[3, 2, 1, 2, 1, 1]$。



云层的变化会导致彩虹树的色彩显现出某种特定的规律。具体地，每种规律可以由一个对应的结点子集 $S \subseteq \{1, 2, \ldots, n-1\}$ 来描述：集合 S 中的每个结点 u 的颜色都与某个祖先颜色相同。形式化地，对于所有 $u \in S$，均存在 u 的祖先 p $(p \ne u)$，满足 u 与 p 颜色相同。尽管受到上述规律的制约，彩虹树仍能呈现出不同的色彩，形成不同的缤纷度。记满足规律 S 的彩虹树的缤纷度的**种类数**为 w_S，其中两种缤纷度不同当且仅当两个序列中至少有一个元素不同。

请计算，对于所有可能的 2^{n-1} 种规律，满足每种规律的彩虹树的缤纷度的种类数之和，即 $\sum_{S \subseteq \{1,2,\ldots,n-1\}} w_S$。由于答案可能较大，只需求出答案对 $998,244,353$ 取模后的结果。

【实现细节】

选手不需要，也不应该实现 main 函数。

选手需要确保提交的程序源文件包含头文件 rainbow.h，即在程序开头加入以下代码：

```
#include "rainbow.h"
```

选手需要在提交的程序源文件 rainbow.cpp 中实现以下函数：

```
int rainbow(int c, int n, std::vector<int> f);
```

- c, n 分别表示测试点编号与彩虹树的结点数。$c = 0$ 表示该测试点为样例。
- f 是一个长度为 n 的序列，其中 $f_0 = 0$，f_i $(1 \le i < n)$ 表示结点 i 的父亲结点。
- 该函数需要返回满足每种规律的彩虹树的缤纷度的种类数之和对 $998,244,353$ 取模后的结果。
- 对于每个测试点，该函数会被评测程序调用恰好一次。

本试题目录下的 template_rainbow.cpp 是提供的示例代码，选手可参考并实现自己的代码。

【测试程序方式】

选手可以在本题目录下使用如下命令编译得到可执行文件：

```
g++ grader.cpp rainbow.cpp -o rainbow -O2 -std=c++14 -static
```

对于编译得到的可执行文件 rainbow：

- 可执行文件将从标准输入读入以下格式的数据：
  - 第一行包含两个非负整数 c, n。
  - 第二行包含 $n-1$ 个非负整数 $f_1, f_2, \ldots, f_{n-1}$。
- 可执行文件将输出以下格式的数据至标准输出：
  - 输出一行一个非负整数表示 rainbow 函数的返回值。

【样例 1 输入】

```
0 3
0 0
```

【样例 1 输出】

```
8
```

【样例 1 解释】

- 满足规律 $S=\varnothing$ 的彩虹树的缤纷度共有 $[1,1,1],[2,1,1],[3,1,1]$ 三种；
- 满足规律 $S=\{1\}$ 的彩虹树的缤纷度共有 $[1,1,1],[2,1,1]$ 两种；
- 满足规律 $S=\{2\}$ 的彩虹树的缤纷度共有 $[1,1,1],[2,1,1]$ 两种；
- 满足规律 $S=\{1,2\}$ 的彩虹树的缤纷度有 $[1,1,1]$ 一种。

因此答案为 $3+2+2+1=8$。

【样例 2 输入】

```
0 6
0 1 0 3 1
```

【样例 2 输出】

```
279
```

【样例 3】

见选手目录下的 ***rainbow/rainbow3.in*** 与 ***rainbow/rainbow3.ans***。
该样例满足测试点 $3,4$ 的约束条件。

【样例 4】

见选手目录下的 ***rainbow/rainbow4.in*** 与 ***rainbow/rainbow4.ans***。
该样例满足测试点 $5\sim7$ 的约束条件。

【样例 5】

见选手目录下的 ***rainbow/rainbow5.in*** 与 ***rainbow/rainbow5.ans***。
该样例满足测试点 $8,9$ 的约束条件。

【样例 6】

见选手目录下的 ***rainbow/rainbow6.in*** 与 ***rainbow/rainbow6.ans***。
该样例满足测试点 $10\sim12$ 的约束条件。

【数据范围】

对于所有测试数据，均有：

- $1 \le n \le 200$；
- 对于所有 $1 \le i < n$，均有 $0 \le f_i < i$。

测试点编号	$n \le$	特殊性质
1	4	无
2	8	
3, 4	16	
5 ∼ 7	50	
8, 9	10^2	B
10 ∼ 12		无
13 ∼ 15	150	
16	200	A
17 ∼ 19		B
20 ∼ 25		无

特殊性质 A：对于所有 $1 \le i < n$，均有 $f_i = i - 1$。

特殊性质 B：对于所有 $0 \le i < n$，至多存在两个 j 满足 $f_j = i$。