20251025 CSP-J 公开模拟赛

P14304 【MX-J27-T1】分块

题目描述

小 L 喜欢分块，于是小 L 给了你一个正整数 n，你需要统计有多少个不超过 n 的正整数 x 满足 $\lfloor\sqrt{x}\rfloor$ 是 x 的因数。

因为小 L 怕你浑水摸鱼，所以小 L 给了你 q 组不同的询问 $n_1,\ldots,n_q$，每组询问的 n_i 可能不同。你需要对每个 $n=n_i$ 求出正确答案。

题面中的 $\lfloor\rfloor$ 为向下取整符号，$\lfloor a\rfloor$ 表示最大的不超过 a 的整数。例如，$\lfloor 1.9\rfloor=1$，$\lfloor 7\rfloor=7$，而 $\lfloor \pi\rfloor=3$。

输入格式

第一行，一个整数 q。

接下来 q 行，第 i 行一个正整数 n_i，表示第 i 组询问对应的 n 的值。

输出格式

输出共 q 行。

第 i 行输出一个整数，表示 $n=n_i$ 时小 L 的问题的答案。

输入输出样例 #1

输入 #1

```
5
1
3
6
10
15
```

输出 #1

```
1
3
5
7
9
```

说明/提示

【样例解释 #1】

对 $n=6$，共有 5 个不超过 6 的正整数 x 符合题意：

- 若 $x=1$，$\lfloor\sqrt{x}\rfloor=1$，由于 1 是 1 的因数，所以 $x=1$ 符合条件；
- 若 $x=2$，$\lfloor\sqrt{x}\rfloor=1$，由于 1 是 2 的因数，所以 $x=2$ 符合条件；
- 若 $x=3$，$\lfloor\sqrt{x}\rfloor=1$，由于 1 是 3 的因数，所以 $x=3$ 符合条件；
- 若 $x=4$，$\lfloor\sqrt{x}\rfloor=2$，由于 2 是 4 的因数，所以 $x=4$ 符合条件；
- 若 $x=5$，$\lfloor\sqrt{x}\rfloor=2$，由于 2 不是 5 的因数，所以 $x=5$ 不符合条件；
- 若 $x=6$，$\lfloor\sqrt{x}\rfloor=2$，由于 2 是 6 的因数，所以 $x=6$ 符合条件。

类似地，可以得到 n 取 $1,3,10,15$ 时的答案分别为 $1,3,7$ 和 9。

【数据范围】

本题共 10 个测试点，每个 10 分。

对于所有数据，保证：

- $1\le q\le 10^5$；
- $1\le n_i\le 10^{18}$。

测试点编号	$n_i\le$	$q\le$	特殊性质
$1\sim 2$	10^6	10	有
3	^	^	无
4	^	10^5	^
5	10^{11}	10	有
6	^	10^5	^
$7\sim 8$	^	^	无
$9\sim 10$	10^{18}	^	^

- 特殊性质：保证 n_i 是完全平方数。

P14304【MX-J27-T1】分块

▍题意

给定一个整数 n，计算满足特定条件的整数数量。具体来说，对于每个整数 d，当 $\lfloor\sqrt{x}\rfloor=d$ 时，需要统计满足条件的 x 的个数。

数据范围：T 组测试数据，n 的范围未明确给出，但算法需要高效处理。

分析

对于每个满足 $\lfloor\sqrt{x}\rfloor = d$ 的整数 d，x 的取值范围为 $[d^2, (d+1)^2-1]$。在该区间内满足条件的 x 个数为：$\left\lfloor\frac{(d+1)^2-1}{d}\right\rfloor - \left\lfloor\frac{d^2-1}{d}\right\rfloor = d+2-\lfloor d-\frac{1}{d}\rfloor = d+2-(d-1) = 3$

设 d_0 为最大的满足 $(d+1)^2-1<n$ 的整数，则：

- 对于 $d=1$ 到 d_0，每个 d 贡献 3 个满足条件的 x
- 对于 $d=d_0+1$，需要特殊处理，计算其贡献

计算 d_0 可以使用二分查找法，找到最大的 d 满足 $d^2+2d\leq n$。

时间复杂度：$O(T\log n)$，其中 T 为测试数据组数。

参考代码

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;

int T, n, ans;

signed main() {
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);

    cin >> T;
    while(T--) {
        cin >> n;
        int l = 1, r = sqrt(n), mid;

        // 二分查找最大的 d 满足 d^2 + 2d <= n
        while(l < r) {
            mid = (l + r) / 2;
            if(mid * mid + 2 * mid <= n) {
                l = mid + 1;
            } else {
                r = mid;
            }
        }
        l--;

        ans = 3 * l;  // 前 d0 个 d 各贡献 3 个 x

        // 特殊处理 d = d0 + 1 的情况
        for(int i = 0; i < 3; i++) {
            if((l + 1) * (l + 1) + i * (l + 1) <= n) {
                ans++;
            }
        }

        cout << ans << "\n";
    }
```

```
    return 0;
}
```

P14305 【MX-J27-T2】转换

题目描述

给出一个仅包含 `char`、`bool` 、`int`、`long long`、`float`、`double` 六种类型的变量和 `+`、`*`、`,` 三种运算符的表达式。你希望求出该表达式运算结果的类型。

在 C++ 里，编译器在表达式求值时，如果发现参与运算的对象类型不一致，会尝试进行隐式类型转换。在本题里，我们只考虑**在 `+` 或 `*` 运算中，部分编译器会自动完成的类型转换操作**：

- 对两个相同类型的变量 $\texttt{a}, \texttt{b}$，表达式 $\texttt{a+b}$ 和 $\texttt{a*b}$ 的返回值的类型同时与 $\texttt{a}, \texttt{b}$ 的类型相同。
- 所有占用字节数小于 `int` 字节数的类型（如 `char`）会**自动转换为 `int`**。
- 对两个整型的运算，编译器会将其转化为 **精度更高（占用字节数更多）的数据类型** 进行运算。
  - 如对于表达式 $\texttt{c+d}$，若 $\texttt{c}, \texttt{d}$ 分别为 `int` 和 `long long` 类型，编译器会先将 $\texttt{c}$ 转换为 `long long` 类型，然后做 `long long` 类型的加法，运算结果也为 `long long` 类型。
  - 对两个浮点类型的运算，其规则类似。
- 当整数类型和浮点类型同时参与运算时，所有整数类型都会转换为其中 **精度最高的浮点类型**。
  - 如对于表达式 $\texttt{e*f}$，若 $\texttt{e}, \texttt{f}$ 分别为 `long long` 和 `float` 类型，编译器会先将 $\texttt{e}$ 转换为 `float` 类型，然后做 `float` 类型的乘法，运算结果也为 `float` 类型。

表达式 $\texttt{f,g}$ 的返回值为 $\texttt{g}$。因此，`,` 运算的运算结果类型与其第二个运算对象相同。注意，运算 `,` 的优先级低于运算 `+` 与 `*`。

为了方便，我们只给出表达式中每个变量的类型，而不涉及其变量名称。也就是说，表达式总形如

$l_1o_1l_2o_2\ldots l_{n-1}o_{n-1}l_n$

的形式，其中 l_i, o_i 都是字符串，满足 $l_i \in \{\texttt{char}, \texttt{bool}, \texttt{int}, \texttt{longlong}, \texttt{float}, \texttt{double}\}$，表示第 i 个变量对应的类型名称（特别地，`long long` 用不带空格的 $\texttt{longlong}$ 表示）；且 $o_i \in \{\texttt{+}, \texttt{*}, \texttt{,}\}$，表示表达式中第 i 个运算符的类型。

输入格式

本题有多组测试数据。

第一行，两个整数 c, T，分别表示测试点编号与测试数据组数。接下来输入每组测试数据。样例满足 $c = 0$。

对于每组测试数据：

- 仅一行，包含一个字符串 s，表示给定的表达式。

保证 s 可以写为 $l_1o_1l_2o_2\ldots l_{n-1}o_{n-1}l_n$ 的形式，该形式在**【题目描述】**中有对应的更严格的约束。

输出格式

对于每组测试数据：

- 输出一行一个字符串，表示给定的表达式的运算结果的类型（类似地，若运算结果的类型为 `long long`，输出不带空格的 `longlong`）。

输入输出样例 #1

输入 #1

```
0 5
char
int+bool
float*int+longlong
int+char*bool+double
float+bool*double,int*longlong+char
```

输出 #1

```
char
int
float
double
longlong
```

说明/提示

【样例解释 #1】

对于第一组数据，没有任何运算符，因此返回值类型即为唯一的变量的类型 `char`。

对于第二组数据，由于 `bool` 会自动转换为 `int`，`int` 与 `int` 加法，返回值类型仍然为 `int`。

对于第三组数据，先计算 `float*int` 得到 `float` 类型，再计算 `float+longlong` 得到 `float` 类型。

对于第四组数据，先计算 `char*bool` 得到 `int` 类型，再计算 `int+int+double` 得到 `double` 类型。

对于第五组数据，先计算 `bool*double` 和 `int*longlong` 得到 `float+double,longlong+char`，再计算 `float+double` 和 `longlong+char` 得到 `double,longlong`，最终返回值的类型为 `longlong`。

【数据范围】

本题共 10 个测试点，每个 10 分。

令 n 为表达式中的运算对象数量。对于所有数据，保证：

- $1 \le T \le 10$；
- $1 \le n \le 10^5$；
- 输入字符串总可以写为 $l_1 o_1 l_2 o_2 \dots l_{n-1} o_{n-1} l_n$ 形式，其中：

- $l_i \in \{\texttt{char}, \texttt{bool}, \texttt{int}, \texttt{longlong}, \texttt{float}, \texttt{double}\}$;
- $o_i \in \{\texttt{+}, \texttt{*}, \texttt{,}\}$。

测试点编号	$n \le$	特殊性质
1	2	无
$2 \sim 3$	100	无
4	10^5	ABC
5	^	AB
6	^	AC
7	^	A
8	^	B
9	^	C
10	^	无

- 特殊性质 A：不存在类型 `float` 和 `double`。
- 特殊性质 B：不存在类型 `char` 和 `bool`。
- 特殊性质 C：不存在运算符 `,`。

P14305【MX-J27-T2】转换

题意

给定 T 个表达式字符串，每个字符串包含变量类型（如 char、bool、int 等）和运算符（如逗号 $,$、乘号 *、加号 $+$ 等）。需要根据优先级规则确定每个表达式的最终类型。规则包括：

- 变量类型有优先级：$\text{char}, \text{bool} < \text{int} < \text{longlong} < \text{float} < \text{double}$。
- 运算符 $,$ 会忽略之前的内容（重置当前类型）。
- 运算符 * 和 $+$ 会将当前类型为 char 或 bool 的变量转换为 int。

分析

核心思路是通过遍历表达式字符串，根据规则动态更新当前类型优先级。使用一个变量 ans 记录当前最高优先级：

- 初始化 $\text{ans} = 0$。
- 遍历字符串的每个字符：
  - 遇到 $.$（逗号）时，重置 $\text{ans} = 0$。
  - 遇到类型关键字（如 c 对应 char）时，更新 ans 为对应优先级的最大值。
  - 遇到运算符 * 或 $+$ 时，若当前优先级低于 int，则强制提升为 int。

- 最终根据 ans 的值输出类型名称。

时间复杂度：$O(T \times |s|)$，其中 $|s|$ 为字符串长度，每个字符处理一次。

▍参考代码

```cpp
#include<bits/stdc++.h>
#define ll long long
using namespace std;

ll c, T, ans;
string s;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    cin >> c >> T;
    while (T--) {
        cin >> s;
        ans = 0;
        for (int i = 0; i < s.size(); i++) {
            if (s[i] == '.') ans = 0; // 逗号，忽略前面内容
            if (s[i] == 'c') ans = max(1ll, ans); // char 优先级为1
            if (s[i] == 'b' && s[i + 1] == 'o') ans = max(2ll, ans); // bool 优先级为2
            if (s[i] == 'i' && s[i + 1] == 'n') ans = max(3ll, ans); // int 优先级为3
            if (s[i] == '*' || s[i] == '+') ans = max(3ll, ans); // * 或 + 强制提升为int
            if (s[i] == 'l' && s[i + 1] == 'o') ans = max(4ll, ans); // longlong 优先级为4
            if (s[i] == 'f') ans = max(5ll, ans); // float 优先级为5
            if (s[i] == 'd') ans = max(6ll, ans); // double 优先级为6
        }
        if (ans == 1) puts("char");
        else if (ans == 2) puts("bool");
        else if (ans == 3) puts("int");
        else if (ans == 4) puts("longlong");
        else if (ans == 5) puts("float");
        else puts("double");
    }
    return 0;
}
```

P14306 【MX-J27-T3】旋律

题目描述

风铃草有一段旋律，旋律可以用 n 个正整数 $a_1, a_2, \ldots, a_n$ 描述。

风铃草喜欢更加悠长的旋律；但音符之间过大的差异又会破坏一段旋律整体的和谐。为此，她定义一段旋律序列的**和谐度**为序列的长度乘以 k（k 为正整数常数）再减去序列内元素的极差。

给定 k，你需要选出序列 $a_1, \ldots, a_n$ 的一个**非空**子序列，最大化它的**和谐度**。你只需要求出最大的和谐度即可。

【提示】

一个序列 $a_1, \ldots, a_n$ 的极差定义为 a 中最大值减最小值得到的结果。换句话说，它等于 $\max(a_1, \ldots, a_n) - \min(a_1, \ldots, a_n)$。

序列 a 是序列 b 的非空子序列，当且仅当 a 非空，且在 b 中删去任意若干个（可能为 0 个）元素后，b 可以变为 a。

输入格式

本题有多组测试数据。

第一行，两个整数 c, T，分别表示测试点编号与测试数据组数。接下来输入每组测试数据。样例满足 $c = 0$。

对于每组测试数据：

- 第一行，两个正整数 n 和 k，分别表示旋律序列的长度和给定的常数。
- 第二行，n 个正整数 $a_1, a_2, \ldots, a_n$。

输出格式

对于每组测试数据：

- 输出一行一个整数，表示所有非空子序列的和谐度的最大值。

输入输出样例 #1

输入 #1

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

输出 #1

```
4
3
```

说明/提示

【样例解释 #1】

对于第一组数据，$a = [3, 1, 8]$，$k = 3$。考虑枚举所有的非空子序列：

1. 选择子序列 $[a_1]$，和谐度为 $3 \times 1 - (3-3) = 3$；
2. 选择子序列 $[a_1, a_2]$，和谐度为 $3 \times 2 - (3-1) = 4$；
3. 选择子序列 $[a_1, a_2, a_3]$，和谐度为 $3 \times 3 - (8-1) = 2$；
4. 选择子序列 $[a_1, a_3]$，和谐度为 $3 \times 2 - (8-3) = 1$；
5. 选择子序列 $[a_2]$，和谐度为 $3 \times 1 - (1-1) = 3$；
6. 选择子序列 $[a_2, a_3]$，和谐度为 $3 \times 2 - (8-1) = -1$；
7. 选择子序列 $[a_3]$，和谐度为 $3 \times 1 - (8-8) = 3$。

因此，和谐度最大的非空子序列为 $[a_1, a_2]$ 即 $[3, 1]$，其和谐度为 4。

对于第二组数据，和谐度最大的非空子序列为 $[1, 1, 1]$，其和谐度为 3。

【数据范围】

本题共 20 个测试点，每个 5 分。

对于所有数据，保证：

- $1 \le T \le 10$；
- $1 \le n \le 10^5$，$1 \le k \le 10^8$；
- $1 \le a_i \le 10^8$。

::cute-table{tuack}

测试点编号	$n \le$	特殊性质
$1 \sim 2$	5	无
$3 \sim 4$	18	^
5	1000	A
6	^	B
$7 \sim 8$	^	C
$9 \sim 11$	^	无
$12 \sim 13$	10^5	A
$14 \sim 15$	^	B
$16 \sim 17$	^	C
$18 \sim 20$	^	无

- 特殊性质 A：保证数列 a 是一个公差非负的等差数列。换句话说，存在一个整数 $d \ge 0$，满足对所有 $1 \le i < n$，都有 $a_{i+1} - a_i = d$。
- 特殊性质 B：保证 $k = 10^8$。
- 特殊性质 C：保证 $k = 1$ 且 $1 \le a_i \le n$。

P14306【MX-J27-T3】旋律

题意

给定 T 组测试数据，每组数据包含一个序列 $a_1, a_2, \ldots, a_n$ 和一个正整数 k。需要选出一个非空子序列，使得和谐度 $L \times k - (max - min)$ 最大，其中 L 是子序列长度，max 和 min 分别是子序列中的最大值和最小值。

数据范围：T 组数据，n 的具体范围未明确，但算法需高效处理 n 较大的情况。

分析

由于和谐度只与子序列的最小值和最大值有关，与元素顺序无关，因此先将序列升序排序。排序后，最优解一定对应一个连续子数组（因为添加中间元素可能增加长度而不增加极差，或净收益为正）。

设排序后的序列为 $b_1, b_2, \ldots, b_n$（$b_i \leq b_{i+1}$）。对于连续子数组 $[i, j]$，和谐度为：
$(j - i + 1) \times k - (b_j - b_i)$
直接枚举所有子数组的复杂度为 $O(n^2)$，需优化。

考虑差分：极差 $b_j - b_i = \sum_{k=i+1}^{j} (b_k - b_{k-1})$。定义贡献值 $w_k = k - (b_k - b_{k-1})$（$k \geq 2$），则和谐度可表示为：
$k + \sum_{k=i+1}^{j} w_k$
其中 k 对应长度为 1 的子序列基础值。问题转化为求 w 数组的最大子段和。使用动态规划：

- 状态 f_i 表示以 w_i 结尾的最大子段和
- 转移：$f_i = \max(f_{i-1} + w_i, 0)$
- 答案：$\max(k, k + \max_i f_i)$

时间复杂度：排序 $O(n \log n)$，动态规划 $O(n)$，总复杂度 $O(n \log n)$。

参考代码

```cpp
#include<bits/stdc++.h>
#define int long long
using namespace std;
const int N = 1e6 + 5;
int T, n, k, a[N], f[N], ans;

signed main() {
    scanf("%lld", &T);
    while (T--) {
        scanf("%lld%lld", &n, &k);
        for (int i = 1; i <= n; i++) scanf("%lld", &a[i]);
        sort(a + 1, a + n + 1);
        ans = k;
        f[1] = 0;
        for (int i = 2; i <= n; i++) {
            f[i] = max(f[i - 1] + k - (a[i] - a[i - 1]), 0LL);
            ans = max(ans, k + f[i]);
        }
        printf("%lld\n", ans);
    }
    return 0;
```

```
22 }
```

P14307 【MX-J27-T4】点灯

题目描述

有一个由 n 座城市构成的国家，其城市之间将由 m 条双向道路互相连接，第 i 条道路连接城市 u_i 和城市 v_i；但由于工程延期，第 i 条道路**只在第 w_i 天及以后开放**。保证这些双向道路两两不同，每条道路连接两个不同的城市，且在所有道路开放后，从城市 1 出发可以到达其余所有城市。

每座城市都设有若干街灯，用于夜间照明。每个夜晚降临后，每位点灯人仅点亮自己所在城市的灯；而日出后，点灯人又会熄灭自己所在城市的灯。初始时，有**充分多的**点灯人在城市 1。这被记作第 0 夜。

为了给国家的每座城市照明，每位点灯人**必须**在每天白天沿城市之间的道路移动。具体地，对每个正整数 t，设第 $t-1$ 夜某位点灯人在城市 x，则他在第 t 天**必须**沿着某条一端为城市 x 且已经开放（即 w 值不超过 t）的道路，随后恰好在第 t 夜到达道路的另一个端点。如果有多条不同的道路，则每位点灯人会独立地随机选择一条；特别地，如果这样的道路不存在，则这位点灯人会失望地离开这个国家。

你想知道是否存在一个非负整数 d，满足在第 d 夜，所有城市内的灯都被点亮；换句话说，在第 d 夜，每个城市内都存在至少一位点灯人。如果存在，你还希望找到符合条件的最小可能的 d。

出于某些原因，给定一个参数 $o \in \{0,1\}$，你只需要在 d 存在时输出 $o \cdot d$ 的值即可。

输入格式

本题有多组测试数据。

第一行，两个整数 c,T，分别表示测试点编号与测试数据组数。接下来输入每组测试数据。样例满足 $c=0$。

对于每组测试数据：

- 第一行，三个正整数 n，m 和 o，分别表示城市数量，道路数量，和给定的参数。
- 接下来 m 行，第 i 行包含三个整数 u_i,v_i,w_i。

保证这些双向道路两两不同，每条道路连接两个不同的城市，且在所有道路开放后，从城市 1 出发可以到达其余所有城市。

输出格式

对于每组测试数据，输出一行一个整数：

- 若存在满足条件的非负整数 d，则输出满足条件的最小可能的 d 与 o 的乘积；
- 若不存在满足条件的非负整数 d，输出 -1。

输入输出样例 #1

输入 #1

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

输出 #1

```
3
-1
```

说明/提示

【样例解释 #1】

对于第一组测试数据：

- 在第 0 夜，只有第 1 个城市存在充分多的点灯人，灯亮的城市为第 1 个城市。
- 在第 1 天，第 1 个城市的点灯人全部移动至城市 3 和 4。注意，点灯人不能移动到城市 2，因为道路 $(1,2)$ 在第 $w=2$ 天后才建设完成。因此，在第 1 夜，灯亮的城市为第 $3,4$ 个城市；由于点灯人数量充分多，所以必然有一些点灯人到达城市 3，而另外一些点灯人到达城市 4。
- 在第 2 天，第 3 个城市的点灯人全部移动到城市 $1,4$，而第 4 个城市的点灯人全部移动到城市 $1,3$。因此，在第 2 夜，灯亮的城市有第 $1,3,4$ 个城市。
- 在第 3 天，第 1 个城市的点灯人全部移动到城市 $2,3,4$，第 3 个城市的点灯人全部移动到城市 $1,4$，而第 4 个城市的点灯人全部移动到城市 $1,3$。因此，在第 3 夜，所有城市的灯都被点亮。

因此，$d=3$，输出 $o\cdot d$ 即 3。

对于第二组测试数据，在第 1 天，城市 1 邻接的所有道路都未开放，因此所有点灯人都无法移动，他们会离开这个国家。因此，不存在符合条件的非负整数 d，输出 -1。

【数据范围】

本题共 25 个测试点，每个 4 分。

对于所有数据，保证：

- $1\le T\le 10$；
- $2\le n\le 2.5\times 10^4$；
- $n-1\le m\le 5\times 10^4$；
- $o\in\{0,1\}$；
- 对所有 $1\le i\le m$，$1\le u_i,v_i\le n$，$u_i\ne v_i$，$1\le w_i\le 10^9$；
- 保证双向道路两两不同；

- 保证在所有道路开放后，从城市 1 出发可以到达其余所有城市。

测试点编号	$n \le$	$m \le$	$o =$	特殊性质
$1 \sim 2$	10	20	1	A
$3 \sim 4$	10^3	2×10^3	^	B
$5 \sim 6$	^	^	0	无
$7 \sim 8$	^	^	1	^
$9 \sim 11$	2.5×10^4	5×10^4	0	B
$12 \sim 14$	^	^	^	无
$15 \sim 16$	^	^	1	B
$17 \sim 19$	10^4	2×10^4	^	C
$20 \sim 21$	2.5×10^4	5×10^4	^	^
$22 \sim 25$	^	^	^	无

- 特殊性质 A：保证 $w_i \le 2 \times 10^5$。
- 特殊性质 B：保证 w_i 全部相等。
- 特殊性质 C：保证非负整数 d 存在。

P14307【MX-J27-T4】点灯

题意

给定一个由 n 个城市和 m 条双向道路组成的图，每条道路有一个开放时间 w_i，表示在第 w_i 天及以后才能使用。点灯人从城市 1 开始，每天必须沿开放的道路移动到一个相邻城市。目标是找到最小的非负整数 d，使得在第 d 夜，每个城市都有至少一个点灯人。如果存在 d，输出 $o \cdot d$；否则输出 -1。

数据范围：T 组测试数据，$2 \le n \le 2.5 \times 10^4$，$n-1 \le m \le 5 \times 10^4$，$o \in \{0,1\}$，$1 \le w_i \le 10^9$。

分析

关键观察：点灯人可以通过在一条边上来回移动来增加路径长度，但不会改变路径长度的奇偶性。因此，对于每个城市，我们只需要关心从城市 1 到达该城市的路径长度的奇偶性。

解决方案：

- 使用分层图模型，将每个城市拆分为两个状态：表示到达该城市时路径长度为偶数的状态和路径长度为奇数的状态。
- 定义 $dis[i][0]$ 为从城市 1 到城市 i 的偶数长度最短路径值，$dis[i][1]$ 为奇数长度最短路径值。

- 通过 Dijkstra 算法计算这些最短路径，考虑边的开放时间限制：当通过一条边时，如果当前路径长度小于边的开放时间 w_i，则需要通过来回移动增加路径长度，直到满足开放时间。具体地，如果当前长度 $t < w_i$，则调整后的长度 $t' = t + 2 \cdot \lceil (w_i - t)/2 \rceil$，确保 $t' \geq w_i$ 且奇偶性不变。
- 计算完所有 $dis[i][0]$ 和 $dis[i][1]$ 后，对于每个城市，我们关心的是能够到达该城市的最小 d，即 d 必须大于等于所有城市所需的最小路径长度。因此，取所有 $dis[i][0]$ 的最大值 max_even 和所有 $dis[i][1]$ 的最大值 max_odd，然后 $d = \min(max\_even, max\_odd)$。
- 如果存在城市无法通过任何路径到达（即 $dis[i][0]$ 和 $dis[i][1]$ 均为无穷大），则无解，输出 -1。
- 此外，需要检查初始条件：如果从城市 1 出发的所有边都要求 $w_i > 0$，则点灯人无法在第一天移动，因此无解。

时间复杂度：使用 Dijkstra 算法，每个节点有两个状态，因此时间复杂度为 $O((n+m)\log n)$。

参考代码

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 25005;
const ll INF = 0x7fffffffll * 100;

struct Edge {
    int v, w;
};

vector<Edge> graph[N];
ll dis[N][2];
bool vis[N][2];

struct Node {
    int u, op;
    ll dist;
    bool operator<(const Node& other) const {
        return dist > other.dist;
    }
};

int main() {
    int c, T;
    scanf("%d %d", &c, &T);
    while (T--) {
        int n, m, o;
        scanf("%d %d %d", &n, &m, &o);
        for (int i = 1; i <= n; i++) {
            graph[i].clear();
            dis[i][0] = dis[i][1] = INF;
            vis[i][0] = vis[i][1] = false;
        }

        bool flag = false;
        for (int i = 0; i < m; i++) {
```

```
            int u, v, w;
            scanf("%d %d %d", &u, &v, &w);
            graph[u].push_back({v, w});
            graph[v].push_back({u, w});
            if ((u == 1 || v == 1) && w == 0) {
                flag = true;
            }
        }

        if (!flag) {
            printf("-1\n");
            continue;
        }

        priority_queue<Node> pq;
        dis[1][0] = 0;
        pq.push({1, 0, 0});

        while (!pq.empty()) {
            Node current = pq.top();
            pq.pop();
            int u = current.u;
            int op = current.op;
            ll dist = current.dist;

            if (vis[u][op]) continue;
            vis[u][op] = true;

            for (const Edge& e : graph[u]) {
                int v = e.v;
                int w = e.w;
                ll new_dist = dist;
                if (new_dist < w) {
                    ll diff = w - new_dist;
                    if (diff % 2 == 0) {
                        new_dist = w;
                    } else {
                        new_dist = w + 1;
                    }
                }
                new_dist += 1;
                int new_op = new_dist % 2;
                if (new_dist < dis[v][new_op]) {
                    dis[v][new_op] = new_dist;
                    pq.push({v, new_op, new_dist});
                }
            }
        }

        ll max_even = 0, max_odd = 0;
        bool possible = true;
        for (int i = 1; i <= n; i++) {
```

```
            if (dis[i][0] == INF && dis[i][1] == INF) {
                possible = false;
                break;
            }
            if (dis[i][0] != INF) {
                max_even = max(max_even, dis[i][0]);
            }
            if (dis[i][1] != INF) {
                max_odd = max(max_odd, dis[i][1]);
            }
        }

        if (!possible) {
            printf("-1\n");
        } else {
            ll d = min(max_even, max_odd);
            printf("%lld\n", o * d);
        }
    }
    return 0;
}
```