2026 全国青少年信息学奥林匹克冬令营

CCF NOI 冬令营 2026

测试

时间：2026 年 2 月 9 日 08:00 ～ 13:00

题目名称	二进制	猫和老鼠	画树
目录	binary	game	paint
每个测试点时限	3.0 秒	6.0 秒	1.0 秒
内存限制	1024 MiB	1024 MiB	1024 MiB
测试点数目	25	25	4
测试点是否等分	是	是	否
预测试点数目	25	25	4

提交源程序文件名

对于 C++ 语言	binary.cpp	game.cpp	paint.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源代码为准。
2. 选手提交的程序源文件名必须使用英文小写。
3. 选手提交的程序源文件大小不得超过 100 KiB。
4. 程序可使用的栈空间内存限制与题目的内存限制一致。
5. 禁止在源代码中改变编译器参数（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。
6. 因违反上述规定而出现的问题，申诉时一律不予受理。
7. 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。

二进制（binary）

【题目描述】

小 H 在学习二进制运算的过程中遇到了一个经典问题：给定一个初始为 0 的整数，每次操作可以将其乘 2 或加 1，求将其变为某个给定正整数 x 的最少操作次数。小 H 发现，这可以根据 x 的二进制表示求出答案。

基于这个问题，小 H 提出了以下问题：给定两个正整数 x, y，定义一次操作为以下四种之一：

1. 将 x 乘 2，即 $x \leftarrow 2x$；
2. 将 y 乘 2，即 $y \leftarrow 2y$；
3. 将 x 加 1，即 $x \leftarrow x + 1$；
4. 将 y 加 1，即 $y \leftarrow y + 1$。

小 H 想知道最少需要多少次操作，才能使得 x 和 y 相等。你需要帮助小 H 求出操作次数的最小值。

【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包含头文件 `binary.h`，即在程序开头加入以下代码：

```
#include "binary.h"
```

选手需要在提交的程序源文件 `binary.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**。

```
long long binary(long long x, long long y);
```

- x, y 表示给定的两个数。
- 该函数需要返回操作次数的最小值。
- 对于每个测试点，该函数会被交互库调用**恰好** t **次**。

注意：在任何情况下，交互库运行所需时间均不会超过 1.8 秒，所用内存为固定大小，且均不超过 64 MiB。

【测试程序方式】

试题目录下的 grader.cpp 是交互库参考实现，最终测试时所用的交互库实现与该参考实现有所不同，因此选手的解法不应该依赖交互库实现。

选手可以在本题目录下使用如下命令编译得到可执行程序：

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

对于编译得到的可执行程序：

- 可执行文件将从**标准输入**读入以下格式的数据：
  - 输入的第一行包含两个非负整数 c, t，分别表示测试点编号和测试数据组数。
  - 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 x, y，表示给定的两个数。
- 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，输出一行一个整数，表示操作次数的最小值。

【样例 1 输入】

```
0 5
1 2
1 5
3 6
7 33
5 9
```

【样例 1 输出】

```
1
3
1
4
2
```

【下发文件说明】

在本试题目录下：

1. `grader.cpp` 是提供的交互库参考实现。
2. `binary.h` 是头文件，选手不用关心具体内容。
3. `template_binary.cpp` 是提供的示例代码，选手可参考并实现自己的代码。

选手注意对所有下发文件做好备份。最终评测时只测试本试题目录下的 `binary.cpp`，对该程序以外文件的修改不会影响评测结果。

【数据范围】

对于所有测试数据，均有：

- $1 \le t \le 5 \times 10^7$；
- $1 \le x < y \le 10^{18}$。

测试点编号	$t =$	$y \le$	特殊性质
$1 \sim 4$	10	10	无
5	10^2	10^2	B
6			无
$7, 8$	10^3	10^3	B
$9 \sim 11$			无
12	10	10^6	A
13			无
14	10^6		A
15			B
16			无
$17, 18$		10^{18}	B
$19 \sim 21$			无
$22 \sim 24$	2.5×10^7		
25	5×10^7		

特殊性质 A：$y - x \le 10^3$。

特殊性质 B：存在 $k \ge 1$ 与 $0 \le z < 2^k$ 满足 $y = x \times 2^k + z$。

【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，如直接与标准输入、输出流进行交互。此类行为将被视为作弊；
- **最终的评测交互库与样例交互库的实现不同。**

本题首先会受到和传统题相同的限制，例如编译错误会导致整道题目得 0 分，运行时错误、超过时间限制、超过空间限制等会导致相应测试点得 0 分等。选手只能在程序中访问自己定义的变量以及交互库给出的变量，尝试访问其他地址空间将可能导致编译错误或运行错误。

在上述条件基础上：

- 对于每个测试点，程序得到满分当且仅当每次调用 `binary` 函数时返回的答案均正确。

猫和老鼠（game）

【题目描述】

《猫和老鼠》是一部家喻户晓的动画片。小 G 根据这部动画片设计了一个游戏。在这个游戏中，玩家需要帮助 Tom 使用机器猫抓住 Jerry。

Jerry 的活动范围是数轴上的区间 $[0, m]$。初始时刻（即第 0 秒时），Jerry 可能位于该区间内的任意位置。之后，它可以在该区间内自由移动，但任意时刻的速度不会超过每秒 1 单位长度。

Tom 有 n 个可供部署的机器猫，其中部署第 i $(1 \le i \le n)$ 个机器猫的成本为 w_i。若部署了第 i $(1 \le i \le n)$ 个机器猫，则其会在第 t_i 秒从位置 a_i 出现，然后以每秒 1 单位长度的速度向位置 b_i 匀速移动，并在抵达后消失。

Jerry 初始拥有 k 点生命值。每当它与一个机器猫完全重合时（即存在某个时刻，两者位置完全相同），其生命值将减少 1，该机器猫也会随之失效。当 Jerry 的生命值**小于或等于** 0 时，Tom 即成功抓住 Jerry。

小 G 设定 Tom 必须在初始时刻就部署好机器猫。因此，玩家需要在游戏开始前选定若干个机器猫进行部署。玩家获胜当且仅当部署好机器猫后，对于 Jerry 所有可能的移动路径，Tom 均能成功抓住 Jerry。

小 G 为这个游戏设计了很多关卡，并邀请你进行测试。为了控制游戏难度，小 G 计划为部署机器猫的成本总和设置一个合理的上限。你需要帮助小 G 求出，玩家获胜所需部署的机器猫的成本总和的最小值。

【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包含头文件 `game.h`，即在程序开头加入以下代码：

```
#include "game.h"
```

选手需要在提交的程序源文件 `game.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**。

```
long long game(int n, int m, int k, std::vector<int> a,
  std::vector<int> b, std::vector<int> t, std::vector<int> w);
```

- n, m, k 分别表示机器猫的个数、Jerry 的活动范围与 Jerry 的初始生命值。
- 对于 $0 \le i < n$，a_i, b_i, t_i, w_i 分别表示第 $i + 1$ 个机器猫的出现位置、最终位置、出现时间与部署成本。

- 该函数需要返回成本总和的最小值。特别地，如果部署所有机器猫也无法获胜，则返回 -1。
- 对于每个测试点，该函数会被交互库调用**恰好** t **次**。

注意：在任何情况下，交互库运行所需时间均不会超过 0.1 秒，所用内存为固定大小，且均不超过 64 MiB。

【测试程序方式】

试题目录下的 `grader.cpp` 是交互库参考实现，最终测试时所用的交互库实现与该参考实现有所不同，因此选手的解法不应该依赖交互库实现。

选手可以在本题目录下使用如下命令编译得到可执行程序：

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

对于编译得到的可执行程序：

- 可执行文件将从**标准输入**读入以下格式的数据：
  - 输入的第一行包含两个非负整数 c, t，分别表示测试点编号和测试数据组数。
  - 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，分别表示机器猫的个数、Jerry 的活动范围与 Jerry 的初始生命值。
    - 第 $i+1$ $(1 \le i \le n)$ 行包含四个非负整数 a_i, b_i, t_i, w_i，分别表示第 i 个机器猫的出现位置、最终位置、出现时间与部署成本。
- 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，输出一行一个整数，表示成本总和的最小值。特别地，如果部署所有机器猫也无法获胜，则输出 -1。

【样例 1 输入】

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

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

【样例 1 输出】

```
6
-1
35
```

【样例 1 解释】

该样例共包含三组测试数据。

对于第一组测试数据，玩家可以选择部署第 $1,2,3$ 个机器猫，成本总和为 $1+2+3=6$。若玩家选择部署第 $1,3$ 个机器猫，则当 Jerry 初始时刻位于位置 7，且从第 6 秒开始以每秒 1 单位长度的速度匀速移动至位置 1 时，Tom 无法成功抓住 Jerry。

对于第二组测试数据，当 Jerry 初始时刻位于位置 5.5 且不移动时，它只会与第 1 个机器猫在第 3.5 秒时完全重合，因此部署所有机器猫也无法获胜。

对于第三组测试数据，玩家可以选择部署第 $1,2,3,4,5,7$ 个机器猫，成本总和为 $7+8+9+3+3+5=35$。

【样例 2】

见选手目录下的 ***game/game2.in*** 与 ***game/game2.ans***。

该样例满足测试点 $3,4$ 的约束条件。

【样例 3】

见选手目录下的 ***game/game3.in*** 与 ***game/game3.ans***。

该样例满足测试点 $5,6$ 的约束条件。

【样例 4】

见选手目录下的 ***game/game4.in*** 与 ***game/game4.ans***。
该样例满足测试点 $10 \sim 12$ 的约束条件。

【样例 5】

见选手目录下的 ***game/game5.in*** 与 ***game/game5.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 ***game/game6.in*** 与 ***game/game6.ans***。
该样例满足测试点 $23, 24$ 的约束条件。

【样例 7】

见选手目录下的 ***game/game7.in*** 与 ***game/game7.ans***。
该样例满足测试点 25 的约束条件。

【下发文件说明】

在本试题目录下：

1. `grader.cpp` 是提供的交互库参考实现。
2. `game.h` 是头文件，选手不用关心具体内容。
3. `template_game.cpp` 是提供的示例代码，选手可参考并实现自己的代码。

选手注意对所有下发文件做好备份。最终评测时只测试本试题目录下的 `game.cpp`，对该程序以外文件的修改不会影响评测结果。

【数据范围】

设 N, S 分别为单个测试点内所有测试数据的 n, nk 的和。对于所有测试数据，均有：

- $1 \le t \le 20$；
- $1 \le n \le 5 \times 10^4$，$N \le 3 \times 10^5$；
- $1 \le m \le 10^9$，$1 \le k \le 10$，$S \le 10^6$；
- 对于所有 $1 \le i \le n$，均有 $0 \le a_i, b_i \le m$，$0 \le t_i, w_i \le 10^9$。

测试点编号	$n \leq$	$k \leq$	特殊性质
$1, 2$	10	10	A
$3, 4$	10^3	1	BC
$5, 6$	5×10^4		CD
$7, 8$	10^3	10	C
9	5×10^4		CD
$10 \sim 12$			C
$13, 14$	10^3	1	无
15	5×10^4		D
$16 \sim 18$			无
$19 \sim 22$	80	10	
$23, 24$	300		
25	5×10^4		

特殊性质 A：$m \leq 10$，且对于所有 $1 \leq i \leq n$，均有 $t_i, w_i \leq 10$。

特殊性质 B：$m \leq 10^3$，且对于所有 $1 \leq i \leq n$，均有 $t_i \leq 10^6$。

特殊性质 C：对于所有 $1 \leq i \leq n$，均有 $w_i = 0$。

特殊性质 D：对于所有 $1 \leq i \leq n$，均有 $a_i \leq b_i$。

【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，如直接与标准输入、输出流进行交互。此类行为将被视为作弊；
- **最终的评测交互库与样例交互库的实现不同。**

本题首先会受到和传统题相同的限制，例如编译错误会导致整道题目得 0 分，运行时错误、超过时间限制、超过空间限制等会导致相应测试点得 0 分等。选手只能在程序中访问自己定义的变量以及交互库给出的变量，尝试访问其他地址空间将可能导致编译错误或运行错误。

在上述条件基础上：

- 对于每个测试点，程序得到满分当且仅当每次调用 `game` 函数时返回的答案均正确。

画树（paint）

【题目描述】

小 F 在美术课上学习了如何画树。他画了一棵 n 个结点的树 T，结点编号为 $1 \sim n$。

由于对当前这棵树不太满意，小 F 打算使用 k 对一一配对的铅笔与橡皮来涂改它。初始时，他把每对铅笔与橡皮放置在树的某个结点上。具体地，对于 $1 \le i \le k$，记铅笔所在的结点为 p_i，橡皮所在的结点为 e_i，则初始时 $p_i = e_i$。在任意时刻，同一个结点上可以同时放置多支铅笔与多只橡皮。

每次涂改时，小 F 将首先选择一对铅笔与橡皮。设小 F 选择的是第 t $(1 \le t \le k)$ 对铅笔与橡皮，则他将按以下步骤进行涂改：

1. 选择**任意**一个结点 x $(1 \le x \le n)$，将铅笔移动至结点 x，并绘制移动形成的边，即在 p_t 与 x 之间添加一条边，然后令 $p_t \leftarrow x$；
2. 选择一个与 e_t **有边直接相连**的结点 y $(1 \le y \le n)$（可以是上一步中新绘制的边），将橡皮移动至结点 y，并擦除移动经过的边，即删除 e_t 与 y 之间的边，然后令 $e_t \leftarrow y$。

当然，小 F 需要保证**每次涂改后得到的图仍然是一棵树**。

小 F 希望使用尽可能少的铅笔与橡皮对将树 T 涂改为另一棵树 T'。你需要帮助小 F 构造一组涂改的方案。具体地，你需要确定铅笔与橡皮对的数量 k，并且指定每对铅笔与橡皮的初始位置 p_i, e_i $(1 \le i \le k,\ p_i = e_i)$，再构造一组涂改序列 (t, x, y) $(1 \le t \le k,\ 1 \le x, y \le n)$，使得依次执行这些涂改后，树 T 能被转变为树 T'。

【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包含头文件 `paint.h`，即在程序开头加入以下代码：

```
#include "paint.h"
```

选手需要在提交的程序源文件 `paint.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**。

```
void paint(int n, std::vector<int> u, std::vector<int> v);
```

- n 表示树 T 的结点个数。
- 对于 $0 \le i < n-1$，u_i, v_i 表示树 T 的一条边。
- 对于 $n-1 \le i < 2n-2$，u_i, v_i 表示树 T' 的一条边。
- 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设置铅笔与橡皮对的数量与每对铅笔与橡皮的初始位置：

```
void setting(int k, std::vector<int> p);
```

- k 表示使用的铅笔与橡皮对的数量。选手需要保证 $0 \le k \le 2n$。
- 对于 $0 \le i < k$，p_i 表示第 $i+1$ 对铅笔与橡皮的初始位置。选手需要保证 p 的长度为 k，且对于所有 $0 \le i < k$，均有 $1 \le p_i \le n$。
- 选手需要保证交互库每次调用 `paint` 时，**恰好调用了一次该函数**。

选手可以通过调用以下函数进行一次涂改：

```
void alter(int t, int x, int y);
```

- t, x, y 分别表示选择的铅笔与橡皮对的编号与结点的编号，具体含义如【题目描述】中所示。选手需要保证 $1 \le t \le k$，$1 \le x, y \le n$，且**涂改后得到的图仍然是一棵树**。
- 选手需要保证交互库每次调用 `paint` 时，调用该函数的次数不超过 10^5，且所有该函数的调用**均在调用 `setting` 函数之后**。

注意：在任何情况下，交互库运行所需时间均不会超过 0.2 秒，所用内存为固定大小，且均不超过 64 MiB。

【测试程序方式】

试题目录下的 `grader.cpp` 是交互库参考实现，最终测试时所用的交互库实现与该参考实现有所不同，因此选手的解法不应该依赖交互库实现。

选手可以在本题目录下使用如下命令编译得到可执行程序：

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

对于编译得到的可执行程序：

- 可执行文件将从**标准输入**读入以下格式的数据：
  - 输入的第一行包含两个非负整数 c, t，分别表示测试点编号和测试数据组数。
  - 接下来依次为每组测试数据，对于每组测试数据：
    - 第一行包含一个正整数 n，表示树 T 的结点个数。
    - 第 $i+1$ $(1 \le i \le n-1)$ 行包含两个正整数 u_i, v_i，表示树 T 的一条边。
    - 第 $i+n$ $(1 \le i \le n-1)$ 行包含两个正整数 u'_i, v'_i，表示树 T' 的一条边。
- 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：
    - 第一行包含两个非负整数 k, m，分别表示使用的铅笔与橡皮对的数量与涂改序列的长度。

* 第二行包含 k 个正整数 $p_1, p_2, \ldots, p_k$，分别表示每对铅笔与橡皮的初始位置。
* 第 $i+2$ $(1 \le i \le m)$ 行包含三个正整数 t, x, y，分别表示第 i 次涂改选择的铅笔与橡皮对的编号与结点的编号。

【样例 1 输入】

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

【样例 1 输出】

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

```
2 5 1
```

【样例 1 解释】

该样例共包含两组测试数据。

对于第一组测试数据，树 T 包含边 $\{(1,3),(3,2),(1,5),(5,4)\}$，树 T' 包含边 $\{(1,4),(2,3),(3,1),(5,3)\}$。一种可行的涂改方案如下：

- 初始时，共有 1 对铅笔与橡皮放置在结点 1。
- 第一次涂改选择 $x=4$，$y=5$，添加边 $(1,4)$，删除边 $(1,5)$，得到的树包含边 $\{(1,3),(3,2),(5,4),(1,4)\}$，此时 $p_1=4$，$e_1=5$；
- 第二次涂改选择 $x=5$，$y=4$，添加边 $(4,5)$，删除边 $(4,5)$，此时 $p_1=5$，$e_1=4$；
- 第三次涂改选择 $x=3$，$y=5$，添加边 $(5,3)$，删除边 $(4,5)$，得到的树包含边 $\{(1,3),(3,2),(1,4),(5,3)\}$，即为树 T'。

【下发文件说明】

在本试题目录下：

1. `grader.cpp` 是提供的交互库参考实现。
2. `paint.h` 是头文件，选手不用关心具体内容。
3. `template_paint.cpp` 是提供的示例代码，选手可参考并实现自己的代码。

选手注意对所有下发文件做好备份。最终评测时只测试本试题目录下的 `paint.cpp`，对该程序以外文件的修改不会影响评测结果。

【数据范围】

对于所有测试数据，均有：

- $1 \le t \le 10$；
- $4 \le n \le 200$；
- 对于所有 $1 \le i < n$，均有 $1 \le u_i, v_i \le n$，且所有的边构成了一棵树；
- 对于所有 $1 \le i < n$，均有 $1 \le u'_i, v'_i \le n$，且所有的边构成了一棵树；
- T 和 T' 均在所有 n 个结点的树中**独立均匀随机**生成。

测试点编号	分值	$n=$
1	14	4
2	23	20
3	26	50
4	37	200

【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，如直接与标准输入、输出流进行交互。此类行为将被视为作弊；
- **最终的评测交互库与样例交互库的实现不同。**

本题首先会受到和传统题相同的限制，例如编译错误会导致整道题目得 0 分，运行时错误、超过时间限制、超过空间限制等会导致相应测试点得 0 分等。选手只能在程序中访问自己定义的变量以及交互库给出的变量，尝试访问其他地址空间将可能导致编译错误或运行错误。

每次调用 `paint` 函数时，若 `setting` 函数或 `alter` 函数调用不合法，或 `alter` 函数调用次数超过 10^5，则相应测试点得 0 分。

在上述条件基础上：

- 在测试点 1 和测试点 2 中，程序得到满分当且仅当每次调用 `paint` 函数返回时树 T 均被转变为树 T'。
- 在测试点 3 中，对于每组测试数据，设 $k_{\min}$ 表示使用的铅笔与橡皮对数的最小值，
  - 若 `paint` 函数返回时树 T 未被转变为树 T'，则获得 0 分；
  - 否则，获得 $\lfloor 26 \cdot 0.97^{k-k_{\min}} \rfloor$ 分。

  程序得到的分数为所有测试数据的得分的**最小值**。
- 在测试点 4 中，对于每组测试数据，设 $k_{\min}$ 表示使用的铅笔与橡皮对数的最小值，
  - 若 $k \neq k_{\min}$，则获得 0 分；
  - 否则，若 `paint` 函数返回时树 T 未被转变为树 T'，则获得 4 分；
  - 否则，获得 $\lfloor 4 + 33 \cdot 0.99^{\max(\lceil m/2000 \rceil - 5, 0)} \rfloor$ 分。

  程序得到的分数为所有测试数据的得分的**最小值**。