2026 全国青少年信息学奥林匹克竞赛

CCF NOI 2026

第一试

时间：2026 年 7 月 20 日 08:00 ～ 13:00

题目名称	线段	传送	布丁
题目类型	传统型	传统型	交互型
目录	segment	teleport	pudding
每个测试点时限	2.5 秒	3.5 秒	10.0 秒
内存限制	512 MiB	1024 MiB	1024 MiB
测试点数目	25	20	3
测试点是否等分	是	是	否
预测试点数目	25	20	3

提交程序源文件名

对于 C++ 语言	segment.cpp	teleport.cpp	pudding.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。

线段（segment）

【题目描述】

小 L 有 n 条包含于 $[1,m]$ 的线段，其中第 i $(0 \le i < n)$ 条为 $[l_i, r_i]$ $(1 \le l_i \le r_i \le m)$。

小 L 认为过于复杂的线段相交关系不够优美，于是对于每一个线段的集合 $S \subseteq \{0,1,\ldots,n-1\}$，小 L 定义 S 是**优美的**，如果满足如下要求：

- 构造一个顶点集合对应 S 的图，其中顶点 u 与顶点 v 间存在一条边，当且仅当线段 u 与线段 v 相交，即存在 $x \in [1,m]$，满足 $l_u \le x \le r_u$ 且 $l_v \le x \le r_v$。称 S 是**优美的**，当且仅当构造出的图恰好为一棵树。

小 L 想知道有多少线段集合是优美的，因此他给定了一个正整数 k $(k \le n)$，你需要计算，对于每个 $s = 1,2,\ldots,k$，有多少个大小为 s 的集合是优美的。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 `segment.h`，即在程序开头加入以下代码：

```
#include "segment.h"
```

选手需要在提交的程序源文件 `segment.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。

```
std::vector<int> segment(int n, int m, int k,
  std::vector<int> l, std::vector<int> r);
```

- n, m, k 分别表示线段的数量、坐标范围上限及需要计算的集合的大小上限。
- l, r 分别表示每条线段的左端点与右端点。
- 该函数需要返回一个长度**恰好**为 $k+1$ 的序列 a，其中 $a_0 = 0$，a_s $(1 \le s \le k)$ 表示大小为 s 的优美的集合数量对 $998,244,353$ 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 t 次。

本试题目录下的 `template_segment.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 segment.cpp -o segment -O2 -std=c++14 -static
```

对于编译得到的可执行文件 segment：

- 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：
    - 第一行包含三个正整数 n, m, k。
    - 第 $i+2$ $(0 \le i < n)$ 行包含两个正整数 l_i, r_i。
- 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 k 个非负整数 $a_1, a_2, \ldots, a_k$。

【样例 1 输入】

```
0 3
3 3 3
1 2
2 3
1 3
4 5 4
1 2
2 3
3 4
4 5
4 2 3
1 2
1 2
1 2
1 1
```

【样例 1 输出】

```
3 3 0
4 3 2 1
4 6 0
```

【样例 1 解释】

对于第一组测试数据，

- 大小为 1 的集合有 $\{0\},\{1\},\{2\}$，均是优美的。
- 大小为 2 的集合有 $\{0,1\},\{1,2\},\{0,2\}$，均是优美的。
- 大小为 3 的集合有 $\{0,1,2\}$，构造出的图是一个三元环，不是优美的。

因此答案分别为 $3,3,0$。

对于第二组测试数据。

- 大小为 1 的集合中，所有 4 个集合均是优美的。
- 大小为 2 的集合中，$\{0,1\},\{1,2\},\{2,3\}$ 是优美的。
- 大小为 3 的集合中，$\{0,1,2\},\{1,2,3\}$ 是优美的。
- 大小为 4 的集合 $\{0,1,2,3\}$ 是优美的。

因此答案分别为 $4,3,2,1$。

【样例 2】

见选手目录下的 ***segment/segment2.in*** 与 ***segment/segment2.ans***。

该样例满足测试点 $6 \sim 8$ 的约束条件。

【样例 3】

见选手目录下的 ***segment/segment3.in*** 与 ***segment/segment3.ans***。

该样例满足测试点 $9,10$ 的约束条件。

【样例 4】

见选手目录下的 ***segment/segment4.in*** 与 ***segment/segment4.ans***。

该样例满足测试点 $11 \sim 15$ 的约束条件。

【样例 5】

见选手目录下的 ***segment/segment5.in*** 与 ***segment/segment5.ans***。

该样例满足测试点 $16 \sim 18$ 的约束条件。

【样例 6】

见选手目录下的 ***segment/segment6.in*** 与 ***segment/segment6.ans***。

该样例满足测试点 $22,23$ 的约束条件。

【样例 7】

见选手目录下的 ***segment/segment7.in*** 与 ***segment/segment7.ans***。
该样例满足测试点 24, 25 的约束条件。

【数据范围】

设 K 为单个测试点内所有测试数据的 k 的和。对于所有测试数据，均有：

- $1 \le t \le 20$；
- $1 \le n \le 3,000$，$1 \le m \le 10^3$，$1 \le k \le n$，$K \le 200$；
- 对于所有 $0 \le i < n$，均有 $1 \le l_i \le r_i \le m$。

测试点编号	$n \le$	$m \le$	$K \le$	$k \le$	特殊性质
$1 \sim 3$	20	10^2	20	20	无
$4, 5$	$3,000$	10^3	200	2	
$6 \sim 8$				3	
$9, 10$	500			200	A
$11 \sim 15$	$3,000$				B
$16 \sim 18$	200	500	50	50	C
$19 \sim 21$	500	10^3	200	200	
$22, 23$	10^3	10^2	30	30	无
$24, 25$	$3,000$	10^3	200	200	

特殊性质 A：对于所有 $0 \le i, j < n$ 且 $i \ne j$，均有线段 i 不包含线段 j，即 $l_i > l_j$ 或 $r_i < r_j$。

特殊性质 B：对于所有 $0 \le i < j < n$，均有线段 i 包含线段 j 或线段 i 与线段 j 不相交，即 $l_i \le l_j \le r_j \le r_i$ 或 $r_i < l_j$ 或 $l_i > r_j$。

特殊性质 C：n 条线段的全部 $2n$ 个端点互不相同，即 $l_0, l_1, \ldots, l_{n-1}, r_0, r_1, \ldots, r_{n-1}$ 两两不同。

传送（teleport）

【题目描述】

C 国共有 n 座城市，编号为 $0 \sim n-1$。这 n 座城市由 $n-1$ 条道路连接，形成树形结构。第 i $(0 \le i < n-1)$ 条道路连接城市 u_i 和 v_i，从其一端的城市到达另一端需要耗费 1 单位的时间。

为了提升通行效率，C 国研发了一种新型传送门。每座城市中均有一扇传送门。使用传送门同样耗费 1 单位时间，但由于系统尚不稳定，它会将使用者**等概率**地传送到所有 n 座城市之一。**注意：使用传送门也可能被传送到当前所在的城市。**

为了检测传送门的效果，C 国进行了 m 次测试。第 i $(0 \le i < m)$ 次测试要求测试员从城市 x_i 出发，去往城市 y_i。在从起点前往终点的过程中，测试员可以选择沿道路移动，或是使用传送门。由于可能的通行方式很多，测试员需要计算出期望耗时最短的通行方式。具体地，定义一种通行方式如下：对于每座非终点的城市，选择一座与其相邻的城市或是使用传送门，每当测试员到达该城市时，均按事先确定的方式移动，即移动至该相邻的城市，或使用传送门。形式化地，一种通行方式可以用一个长度为 n 的序列 $[a_0, \ldots, a_{n-1}]$ 表示，其中 $a_{y_i} = -1$，且对于所有 $j \ne y_i$，均有 a_j 与 j 相邻，或 $a_j = n$。每当测试员到达城市 j $(j \ne y_i)$ 时，若 $a_j < n$，则测试员将移动到 a_j，否则测试员将使用传送门。称一种通行方式是合理的，当且仅当其期望耗时为有限值。

对于每次测试，请计算在所有合理的通行方式中，期望耗时的最小值。

【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 `teleport.h`，即在程序开头加入以下代码：

```
#include "teleport.h"
```

选手需要在提交的程序源文件 `teleport.cpp` 中实现以下函数：

```
std::vector<std::pair<long long, int>> teleport(int c, int
    n, int m, std::vector<int> u, std::vector<int> v,
    std::vector<int> x, std::vector<int> y);
```

- c, n, m 分别表示测试点编号、城市数量和测试的次数。$c = 0$ 表示该测试点为样例。
- u, v 分别表示每条道路连接的两座城市。
- x, y 分别表示每次测试的起点与终点。

- 该函数需要返回一个长度**恰好**为 m 的**二元组**序列 $(a_0, b_0), (a_1, b_1), \ldots, (a_{m-1}, b_{m-1})$，其中 a_i, b_i $(0 \leq i < m)$ 表示第 i 次测试中，期望耗时的最小值的**最简分数形式**为 a_i/b_i。特别地，若期望耗时的最小值为正整数，则视为 $b_i = 1$。
- 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_teleport.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 teleport.cpp -o teleport -O2 -std=c++14
   -static
```

对于编译得到的可执行文件 `teleport`：

- 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 $i+2$ $(0 \leq i < n-1)$ 行包含两个非负整数 u_i, v_i。
  - 第 $i+n+1$ $(0 \leq i < m)$ 行包含两个非负整数 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 $i+1$ $(0 \leq i < m)$ 行包含两个正整数 a_i, b_i。

【样例 1 输入】

```
0 4 4
0 1
1 2
2 3
0 3
0 1
0 2
1 2
```

【样例 1 输出】

```
7 3
1 1
2 1
```

```
4 1 1
```

【样例 1 解释】

对于第 0 次测试，

- 若通行方式为 $[1,2,3,-1]$，则耗时为固定值 3；
- 若通行方式为 $[4,2,3,-1]$，则测试员将不断使用传送门直至离开城市 0，因此期望耗时为 7/3；
- 若通行方式为 $[4,4,3,-1]$，则测试员将不断使用传送门直至到达城市 2 或城市 3，因此期望耗时为 3；
- 若通行方式为 $[1,0,4,-1]$，则测试员将永远在城市 0 与城市 1 间移动，因此该通行方式不是合理的。

可以证明，期望耗时的最小值为 7/3。

【样例 2】

见选手目录下的 ***teleport/teleport2.in*** 与 ***teleport/teleport2.ans***。
该样例满足测试点 2, 3 的约束条件。

【样例 3】

见选手目录下的 ***teleport/teleport3.in*** 与 ***teleport/teleport3.ans***。
该样例满足测试点 4 ∼ 6 的约束条件。

【样例 4】

见选手目录下的 ***teleport/teleport4.in*** 与 ***teleport/teleport4.ans***。
该样例满足测试点 7 ∼ 8 的约束条件。

【样例 5】

见选手目录下的 ***teleport/teleport5.in*** 与 ***teleport/teleport5.ans***。
该样例满足测试点 9 的约束条件。

【样例 6】

见选手目录下的 ***teleport/teleport6.in*** 与 ***teleport/teleport6.ans***。
该样例满足测试点 16 的约束条件。

【样例 7】

见选手目录下的 ***teleport/teleport7.in*** 与 ***teleport/teleport7.ans***。
该样例满足测试点 17 ∼ 20 的约束条件。

【数据范围】

对于所有测试数据，均有：

- $2 \le n \le 5 \times 10^5$，$1 \le m \le 10^6$；
- 对于所有 $0 \le i < n-1$，均有 $0 \le u_i, v_i < n$，且所有 (u_i, v_i) 构成一棵树；
- 对于所有 $0 \le i < m$，均有 $0 \le x_i, y_i < n$ 且 $x_i \ne y_i$。

测试点编号	$n \le$	$m \le$	特殊性质
1	4	20	无
2, 3	5	30	
4 ∼ 6	10^2	1	
7, 8	10^3	2,000	A
9		10^6	无
10, 11	10^5		A
12 ∼ 15			无
16	5×10^5	5×10^5	B
17 ∼ 20		10^6	无

特殊性质 A：对于所有 $0 \le i < n-1$，均有 $u_i = i$ 且 $v_i = i+1$。
特殊性质 B：对于所有 $0 \le i < m$，均有 $y_i = 0$。

布丁（pudding）

本题为交互题。

【题目描述】

小 L 和小 S 都非常喜欢香甜而软糯的布丁。在品尝过各种布丁后，她们将所有布丁的美味度量化为不超过 4500 的正整数。

小 L 正在学习制作布丁。因熟练程度有限，她只能制作出美味度不超过常数 m 的布丁。在某次尝试中，小 L 成功制作出了一块美味度为 w 的布丁。小 S 想品尝小 L 制作的布丁，但需要按照小 L 要求的方式求出她制作的布丁的美味度。

具体地，小 S 可以从商店购买若干块布丁并交给小 L。小 L 会把自己制作的布丁混入其中，并将这些布丁按照美味度升序排序。然后，小 L 会计算全部相邻布丁美味度的最大公约数之和并告诉小 S。最后，她会将所有小 S 买来的布丁吃掉。形式化地，设小 S 购买了 k 块布丁，美味度分别为 $a_0, a_1, \ldots, a_{k-1}$。将 $[a_0, a_1, \ldots, a_{k-1}, w]$ 按升序排序后的结果记为 $[b_0, b_1, \ldots, b_{k-1}, b_k]$，则小 L 会将 $\sum_{i=1}^{k} \gcd(b_{i-1}, b_i)$ 的值告诉小 S。

由于前往商店的时间成本与购买布丁的经济成本都很高，小 S 希望尽可能减少购买的次数以及购买的布丁总数。你需要帮助小 S 制定购买策略，以求出小 L 制作的布丁的美味度。

【实现细节】

选手不需要，也不应实现 `main` 函数。

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

```
#include "pudding.h"
```

选手需要在提交的程序源文件 `pudding.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交互库调用恰好一次。

```
int find_tastiness(int c, int m);
```

- c, m 分别表示测试点编号与小 L 制作的布丁的美味度的上界；
- 该函数需要返回一个正整数 w，表示小 L 制作的布丁的美味度。
- 对于每个测试点，该函数会被交互库调用恰好 t 次。

选手可以通过调用以下函数进行一次询问：

```
int query_tastiness(std::vector<int> a);
```

- a 表示小 S 购买的布丁的美味度的序列。选手需要确保 a 非空，且其中的每个元素均为不超过 4500 的正整数。
- 该函数会返回小 L 告诉小 S 的值，具体含义如【题目描述】中所示。
- 选手需要确保交互库每次调用 `find_tastiness` 时，调用该函数的次数不超过 15，且调用该函数时传入的 a 的长度之和不超过 3000。

在任何情况下，交互库运行所需时间均不会超过 1.5 秒，所用内存不会超过 64 MiB。

本试题目录下的 `template_pudding.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 pudding.cpp -o pudding -O2 -std=c++14 -static
```

对于编译得到的可执行文件 `pudding`：

- 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, t, m。
  - 第二行包含 t 个正整数，分别表示每组测试数据中 w 的值。
- 可执行文件将输出以下格式的数据至标准输出：
  - 若 t 次调用 `find_tastiness` 的返回值均正确，则：
    - 输出的第一行为 `Correct!`；
    - 输出的第二行为 `Max queries used: Q`，其中 Q 表示所有测试数据中调用 `query_tastiness` 的次数的最大值；
    - 输出的第三行为 `Max total puddings queried: S`，其中 S 表示所有测试数据中调用 `query_tastiness` 时传入的 a 的长度之和的最大值。
  - 若至少一次调用 `find_tastiness` 的返回值不正确，则只会输出一行 `Wrong answer.`。
- 若调用 `query_tastiness` 时传入的参数不符合要求，或调用次数超过上限，则可执行文件会向**标准错误流**输出错误信息，并返回 -1。
- 选手可以在运行可执行文件时启用 `-v` 或 `--verbose` 参数，此时可执行文件将会额外输出以下内容：
  - 每次调用 `find_tastiness` 的返回值、正确性、调用 `query_tastiness` 的次数与传入的 a 的长度之和。
  - 每次调用 `query_tastiness` 时传入的参数、计算过程以及返回值。
  - 程序最终获得的分数比例，具体可见【评分方式】一节。

【样例 1 输入】

```
0 2 197
26 121
```

【样例 1 输出】

```
Correct!
Max queries used: 2
Max total puddings queried: 5
```

【样例 1 解释】

对于第一组测试数据，小 L 制作的布丁的美味度为 26。以下是一种可能的交互过程：

- 调用 `query_tastiness([2026, 7, 20])`，则 $b = [7, 20, 26, 2026]$，因此函数返回 $\gcd(7, 20) + \gcd(20, 26) + \gcd(26, 2026) = 1 + 2 + 2 = 5$；
- 调用 `query_tastiness([13, 52])`，则 $b = [13, 26, 52]$，因此函数返回 $\gcd(13, 26) + \gcd(26, 52) = 13 + 26 = 39$；
- 返回 26，答案正确；
- 调用 `query_tastiness` 的次数为 2，调用 `query_tastiness` 时传入的 a 的长度之和为 $3 + 2 = 5$。

【样例 2】

见选手目录下的 ***pudding/pudding2.in*** 与 ***pudding/pudding2.ans***。
该样例满足测试点 1 的约束条件。

【样例 3】

见选手目录下的 ***pudding/pudding3.in*** 与 ***pudding/pudding3.ans***。
该样例满足测试点 2 的约束条件。

【样例 4】

见选手目录下的 ***pudding/pudding4.in*** 与 ***pudding/pudding4.ans***。
该样例满足测试点 3 的约束条件。

【数据范围】

对于所有测试数据，均有：

- $1 \le t \le 3000$；
- $1 \le m \le 3000$，$1 \le w \le m$。

测试点编号	分值	$t=$	$m=$	特殊性质
1	10	35	35	无
2	20	430	3,000	A
3	70	3,000		无

特殊性质 A：w 为质数。

【评分方式】

注意：

- 选手不应当通过非法方式获取交互库的内部信息，如试图直接读取 w 的值，或直接与标准输入、输出流进行交互。此类行为将被视为作弊。
- 交互库不是适应性的，即每次调用 `find_tastiness` 时 w 的值就已经确定，不会随交互过程变化。
- 最终的评测交互库与样例交互库的实现不同。

若 `find_tastiness` 函数的返回值不正确，或调用 `query_tastiness` 时传入的参数不符合要求，则相应测试点得 0 分。

在上述条件基础上：

- 对于每个测试点，设 Q 表示所有测试数据中调用 `query_tastiness` 的次数的最大值，S 表示所有测试数据中调用 `query_tastiness` 时传入的 a 的长度之和的最大值，$score$ 表示该测试点的分值，则程序获得 $\lfloor f(Q) \cdot g(S) \cdot score \rfloor$ 分，其中 f 与 g 的计算方式如下：

Q	$f(Q)$
$Q \le 4$	1
$5 \le Q \le 15$	0.7^{Q-4}

S	$g(S)$
$S \le 35$	1
$36 \le S \le 75$	$1-(S-35)/100$
$76 \le S \le 235$	$0.2+\sqrt{(235-S)/1000}$
$236 \le S \le 3000$	$0.2 \times 2^{-(S-235)/1500}$