Московский Авиационный Институт (национальный исследовательский университет)

Институт информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Журнал по исследовательской практике (индивидуальный план)

Студенты	Группа
Артемьев Дмитрий Иванович	М8О-406Б-18
Белоусов Егор Владимирович	М8О-207Б-20
Инютин Максим Андреевич	М8О-307Б-19
Команда:	MAI #2 Artemiev, Belousov, Inyutin

Москва, 2021

Сводная таблица за осень 2021

Дата	Название	Время	Место проведения	Решенные задачи	Дорешанные задачи
12.09.2021	Grand Prix of Dolgoprudny WA	11:00-16:00	Дистанционно	A, B	C
19.09.2021	Grand Prix of IMO WA	11:00-16:00	Дистанционно	A, B	C
10.10.2021	XXII Открытая Всесибирская олимпиада WA	10:00-15:00	Дистанционно	A, B	C
17.10.2021	ICPC training MAI 21-22 WA	11:00-16:00	Дистанционно	A, B	C
24.10.2021	Grand Prix of Korea !!!	11:00-16:00	Дистанционно	H, G	C
07.11.2021	Grand Prix of Siberia	11:00-16:00	Дистанционно	1, 3, 5, 8	-
14.11.2021	Grand Prix of EDG	11:00-16:00	Дистанционно	A, D, E, G, I	B, F, K
21.11.2021	RuCode 4.0 Div A-B Champoinship	11:00-16:00	Дистанционно	B, K	D
22.11.2021	Div A + B Contest 1	09:00-14:00	Дистанционно	C, D, H, L	F, J, K
26.11.2021	Div A Contest 4: The Korean Contest	09:00-14:00	Дистанционно	B, C, L	D, G
12.12.2021	Grand Prix of Nanjing !!!	11:00-16:00	Дистанционно	A, C, H, M	-
19.12.2021	Moscow Regional Contest	11:00-16:00	Дистанционно	A, D, E, F, G, H, N	-

Явка на контесты

Дата	Название	Присутствующие
12.09.2021	Grand Prix of Dolgoprudny	Артемьев, Белоусов, Инютин
19.09.2021	Grand Prix of IMO	Артемьев, Белоусов, Инютин
10.10.2021	XXII Открытая Всесибирская олимпиада	Артемьев, Белоусов, Инютин
17.10.2021	ICPC training MAI 21-22	Артемьев, Белоусов, Инютин
24.10.2021	Grand Prix of Korea	Артемьев, Белоусов, Инютин
07.11.2021	Grand Prix of Siberia	Артемьев, Белоусов, Инютин
14.11.2021	Grand Prix of EDG	Артемьев, Белоусов, Инютин
21.11.2021	RuCode 4.0 Div A-B Champoinship	Артемьев, Белоусов, Инютин
22.11.2021	Div A + B Contest 1	Артемьев, Белоусов, Инютин
26.11.2021	Div A Contest 4: The Korean Contest	Артемьев, Белоусов, Инютин
12.12.2021	Grand Prix of Nanjing	Артемьев, Белоусов, Инютин
19.12.2021	Moscow Regional Contest	Артемьев, Белоусов, Инютин

Problem G. Nikanor Loves Games

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Nikanor spends all his free time on games. Because of this, he gets bad marks at the university, but that's another story. He also likes gambling. In this problem, we consider the modification of the game called "Orlyanka". There are two players, and each of them has his own coin. Each of the two sides of a coin contains an integer. Players toss their coins, and the winner is the one with the highest number. We can assume that for each coin the probabilities of coming up both sides are equal.

Tonight Nikanor is playing this game with his friends. Nikanor has n friends, and he will play with each of them for a bet of x_i rubles. Fortunately, Nikanor knows that his i-th friend has a coin with the numbers a_i and b_i. If Nikanor wins against his friend, he will receive x_i rubles. Otherwise, he will pay x_i rubles to his friend. If Nikanor and his friend dropped the same value, Nikanor is declared the winner.

Now Nikanor is going to go to the store and buy one coin for all games to maximize his expected profit, taking the coin cost into account. In this shop, a coin with the numbers a and b costs $a \cdot b$ rubles. Nikanor can buy any coin with positive integers.

It's so hard for Nikanor to make the right decision... Nikanor asks you to help him choose a coin so that the expected profit is as high as possible.

Input

The first line contains one integer n ($1 \le n \le 2 \cdot 10^5$) denoting the number of friends.

Each of the following n lines contains three integers a_i, b_i, and x_i ($1 \le a_i, b_i, x_i \le 10^9$) representing the numbers on i-th friend's coin and i-th bet in rubles.

Output

Print a single integer — the maximum expected profit.

Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer will be accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Examples

standard input	standard output
2 1 4 15 3 5 10	2.5000000000
1 2 2 8	4.0000000000

Идея

Очевидно, что нужно всегда брать грани монет, которые присутствуют у оппонентов, т.е. при наличии только одной монеты с значениями 1 и 2 не имеет смысла брать номинал в 3 на одну из граней нашей монеты, так как можно взять 2 и потратить меньше денег. Пока мы не взяли ни одной грани полагаем, что наше мат. ожидание равняется $bad = -\sum_i x_i$, пусть тогда мы взяли на одну из сторон монеты a', а на другую b', тогда наше мат. ожидание равняется $E(a', b') = \sum_{i:a_i \le a'}(\frac{x_i}{2}) + \sum_{i:b_i \le a'}(\frac{x_i}{2}) + \sum_{i:a_i \le b'}(\frac{x_i}{2}) + \sum_{i:b_i \le b'}(\frac{x_i}{2}) + bad - a' \cdot b'$. Закинем все значения a_i и b_i в массив z с коэффициентами $\frac{1}{2}$, отсортируем, посчитаем $pref_i = \sum_{j=0}^{i-1} z_i$. Упростим нашу сумму для подсчета мат. ожидания: $E(a', b') = pref_{i'} + pref_{j'} + bad - a \cdot b$, где $i' : z_{i'} = a'$ и $j' : z_{j'} = b'$. В таком случае мы можем посчитать ответ за $O(n^2 + nlogn) \approx O(n^2)$ перебрав индексы в z. Улучшим решение до $O(nlogn)$. Пусть мы уже выбрали a', тогда мы хотим выбрать b' такое, что $E(a', b') => max$. Т.е. $E(a') = max_{b' \le a'}(a' \cdot b' + pref_{j'}) + bad + pref_{i'}$. Заметим, что $a' \cdot b' + pref_{j'}$ при фиксированной a' является прямой с $k = b'$ и $b = pref_{j'}$. В таком случае можно воспользоваться *Convex hull trick*. Мы будем хранить множество прямых слева направо, так как угол наклона растет, то новую прямую мы будем добавлять справа, а также a' тоже растет, поэтому мы сможем быстро узнавать максимум. Этот прием работает амортизованно за $O(n)$. Итоговая сложность: $O(n + nlogn) \approx O(nlogn)$.

Исходный код

```cpp
#include <bits/stdc++.h>

using namespace std;

#define flush cout.flush

using ll = long long;
using ull = unsigned long long;
using ld = long double;
using pl = pair<ll, ll>;
const ll INF = 1e9 + 7;
const ll mod = 1e9 + 7;
const ll mod2 = 998244353;
const ld eps = 1e-9;
const ld PI = acos(-1);
const ll MAGIC = 32;

int main() {
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    ll n;
    cin >> n;
```

```
    vector<ll> a(n), b(n), x(n);
    for (ll i = 0; i < n; ++i)cin >> a[i] >> b[i] >> x[i];
    vector<pl> z;
    for (ll i = 0; i < n; ++i) {
        z.push_back({a[i], i});
        z.push_back({b[i], i});
    }
    z.push_back({1, -1});
    sort(z.begin(), z.end());
    ll m = z.size();
    vector<ld> pref(m, 0.0);
    ld bad = 0.0;
    for (ll i = 1; i < m; ++i) {
        pref[i] = pref[i - 1] + (ld) x[z[i].second] * 0.5;
        bad += (ld) x[z[i].second] * 0.5;
    }
    ld res = -1e18;
    for (ll i = 0; i < m; ++i) {
        for (ll j = max(0ll, i - MAGIC); j <= i; ++j) {
            ld E = pref[i] + pref[j] - bad;
            E -= (ld) (z[i].first * z[j].first);
            res = max(res, E);
        }
      for (ll j = 0; j <= min(i, MAGIC); ++j) {
            ld E = pref[i] + pref[j] - bad;
            E -= (ld) (z[i].first * z[j].first);
            res = max(res, E);
        }
    }
    cout << fixed << setprecision(20);
    cout << res << "\n";
    return 0;
}
```

Положение команды

№	Участник	A 0/24	B 2/11	C 3/11	D 59/605	E 13/74	F 1/14	G 104/442	H 128/330	I 73/140	J 36/111	K 2/98	Очки	Штраф
104	MAI #2 :	—	—	—	—	—	—	+10 04:22	+12 02:30	-3 02:41	—	—	2	852

Grand Prix of Korea 24.10.2021

Problem J. Periodic Ruler

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

Hitagi has a ruler of infinite length. It has a mark on every integer, where the mark on integer i has color c_i. Each color is represented by an integer from 1 to 100.

She noticed that the ruler's color pattern repeats with a period of t. The period t is defined by the **smallest** positive integer that satisfies $c_i = c_{i+t}$ for all integers i.

Hitagi told Koyomi the colors of n marks of her choice. Koyomi wants to find all positive integers that **cannot** be a period of the ruler, regardless of the colors of unchosen marks. Write a program to find all such numbers, and output their count and sum.

Input

The first line contains a single integer n ($1 \le n \le 50$).

The following n lines each contain two integers x_i ($|x_i| \le 10^9$) and a_i ($1 \le a_i \le 100$). This indicates that the integer x_i is marked with the color a_i.

If $i \neq j$, then $x_i \neq x_j$.

Output

Output two integers on one line. The first integer is the number of positive integers that cannot be the period of the ruler. The second integer is their sum.

Examples

standard input	standard output
3 -1 1 1 2 2 1	2 3
5 1 1 2 1 3 1 4 1 5 1	4 14
1 1000000000 100	0 0

Идея

Напишите сюда что-то

Исходный код

```cpp
#include <bits/stdc++.h>

using namespace std;

#define flush cout.flush

using ll = long long;
using ull = unsigned long long;
using ld = long double;
using pl = pair<ll, ll>;
const ll INF = 1e9 + 7;
const ll mod = 1e9 + 7;
const ll mod2 = 998244353;
const ld eps = 1e-9;
const ld PI = acos(-1);

const ll N = 150;

set<ll> res;


ll k(ll x, ll p) {
    ll rem = ((x % p) + p) % p;
    return (x - rem) / p;
}


void gen(ll x, ll y) {
    ll d = abs(y - x);
    if (d == 0) {
        return;
    }
    vector<ll> divs;
    for (ll i = 1; i * i <= d; ++i) {
        if (d % i == 0) {
            divs.push_back(i);
            if (i != d / i)divs.push_back(d / i);
        }
    }
    for (auto cur : divs) {
        ll k1 = k(x, cur), k2 = k(y, cur);
        if (k2 - k1 == 0 || d % abs(k2 - k1) != 0)continue;
```

```
43          res.insert(cur);
44      }
45  }
46
47  int main() {
48      ios::sync_with_stdio(false);
49      cin.tie(NULL);
50      ll n;
51      cin >> n;
52      vector<vector<ll>> a(N);
53      for (ll i = 0; i < n; ++i) {
54          ll pos, x;
55          cin >> pos >> x;
56          a[x].push_back(pos);
57      }
58      for (ll i = 0; i < N; ++i) {
59          for (ll j = 0; j < i; ++j) {
60              for (auto x : a[i]) {
61                  for (auto y : a[j]) {
62                      gen(x, y);
63                  }
64              }
65          }
66      }
67      for (ll i = 1; i < N; ++i) {
68          vector<ll> have(i, -1);
69          for (ll j = 0; j < N; ++j) {
70              for (auto p : a[j]) {
71                  have[(p % i + i) % i] = j;
72              }
73          }
74          bool ok = true;
75          for (auto x : have) {
76              if (x == -1)ok = false;
77          }
78          if (!ok)continue;
79          vector<ll> cnt(N, 0);
80          for (ll j = 0; j < i; ++j) {
81              cnt[have[j]]++;
82          }
83          for (auto x : cnt) {
84              if (x == 1) {
85                  ok = false;
86              }
87          }
88          if (!ok)continue;
89          set<ll> check;
90          for (ll j = 0; j < i; ++j) {
91              for (ll k = 0; k < j; ++k) {
```

```
92                  if (have[j] == have[k])check.insert(j - k);
93              }
94          }
95          for (auto step : check) {
96              bool ok = true;
97              for (ll j = step; j < i; ++j) {
98                  if (have[j] != have[j - step])ok = false;
99              }
100             if (ok && res.count(step) == 0) {
101                 res.insert(i);
102                 break;
103             }
104         }
105     }
106     cout << res.size() << " ";
107     ll s = 0;
108     for (auto x : res) {
109         s += x;
110     }
111     cout << s << "\n";
112     return 0;
113 }
```

Положение команды

№	Участник	A 47/108	B 7/52	C 64/151	D 4/14	E 44/229	F 1/4	G 18/28	H 70/185	I 2/3	J 68/166	K 34/163	L 2/31	M 4/11	Очки	Штраф
69	MAI #2 Artemiev, Belousov, Inyutin :	—	—	—	—	-6 04:59	—	—	+7 02:29	—	+3 03:43	—	—	—	2	572

Grand Prix of Siberia 07.11.2021

Problem 5. Hockey

Input file:	`input.txt` or *standard input*
Output file:	`output.txt` or *standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Rules described in this problem differ from the conventional hockey rules.

A hockey match lasts 60 minutes, with two teams trying to score as many goals as possible. A hockey team consists of five field players and a goalkeeper.

Penalties are an important part of hockey. A field player can be given a penalty: in this case, the offending player leaves the ice for a period of time which depends on the violation. As the result, the number of players on the ice temporarily decreases for the team the offending player belongs to. There are two types of penalties in hockey: *major* and *minor*. A major penalty means the player leaves the ice for five minutes; with minor penalty, it is two minutes. When penalty time runs out, the player returns to the ice.

A minor penalty can be ended prematurely. A team is said to be playing short-handed when it has less players on the ice than the other team. If a team is playing short-handed and opponent scores a goal, then one of its players with minor penalty returns to the ice with his penalty expired ahead of time. If the team has several players with minor penalty, only the player who got the penalty first returns to the ice. If there are no players with minor penalties in the team, no one returns ahead of time.

Penalties during the game mean that the teams can play in various formats regarding the number of field players on the ice. We will denote the game format by `AxB`, meaning the first team currently has `A` field players on the ice, and the second team has `B`. For instance, in the beginning of the game each team has five players on the ice, and this format is denoted as `5x5`. If the first team currently has two players with penalty, and the second team has one, the format is denoted as `3x4`.

You are given a game protocol, registering the time of all penalties and goals. Calculate which formats happened during the game and for how long each format was played.

Input

The first line of the input file contains an integer N — the number of events in the match ($0 \le N \le 1\,000$).

The following N lines describe the events of the match, one per line. Events are described in the following format:

```
mm:ss.d team type
```

Where `mm:ss.d` — time of event with the precision of tenths of a second ($0 \le$ `mm` ≤ 59, $0 \le$ `ss` ≤ 59, $0 \le$ `d` ≤ 9), `team` — team number (either 1 or 2), `type` — event type:

- `goal` — team scores a goal;
- `minor` — team player receives minor penalty;
- `major` — team player receives major penalty.

It is guaranteed that events of the type `goal` have non-zero decimal of a second, i.e. `d` $\neq 0$, and events of the type `minor` and `major` always have zero decimals of a second, i.e. `d` $= 0$.

Events are listed chronologically, i.e. they are arranged in the order of non-reduction of event times. It is guaranteed that at any moment of time each team has no more than 5 players.

Output

For each format of the game in which the teams have played non-zero time, print the format denotation and the time spent by the teams in this format in a separate line, separated by a space character. The format of time must be exactly the same as the format used in the input data. Lines can be printed in arbitrary order.

Grand Prix of Siberia 07.11.2021

Example

input.txt or standard input	output.txt or standard output
10 06:41.0 1 minor 07:20.4 2 goal 22:22.0 2 minor 22:32.0 1 minor 23:00.1 1 goal 23:12.0 2 minor 23:59.9 1 goal 41:02.0 1 major 41:04.5 2 goal 59:00.0 1 minor	4x3 00:47.9 4x4 01:12.1 4x5 06:39.4 5x4 00:50.0 5x5 50:30.6

Example explanation

The game from the example had the following intervals:

- `[00:00.0; 06:41.0)` — until the first penalty, the game went in the initial format `5x5`;
- `[06:41.0; 07:20.4)` — after a penalty, the teams were playing in the format `4x5` until the first team lost a goal while playing short-handed, and the player removed for a minor penalty returned to the ice;
- `[07:20.4; 22:22.0)` — the teams were playing in full numbers `5x5` until a penalty;
- `[22:22.0; 22:32.0)` — until the next penalty, the teams were playing in the format `5x4`;
- `[22:32.0; 23:00.1)` — until a goal was scored, the teams were playing in the format `4x4`, however, no players returned to the ice after the goal, because it was scored with equally-sized teams;
- `[23:00.1; 23:12.0)` — the teams continued playing in the format `4x4` until another penalty;
- `[23:12.0; 23:59.9)` — after that, the teams were playing in the format `4x3` until a goal was scored, and the second team player who had been penalized at `22:22.0` returned to the ice;
- `[23:59.9; 24:32.0)` — the teams were playing `4x4` until the first team player's penalty ran out;
- `[24:32.0; 25:12.0)` — the teams were playing in the `5x4` format until the second team player's penalty ran out;
- `[25:12.0; 41:02.0)` — the teams were playing in the full format `5x5` until a major penalty;
- `[41:02.0; 41:04.5)` — before the goal, the teams played in the format `4x5`, but because a player of the team that lost a goal had a major penalty, that player does **not** leave the penalty box;
- `[41:04.5; 46:02.0)` — the teams continued playing as `4x5`, until the first player's penalty ran out;
- `[46:02.0; 59:00.0)` — before the penalty, the teams were playing with all players `5x5`;
- `[59:00.0; 60:00.0)` — teams ending the game in the format `4x5`.

Идея

Основная сложность — правильно реализовать систему штрафов в хоккее. Для этого каждую секунду игры. Если в какой-то момент времени на поле произошло событие, то его следует обработать согласно правилам игры. Для этого для каждого игрока обеих команд будем хранить тип штрафа и оставшееся время вне игры, постепенно уменьшая его. Остаётся проверять, сколько игроков на поле и увеличивать время для соответствующего состояния игры. Моделирование работает за константу, поэтому сложность решения $O(60 \cdot 60 \cdot 10 + n) \approx O(n)$.

Исходный код

```
#include <bits/stdc++.h>

int parse_time(const std::string & s) {
    int minutes = std::stoi(s.substr(0, 2));
    int seconds = std::stoi(s.substr(3, 2));
    int mseconds = s[6] - '0';
    return 60 * 10 * minutes + 10 * seconds + mseconds;
}

std::string to_time(int time) {
    int mseconds = time % 10;
    int seconds = (time / 10) % 60;
    int minutes = time / 600;
    std::string s1;
    s1.push_back(minutes / 10 + '0');
    s1.push_back(minutes % 10 + '0');
    std::string s2;
    s2.push_back(seconds / 10 + '0');
    s2.push_back(seconds % 10 + '0');
    std::string s3;
    s3.push_back(mseconds + '0');
    return s1 + ":" + s2 + "." + s3;
}

struct item_t {
    int t;
    int team;
    std::string what;

    friend bool operator < (const item_t & lhs, const item_t & rhs) {
        return lhs.t < rhs.t;
    }
};

int main() {
```

```
36      std::ios::sync_with_stdio(false);
37      std::cout.tie(0);
38      std::cin.tie(0);
39
40      const int MAX_TIME = 60 * 60 * 10;
41
42      int n;
43      std::cin >> n;
44      std::vector<item_t> data;
45      for (int i = 0; i < n; ++i) {
46          std::string tt;
47          int team;
48          std::string s;
49          std::cin >> tt >> team >> s;
50          data.push_back({parse_time(tt), team, s});
51      }
52      std::stable_sort(data.begin(), data.end());
53
54      const int TN = 5;
55      const int PLAY = 0;
56      const int MINOR = 1;
57      const int MAJOR = 2;
58      const int MINOR_TIME = 2 * 60 * 10;
59      const int MAJOR_TIME = 5 * 60 * 10;
60      std::vector<int> penalty_a(TN);
61      std::vector<int> penalty_ta(TN);
62      std::vector<int> penalty_b(TN);
63      std::vector<int> penalty_tb(TN);
64      std::vector< std::vector<int> > res(TN + 1, std::vector<int>(TN + 1));
65      int j = 0;
66      for (int i = 0; i < MAX_TIME; ++i) {
67
68          for (int ii = 0; ii < TN; ++ii) {
69              if (penalty_ta[ii] != PLAY) {
70                  --penalty_a[ii];
71                  if (penalty_a[ii] == 0) {
72                      penalty_ta[ii] = PLAY;
73                  }
74              }
75              if (penalty_tb[ii] != PLAY) {
76                  --penalty_b[ii];
77                  if (penalty_b[ii] == 0) {
78                      penalty_tb[ii] = PLAY;
79                  }
80              }
81          }
82
83          while (j < n and data[j].t <= i) {
84              item_t event = data[j];
```

```
85              if (event.team == 1) {
86                  if (event.what == "major") {
87                      for (int ii = 0; ii < TN; ++ii) {
88                          if (penalty_ta[ii] == PLAY) {
89                              penalty_ta[ii] = MAJOR;
90                              penalty_a[ii] = MAJOR_TIME;
91                              break;
92                          }
93                      }
94                  }
95
96                  if (event.what == "minor") {
97                      for (int ii = 0; ii < TN; ++ii) {
98                          if (penalty_ta[ii] == PLAY) {
99                              penalty_ta[ii] = MINOR;
100                             penalty_a[ii] = MINOR_TIME;
101                             break;
102                         }
103                     }
104                 }
105
106                 if (event.what == "goal") {
107                     int count_a = 0;
108                     int count_b = 0;
109                     for (int ii = 0; ii < TN; ++ii) {
110                         if (penalty_ta[ii] == PLAY) {
111                             ++count_a;
112                         }
113                         if (penalty_tb[ii] == PLAY) {
114                             ++count_b;
115                         }
116                     }
117                     if (count_b < count_a) {
118                         int min_time_left = -1;
119                         for (int ii = 0; ii < TN; ++ii) {
120                             if (penalty_tb[ii] == MINOR) {
121                                 if (min_time_left == -1 or penalty_b[ii] < penalty_b[
                                        min_time_left]) {
122                                     min_time_left = ii;
123                                 }
124                             }
125                         }
126                         if (min_time_left != -1) {
127                             penalty_tb[min_time_left] = PLAY;
128                             penalty_b[min_time_left] = 0;
129                         }
130                     }
131                 }
132             }
```

```
133
134             if (event.team == 2) {
135                 if (event.what == "major") {
136                     for (int ii = 0; ii < TN; ++ii) {
137                         if (penalty_tb[ii] == PLAY) {
138                             penalty_tb[ii] = MAJOR;
139                             penalty_b[ii] = MAJOR_TIME;
140                             break;
141                         }
142                     }
143                 }
144
145                 if (event.what == "minor") {
146                     for (int ii = 0; ii < TN; ++ii) {
147                         if (penalty_tb[ii] == PLAY) {
148                             penalty_tb[ii] = MINOR;
149                             penalty_b[ii] = MINOR_TIME;
150                             break;
151                         }
152                     }
153                 }
154
155                 if (event.what == "goal") {
156                     int count_a = 0;
157                     int count_b = 0;
158                     for (int ii = 0; ii < TN; ++ii) {
159                         if (penalty_ta[ii] == PLAY) {
160                             ++count_a;
161                         }
162                         if (penalty_tb[ii] == PLAY) {
163                             ++count_b;
164                         }
165                     }
166                     if (count_a < count_b) {
167                         int min_time_left = -1;
168                         for (int ii = 0; ii < TN; ++ii) {
169                             if (penalty_ta[ii] == MINOR) {
170                                 if (min_time_left == -1 or penalty_a[ii] < penalty_a[
                                        min_time_left]) {
171                                     min_time_left = ii;
172                                 }
173                             }
174                         }
175                         if (min_time_left != -1) {
176                             penalty_ta[min_time_left] = PLAY;
177                             penalty_a[min_time_left] = 0;
178                         }
179                     }
180                 }
```

```
            }
            ++j;
        }
        {
            int count_a = 0;
            int count_b = 0;
            for (int ii = 0; ii < TN; ++ii) {
                if (penalty_ta[ii] == PLAY) {
                    ++count_a;
                }
                if (penalty_tb[ii] == PLAY) {
                    ++count_b;
                }
            }
            ++res[count_a][count_b];
        }
    }

    for (int i = 0; i < TN + 1; ++i) {
        for (int j = 0; j < TN + 1; ++j) {
            if (res[i][j] > 0) {
                std::cout << i << 'x' << j << " " + to_time(res[i][j]) << '\n';
            }
        }
    }
}
```

Положение команды

№	Участник	Я	1 52/631	2 26/214	3 70/70	4 8/35	5 65/151	6 58/165	7 43/114	8 69/174	9 47/194	10 33/198	11 5/12	Очки	Штраф
57	MAI #2 Artemiev, Belousov, Inyutin		+10 02:11	—	+ 00:14	—	+3 02:20	-16 04:56	-5 04:59	+1 00:33	—	-2 03:54	—	4	600

Grand Prix of EDG 14.11.2021

Problem B. A Plus B Problem

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	512 mebibytes

JB gets a machine that can solve "A Plus B Problem" and feels curious about the mechanism. He hears that you are proficient in competitive programming and have learned many advanced data structures and algorithms such as Link-Cut tree, Lagrange Inversion formula, Sweepline Mo, and so on. Hence, he asks you to help implement a program that can solve "A Plus B Problem" as same as the machine.

The machine consists of $3 \times n$ digits. The digits of the first two rows can be changed arbitrarily, and the third row always equals the decimal sum of the first two rows. The third row only consists of the lowest n digits even if the sum exceeds n digits.

For example, when $n = 5$, the three rows can be "01234", "56789", "58023" or "56789", "58023", "14812".

To test your function, you are given q queries. In the i-th query, the c_i-th digit of the r_i-th row is updated to d_i (the digit may not change). Because the digits are too many and JB has no time to check your answer, he only asks you to find the c_i-th digit of the third row after the query and how many digits of the machine change in the query.

Input

The first line contains two integers n and q ($1 \le n, q \le 10^6$).

The second line contains a string consisting of n digits, representing the first row of the machine.

The third line contains a string consisting of n digits, representing the second row of the machine.

There are q lines in the following. The i-th of the following line consists of three integers r_i, c_i and d_i ($1 \le r_i \le 2$, $1 \le c_i \le n$, $0 \le d_i \le 9$).

Output

Output q lines. In the i-th line, output two integers - the c_i-th digit of the third row after the query and how many digits of the machine change in the query.

Example

standard input	standard output
5 5 01234 56789 2 1 0 2 2 1 2 3 2 2 4 3 2 5 4	0 2 3 2 5 3 7 3 8 3

Note

In the example, the initial rows are "01234", "56789", "58023".

After the 1-st query, the rows are "01234", "06789", "08023".

After the 2-nd query, the rows are "01234", "01789", "03023".

After the 3-th query, the rows are "01234", "01289", "02523".

After the 4-th query, the rows are "01234", "01239", "02473".

Идея

Тривиальный случай, когда изменяется только одна цифра суммы, нам не очень интересен, так как изменяется всего две цифры во всех числах. Гораздо более сложный случай — замена девяток на нули и нулей на девятки при изменении суммы. Заметим, что для какого-то префикса числа достаточно знать количество цифр «0» и «9», чтобы корректно отвечать на запрос.

Используем струкуру данных, поддерживающую модификацию на отрезке для эффективного ответа на запрос и изменение. Во время контеста было реализовано решение с использованием Декартова дерева, которое не прошло по времени из-за большой константы. При дорешивании было реализовано решение, использующее дерево отрезков.

Сложность операций с деревом $O(\log n)$. Асимптотика решения $O(n \cdot \log n + q \cdot \log n)$.

Исходный код

```
#include <bits/stdc++.h>

#ifndef TREAP_HPP
#define TREAP_HPP

#include <ctime>
#include <iostream>

class treap_t {
    static const int TAB_SIZE = 4;

    struct treap_node_t {
        treap_node_t * _left;
        treap_node_t * _right;
        int _key;
        int _priority;

        int _value;
        int _delay;
        int _suf0;
        int _suf9;

        treap_node_t(int key, int num) {
            _left = nullptr;
            _right = nullptr;
            _key = key;
            _priority = std::rand();
            _value = num;
            _suf0 = 0;
            _suf9 = 0;
```

```
31            if (num == 0) {
32                ++_suf0;
33            }
34            if (num == 9) {
35                ++_suf9;
36            }
37            _delay = -1;
38        }
39    };
40
41    using treap_ptr = treap_node_t *;
42
43    treap_ptr root;
44
45    int get_key(treap_ptr t) {
46        if (t != nullptr) {
47            return t->_key;
48        } else {
49            return 0;
50        }
51    }
52
53    int get_value(treap_ptr t) {
54        if (t != nullptr) {
55            return t->_value;
56        } else {
57            return 0;
58        }
59    }
60
61    int get_suf0(treap_ptr t) {
62        if (t != nullptr) {
63            if (t->_delay != -1) {
64                return t->_delay == 0 ? t->_key : 0;
65            } else {
66                return t->_suf0;
67            }
68        } else {
69            return 0;
70        }
71    }
72
73    int get_suf9(treap_ptr t) {
74        if (t != nullptr) {
75            if (t->_delay != -1) {
76                return t->_delay == 9 ? t->_key : 0;
77            } else {
78                return t->_suf9;
79            }
```

```
80          } else {
81              return 0;
82          }
83      }
84
85      void enqueue_delay(treap_ptr t, int delay_value) {
86          if (t != nullptr) {
87              t->_delay = delay_value;
88          }
89      }
90
91      void update(treap_ptr t) {
92          if (t != nullptr) {
93              t->_key = 1 + get_key(t->_left) + get_key(t->_right);
94              int delay_val = t->_delay;
95              if (delay_val != -1) {
96                  t->_value = delay_val;
97                  t->_delay = -1;
98                  t->_suf0 = 0;
99                  t->_suf9 = 0;
100                 if (delay_val == 0) {
101                     t->_suf0 = t->_key;
102                 }
103                 if (delay_val == 9) {
104                     t->_suf9 = t->_key;
105                 }
106                 enqueue_delay(t->_left, delay_val);
107                 enqueue_delay(t->_right, delay_val);
108             } else {
109                 if (get_key(t->_right) == get_suf0(t->_right)) {
110                     if (t->_value == 0) {
111                         t->_suf0 = get_suf0(t->_left) + get_suf0(t->_right) + 1;
112                     } else {
113                         t->_suf0 = get_suf0(t->_right);
114                     }
115                 } else {
116                     t->_suf0 = get_suf0(t->_right);
117                 }
118
119                 if (get_key(t->_right) == get_suf9(t->_right)) {
120                     if (t->_value == 9) {
121                         t->_suf9 = get_suf9(t->_left) + get_suf9(t->_right) + 1;
122                     } else {
123                         t->_suf9 = get_suf9(t->_right);
124                     }
125                 } else {
126                     t->_suf9 = get_suf9(t->_right);
127                 }
128             }
```

```
129         }
130     }
131
132     void split(treap_ptr t, int key, treap_ptr & l, treap_ptr & r) {
133         if (t == nullptr) {
134             l = nullptr;
135             r = nullptr;
136             return;
137         }
138         update(t);
139         int key_left_subtree = get_key(t->_left);
140         if (key > key_left_subtree + 1) {
141             split(t->_right, key - key_left_subtree - 1, t->_right, r);
142             l = t;
143         } else {
144             split(t->_left, key, l, t->_left);
145             r = t;
146         }
147         update(l);
148         update(r);
149     }
150
151     treap_ptr merge(treap_ptr l, treap_ptr r) {
152         if (l == nullptr) {
153             return r;
154         }
155         if (r == nullptr) {
156             return l;
157         }
158         update(l);
159         update(r);
160         if (l->_priority > r->_priority) {
161             l->_right = merge(l->_right, r);
162             update(l);
163             return l;
164         } else {
165             r->_left = merge(l, r->_left);
166             update(r);
167             return r;
168         }
169     }
170
171     void delete_treap(treap_ptr t) {
172         if (t != nullptr) {
173             delete_treap(t->_left);
174             delete_treap(t->_right);
175             delete t;
176         }
177     }
```

```
178
179
180     friend std::ostream & operator << (std::ostream & out, treap_t t) {
181         print_treap(out, t.root, 0);
182         return out;
183     }
184
185     friend std::ostream & operator << (std::ostream & out, treap_ptr t) {
186         print_treap(out, t, 0);
187         return out;
188     }
189
190     friend void print_treap(std::ostream & out, treap_ptr t, int h) {
191         if (t != nullptr) {
192             print_treap(out, t->_left, h + 1);
193             for (int i = 0; i < TAB_SIZE * h; ++i) {
194                 out << " ";
195             }
196             out << "{ x = " << t->_key << ", val = " << t->_value << ", suf0 = " << t->
                    _suf0 << ", suf9 = " << t->_suf9 << ", delay = " << t->_delay << "}\n";
197             print_treap(out, t->_right, h + 1);
198         }
199     }
200
201 public:
202     treap_t() {
203         std::srand(std::time(NULL));
204         root = nullptr;
205     }
206
207     void enqueue(int key_l, int key_r, int value) {
208         treap_ptr l = nullptr;
209         treap_ptr m = nullptr;
210         treap_ptr r = nullptr;
211         split(root, key_r + 1, m, r);
212         split(m, key_l, l, m);
213         enqueue_delay(m, value);
214         root = merge(merge(l, m), r);
215     }
216
217     int get(int key_l, int key_r) {
218         treap_ptr l = nullptr;
219         treap_ptr m = nullptr;
220         treap_ptr r = nullptr;
221         split(root, key_r + 1, m, r);
222         split(m, key_l, l, m);
223         int ans = get_value(m);
224         root = merge(merge(l, m), r);
225         return ans;
```

```
226     }
227 
228     int get0(int key_l, int key_r) {
229         treap_ptr l = nullptr;
230         treap_ptr m = nullptr;
231         treap_ptr r = nullptr;
232         split(root, key_r + 1, m, r);
233         split(m, key_l, l, m);
234         int ans = get_suf0(m);
235         root = merge(merge(l, m), r);
236         return ans;
237     }
238 
239     int get9(int key_l, int key_r) {
240         treap_ptr l = nullptr;
241         treap_ptr m = nullptr;
242         treap_ptr r = nullptr;
243         split(root, key_r + 1, m, r);
244         split(m, key_l, l, m);
245         int ans = get_suf9(m);
246         root = merge(merge(l, m), r);
247         return ans;
248     }
249 
250     void push_back(int value) {
251         treap_ptr m = new treap_node_t(1, value);
252         root = merge(root, m);
253     }
254 
255     void push_front(int value) {
256         treap_ptr m = new treap_node_t(1, value);
257         root = merge(m, root);
258     }
259 
260     void destruct() {
261         delete_treap(root);
262     }
263 };
264 
265 #endif /* TREAP_HPP */
266 
267 int main() {
268     std::ios::sync_with_stdio(false);
269     std::cout.tie(0);
270     std::cin.tie(0);
271 
272     int n, q;
273     std::cin >> n >> q;
274     std::string str1, str2;
```

```
    std::cin >> str1 >> str2;
    std::vector<int> s1(n);
    std::vector<int> s2(n);
    for (int i = 0; i < n; ++i) {
        s1[i] = str1[i] - '0';
        s2[i] = str2[i] - '0';
    }
    std::vector<int> arr(n);
    int remainder = 0;
    for (int i = n - 1; i >= 0; --i) {
        arr[i] = s1[i] + s2[i] + remainder;
        remainder = 0;
        if (arr[i] > 9) {
            arr[i] = arr[i] - 10;
            remainder = 1;
        }
    }
    treap_t t;
    for (int i = 0; i < n; ++i) {
        t.push_back(arr[i]);
    }
    // std::cout << t << "------" << std::endl;
    while (q--) {
        int r, c, d;
        std::cin >> r >> c >> d;
        // std::cout << "query " << r << ", " << c << ", " << d << '\n';
        bool dec = true;
        int num_was = t.get(c, c);
        int delta = 0;
        if (r == 1) {
            if (s1[c - 1] == d) {
                std::cout << num_was << " 0\n";
                continue;
            }
            delta = std::abs(s1[c - 1] - d);
            if (s1[c - 1] < d) {
                dec = false;
            }
            s1[c - 1] = d;
        }
        if (r == 2) {
            if (s2[c - 1] == d) {
                std::cout << num_was << " 0\n";
                continue;
            }
            delta = std::abs(s2[c - 1] - d);
            if (s2[c - 1] < d) {
                dec = false;
            }
```

```
            s2[c - 1] = d;
        }
        int ans = 1;
        if (dec) {
            int num_next = num_was - delta;
            if (num_next < 0) {
                int suf = t.get0(1, c - 1);
                int pos = c - 1 - suf;
                t.enqueue(c, c, num_next + 10);
                // std::cout << "dec suf = " << suf << ", pos = " << pos << ", num_next
                    = " << num_next << std::endl;
                if (suf == c - 1) {
                    t.enqueue(1, c - 1, 9);
                    ans = ans + suf + 1;
                } else {
                    t.enqueue(pos + 1, c - 1, 9);
                    // std::cout << t << std::endl;
                    t.enqueue(pos, pos, t.get(pos, pos) - 1);
                    ans = ans + suf + 2;
                }
                // std::cout << t << std::endl;
            } else {
                t.enqueue(c, c, num_next);
                ++ans;
            }
        } else {
            int num_next = num_was + delta;
            if (num_next > 9) {
                int suf = t.get9(1, c - 1);
                int pos = c - 1 - suf;
                t.enqueue(c, c, num_next - 10);
                // std::cout << "inc suf = " << suf << ", pos = " << pos << ", num_next
                    = " << num_next << std::endl;
                // std::cout << t << std::endl;
                if (suf == c - 1) {
                    t.enqueue(1, c - 1, 0);
                    ans = ans + suf + 1;
                } else {
                    t.enqueue(pos + 1, c - 1, 0);
                    t.enqueue(pos, pos, 1 + t.get(pos, pos));
                    ans = ans + suf + 2;
                }
                // std::cout << t << std::endl;
            } else {
                t.enqueue(c, c, num_next);
                ++ans;
            }
        }
        // std::cout << t << std::endl;
```

```
371         std::cout << t.get(c, c) << ' ' << ans << '\n';
372
373         // std::cout << t << "------" << std::endl;
374     }
375 }
```

Положение команды

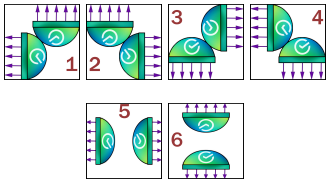
№	Участник	A	B	C	D	E	F	G	H	I	J	K	L	Очки	Штраф
		86/104	58/286	1/42	67/171	84/173	51/290	85/152	9/35	86/92	43/174	50/266	5/54		
74	MAI #2 Artemiev, Belousov, Inyutin	+2 00:28	-4 04:59	—	+1 02:40	+1 01:50	—	+ 01:13	—	+ 00:11	—	—	—	5	463

Задача B. Склады СберМаркет

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	2 секунды
Ограничение по памяти:	512 мегабайт

Поддержка работы крупного онлайн-магазина, доставляющего товары по всей стране, такого, как СберМаркет, требует организации огромных складов. Склады расположены под землёй и обслуживаются автоматическими тележками. На некоторых участках тележки должны подниматься на поверхность. Для этого устанавливаются специальные впуско-выпускные ворота.

На участке 1×1 километр допустимы следующие конфигурации установки ворот:



Для СберМаркета оборудуется новый центральный склад. Территория склада представляет собой огороженный прямоугольник w на h километров. В каждом секторе размера 1×1 километр **нужно** разместить одну из 6 конфигураций ворот.

Из стороны, где располагаются ворота, проведены фиолетовые стрелочки.

Автоматические тележки двигаются строго по прямой от одних ворот до других; в случае встречи двух тележек, едущих навстречу, они разъезжаются с помощью системы предотвращения столкновений и продолжают двигаться по той же прямой каждая в прежнем направлении.

Автоматические тележки могут выезжать из любых ворот и въезжать в любые ворота; также в конфигурациях 5 и 6 тележка может проехать между воротами по вертикали для конфигурации 5 и по горизонтали для конфигурации 6.

Если тележка попадает в ворота с обратной стороны или врезается в ограждение склада, происходит авария.

Требуется указать для каждого сектора склада, какая из 6 конфигураций ворот должна в нём располагаться, чтобы аварии были невозможны.

Формат входных данных

Первая строка входных данных содержит два целых числа w и h ($1 \leqslant w, h \leqslant 10^3$) — ширина и высота склада СберМаркета.

Формат выходных данных

Выведите h строк, в каждой строке w чисел в диапазоне от 1 до 6, разделённые пробелами — номера конфигураций ворот в соответствующих квадратах склада. Если ответов несколько, выведите любой.

Если расположить ворота без риска аварий невозможно, выведите -1.

Примеры

стандартный ввод	стандартный вывод
2 3	3 4 6 6 2 1
4 4	3 5 5 4 6 3 4 6 6 2 1 6 2 5 5 1

Идея

Решение задачи полностью конструктивное, полностью описано в программе. Самый сложный случай — при нечётных ширине или высоте складов СберМаркета. Асимптотика $O(w \cdot h)$.

Исходный код

```
#include <bits/stdc++.h>

using grid_t = std::vector< std::vector<int> >;

void draw_sq(grid_t & a, int x1, int y1, int x2, int y2) {
    a[x1][y1] = 3;
    a[x2][y1] = 2;
    a[x1][y2] = 4;
    a[x2][y2] = 1;
    for (int i = x1 + 1; i < x2; ++i) {
        a[i][y1] = 6;
        a[i][y2] = 6;
    }
    for (int i = y1 + 1; i < y2; ++i) {
        a[x1][i] = 5;
        a[x2][i] = 5;
    }
}

int rot[7] = {0, 1, 4, 3, 2, 6, 5};

grid_t transpose(const grid_t & gr) {
    int h = gr.size();
    int w = gr.back().size();
    grid_t res(w, std::vector<int>(h));
    for (int i = 0; i < w; ++i) {
        for (int j = 0; j < h; ++j) {
            res[i][j] = rot[gr[j][i]];
        }
    }
    return res;
}

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

    int w, h;
    std::cin >> w >> h;
```

```
41      grid_t ans(h, std::vector<int>(w));
42      if (w == 1 or h == 1) {
43          std::cout << "-1\n";
44          return 0;
45      }
46      if (w == 2 or h == 2) {
47          draw_sq(ans, 0, 0, h - 1, w - 1);
48      } else if (w & 1) {
49          if (h & 1) {
50              if (h == 3 and w == 3) {
51                  std::cout << "-1\n";
52                  return 0;
53              }
54              bool flag = false;
55              if (h == 3) {
56                  std::swap(h, w);
57                  flag = true;
58                  ans = transpose(ans);
59              }
60              ans[0][0] = 3;
61              ans[0][1] = 5;
62              for (int i = 2; i < w - 1; ++i) {
63                  ans[0][i] = (i & 1 ? 3 : 4);
64              }
65              ans[0].back() = 4;
66              ans[1][0] = 2;
67              ans[1][1] = 4;
68              for (int i = 2; i < w; ++i) {
69                  ans[1][i] = 6;
70              }
71              for (int j = 2; j < h - 2; ++j) {
72                  ans[j][0] = (j & 1 ? 2 : 3);
73                  for (int i = 1; i < w - 1; ++i) {
74                      ans[j][i] = 6;
75                  }
76                  ans[j].back() = (j & 1 ? 4 : 1);
77              }
78              ans[h - 2][w - 1] = 4;
79              ans[h - 2][w - 2] = 2;
80              for (int i = 0; i < w - 2; ++i) {
81                  ans[h - 2][i] = 6;
82              }
83              ans.back()[0] = 2;
84              for (int i = 1; i < w - 2; ++i) {
85                  ans.back()[i] = (i & 1 ? 1 : 2);
86              }
87              ans.back()[w - 2] = 5;
88              ans.back()[w - 1] = 1;
89              if (flag) {
```

```
90                  std::swap(h, w);
91                  ans = transpose(ans);
92              }
93          } else {
94              for (int i = 0; i < std::min(h / 2, w / 2); ++i) {
95                  draw_sq(ans, i, i + 1, h - 1 - i, w - 1 - i);
96              }
97              ans[0][0] = 3;
98              ans[0][1] = 5;
99              for (int i = 1; i < h - 1; i = i + 2) {
100                 ans[i][0] = 2;
101                 ans[i][1] = 4;
102                 ans[i + 1][0] = 3;
103                 ans[i + 1][1] = 1;
104             }
105             ans.back()[0] = 2;
106             ans.back()[1] = 5;
107         }
108     } else {
109         if (h & 1) {
110             for (int i = 0; i < std::min(h / 2, w / 2); ++i) {
111                 draw_sq(ans, i + 1, i, h - 1 - i, w - 1 - i);
112             }
113             ans[0][0] = 3;
114             ans[1][0] = 6;
115             for (int i = 1; i < w - 1; i = i + 2) {
116                 ans[0][i] = 4;
117                 ans[1][i] = 2;
118                 ans[0][i + 1] = 3;
119                 ans[1][i + 1] = 1;
120             }
121             ans[0].back() = 4;
122             ans[1].back() = 6;
123         } else {
124             for (int i = 0; i < std::min(h / 2, w / 2); ++i) {
125                 draw_sq(ans, i, i, h - 1 - i, w - 1 - i);
126             }
127         }
128     }
129     for (int i = 0; i < h; ++i) {
130         for (int j = 0; j < w; ++j) {
131             std::cout << ans[i][j] << ' ';
132         }
133         std::cout << '\n';
134     }
135 }
```

Положение команды

№	Участник Я	A 4/53	B 24/221	C 10/52	D 29/44	E 1/2	F 37/158	G 5/8	H 4/133	I 30/122	J 41/196	K 59/240	L 41/129	Очки	Штраф
50	Moscow AI #2 (Maxim Inyutin, Egor Belousov, Dmitry Artemyev)	—	+4 02:54	—	—	—	—	—	—	-12 04:38	—	+4 01:53	—	2	448

Problem F. Fence

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

Bajtazar is widely believed to be the greatest scrooge in the whole borough. One can find many examples to support this claim, and the least important one of them is that his estate has not even got a fence. However, Bajtazar has recently found n old planks in his basement, so he decided to build at least a fragment of a hoarding.

He stacked the planks one over the other such that their consecutive lengths were $a_1, \ldots, a_n$. He took the first one, cut out a fragment of length b, and nailed it as the leftmost piece of the fence. Then, he cut out the next fragment of length b, and nailed it next to the previous one. He continued doing so until what was left in his hands was a piece of length c ($1 \leq c \leq b$). *Well, this one might seem a little bit too short, but it would be a pity if such a good plank went to waste*, he thought... and added it to the fence as well. He then took the second plank from the stack, then the next one, and repeated the whole procedure for each one of them.

When the job was done, Bajtazar looked at the result... and concluded that using those shorter pieces might really not have been the best idea. *This doesn't look like a cohesive design at all* – he thought – and decided to paint the whole fence white to give it at least a pretence of consistency. *Still*, it occurred to him suddenly, *if I only paint every other plank white, and leave the remaining ones brown, I could use (roughly) twice as little paint and yet the fence will still look like a well thought-out, coherent construction!* And so he only painted every second plank, starting from the leftmost one (apparently, the rumours of Bajtazar's meanness must have been somewhat exaggerated. He could have started from the second leftmost plank after all.).

However, in the evening, a frightening thought struck him: maybe if he had chosen a different value of b, the overall amount of paint used could have been smaller? Well, there's not much that can be done anymore, but just the thought of such an unnecessary wastage keeps Bajtazar awake – he needs to know how much paint he would have used if he had chosen to build a fence of any other possible height. Help him to find the answer so he can finally fall asleep peacefully (or not, depending on the result of your computation).

Input

The first line of input contains the number of test cases z ($1 \leq z \leq 5$).

The descriptions of the test cases follow.

The first line of every test case contains a single integer n ($1 \leq n \leq 1\,000\,000$) – the number of planks. Then follow n integers a_i ($1 \leq a_i \leq 1\,000\,000$, $\sum_{i=1}^{n} a_i \leq 1\,000\,000$) – the lengths of consecutive planks.

Output

For each test case, output M lines, where M is the maximum of all values of a_i within this test case. The i-th line should contain a single integer f_i: the total length of planks which Bajtazar would have needed to paint white if he had chosen the fence height to be $b = i$.

Div A + B Contest 1 22.11.2021

Example

standard input	standard output
1 4 10 7 2 8	14 13 15 13 15 16 21 23 24 12

Note

For height $b = 4$, consecutive fence pickets would have heights:

4 4 2 4 3 2 4 4.

Bajtazar would have needed to paint the total length of $4 + 2 + 3 + 4$, so the answer in the 4-th line is 13.

For height $b = 5$, consecutive fence pickets would have heights:

5 5 5 2 2 5 3.

Bajtazar would have needed to paint the total length of $5 + 5 + 2 + 3$, so the answer in the 5-th line is 15.

Идея

Для решения задачи построим дерево отрезков на сумму чисел, стоящих на чётных и нечётных позициях для всего забора. Будет обновлять каждый фрагмент забора и выводить требуемую сумму. Так как суммарно обновлений в дереве будет не более $\sum a_i \leqslant 10^6$ и $n \leqslant 10^6$, то итоговая временная сложность решения $O(n \cdot \log n)$.

Исходный код

```
#include <bits/stdc++.h>

using pii = std::pair<int, int>;

class segment_tree_t {
private:
    struct item_t {
        int64_t even;
        int64_t odd;
        int n;

        item_t() = default;

        item_t(int64_t x, int64_t b) {
            if (x % b == 0) {
                n = x / b;
                odd = b * (n / 2);
                even = b * (n / 2);
                if (n & 1) {
                    even += b;
                }
            } else {
                n = (x + b - 1) / b;
                odd = b * (n / 2);
                even = b * (n / 2);
                if (n & 1) {
                    even = even + x % b;
                } else {
                    odd = odd - b + x % b;
                }
            }
            // std::cout << "x = " << x << ", b = " << b << ", odd = " << odd << ",
                even = " << even << ", n = " << n << std::endl;
        }

        friend item_t operator + (const item_t & lhs, const item_t & rhs) {
            item_t res;
            res.n = lhs.n + rhs.n;
            if (lhs.n & 1) {
```

```
39                  res.even = lhs.even + rhs.odd;
40                  res.odd = lhs.odd + rhs.even;
41              } else {
42                  res.even = lhs.even + rhs.even;
43                  res.odd = lhs.odd + rhs.odd;
44              }
45              return res;
46          }
47      };
48
49      std::vector<item_t> data;
50      int n;
51
52  public:
53      segment_tree_t(int _n) {
54          n = _n;
55          data.resize(4 * n);
56      }
57
58      void set(int p, int64_t x, int64_t b) {
59          set(1, 1, n, p, x, b);
60      }
61
62      void set(int id, int l, int r, int p, int64_t x, int64_t b) {
63          if (l >= r) {
64              data[id] = item_t(x, b);
65              return;
66          }
67          int m = (l + r) / 2;
68          if (p <= m) {
69              set(id * 2, l, m, p, x, b);
70          } else {
71              set(id * 2 + 1, m + 1, r, p, x, b);
72          }
73          data[id] = data[id * 2] + data[id * 2 + 1];
74      }
75
76      int64_t get(int l, int r) {
77          item_t ans = get(1, 1, n, l, r);
78          return ans.even;
79      }
80
81      item_t get(int id, int l, int r, int ql, int qr) {
82          if (ql <= l and r <= qr) {
83              return data[id];
84          }
85          int m = (l + r) / 2;
86          if (qr <= m) {
87              return get(id * 2, l, m, ql, qr);
```

```
88          }
89          if (ql > m) {
90              return get(id * 2 + 1, m + 1, r, ql, qr);
91          }
92          return get(id * 2, l, m, ql, qr) + get(id * 2 + 1, m + 1, r, ql, qr);
93      }
94  };
95
96  int main() {
97      std::ios::sync_with_stdio(false);
98      std::cin.tie(0);
99
100     int t;
101     std::cin >> t;
102     while (t--) {
103         int n;
104         std::cin >> n;
105         using item_t = std::pair<int64_t, int>;
106         std::list<item_t> lst;
107         int64_t max_a = 0;
108         for (int i = 0; i < n; ++i) {
109             int64_t a;
110             std::cin >> a;
111             max_a = std::max(a, max_a);
112             lst.push_back({a, i + 1});
113         }
114         segment_tree_t st(n);
115         using iter = std::list<item_t>::iterator;
116         for (int b = 1; b <= max_a; ++b) {
117             for (iter it = lst.begin(); it != lst.end();) {
118                 iter next = it;
119                 ++next;
120                 item_t elem = *it;
121                 if (elem.first < b) {
122                     lst.erase(it);
123                 } else {
124                     st.set(elem.second, elem.first, b);
125                 }
126                 it = next;
127             }
128             std::cout << st.get(1, n) << '\n';
129         }
130     }
131 }
```

Положение команды

№	Участник Я	A1 18/67	B1 1/8	C1 44/131	D1 47/102	E1 8/32	F1 33/59	G1 12/30	H1 58/66	I1 15/97	J1 47/134	K1 42/278	L1 43/268	M1 0/0	Очки	Штраф
18	Moscow AI #2 (Maxim Inyutin, Egor Belousov, Dmitry Artemyev)	—	—	+2 6д. 20ч.	+1 6д. 17ч.	—	+ 6д. 21ч.	—	+ 4д. 18ч.	—	+1 4д. 16ч.	+2 4д. 18ч.	+2 6д. 19ч.	—	36	227718

Div A Contest 4: The Korean Contest 26.11.2021

Problem L. Trio

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	1024 mebibytes

Let A be any set of n natural numbers whose decimal representations consist of exactly four digits without 0 in any decimal place.

A *trio* is a set of three numbers $\{a, b, c\}$ chosen from A such that the following conditions are fulfilled simultaneously:

- The ones decimals of three numbers a, b, c are either all equal or all distinct.
- The tens decimals of three numbers a, b, c are either all equal or all distinct.
- The hundreds decimals of three numbers a, b, c are either all equal or all distinct.
- The thousands decimals of three numbers a, b, c are either all equal or all distinct.

For examples, $\{1425, 1113, 1354\}$ is a trio if the three numbers are members of A because the ones decimals of the three numbers are all distinct, their tens decimals are all distinct, their hundreds decimals are all distinct, and their thousands decimals are all equal. The set $\{1425, 1113, 5436\}$, however, is not a trio, even if A contains those three numbers.

Given a set A as input, write a program that computes and prints out the number of different trios.

Input

Your program is to read from standard input. The input starts with a line consisting of a single integer n ($1 \le n \le 2,000$) that represents the number of members in A. Each of the following n lines consists of a positive integer in decimal form that consists of exactly four digits without 0 in any decimal place. These n numbers are supposed to be all distinct and the members of the input set A.

Output

Your program is to write to standard output. Print exactly one line. The line should consists of a single integer that represents the number of different trios for the input set A.

standard input	standard output
6 1234 1235 1244 1233 7133 8133	1
9 1234 5678 9123 4567 8912 3456 7891 2345 6789	84

Идея

Используем *std::bitset* для эффективного хранения чисел. Для каждого разряда и для каждой цифры хранится битовое множество позиций чисел из множества A. Добавление и удаление чисел из такой структуры очень простое и выполняется фактически за $O(1)$. Пространственная сложность такого хранения $O(\frac{4 \cdot 9 \cdot n}{64}) \approx O(n)$.

Зафиксируем два числа тройки. Выберем только те цифры, которые удовлетворяют условию тройки, добавим в ответ количество индексов чисел. Операция *count* для *std::bitset* выполняется за $O(\frac{n}{64})$, поэтому временная сложность решения $O(\frac{n^3}{64})$.

Исходный код

```
#include <bits/stdc++.h>

using pii = std::pair<int, int>;
using bs = std::bitset<MAX_N>;

const int MAX_N = 2021;

int get_num(int x, int pos) {
    pos = 3 - pos;
    int res = x % 10;
    while (pos--) {
        x = x / 10;
        res = x % 10;
    }
    return res - 1;
}

std::vector< std::vector<bs> > dat(4, std::vector<bs>(9));

void add_num(const std::vector<int> & a, int i) {
    for (int j = 0; j < 4; ++j) {
        dat[j][get_num(a[i], j)][i] = 1;
    }
}

void rm_num(const std::vector<int> & a, int i) {
    for (int j = 0; j < 4; ++j) {
        dat[j][get_num(a[i], j)][i] = 0;
    }
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(0);
    int n;
```

```
36      std::cin >> n;
37      std::vector<int> a(n);
38      for (int i = 0; i < n; ++i) {
39          std::cin >> a[i];
40      }
41      for (int i = 0; i < n; ++i) {
42          add_num(a, i);
43      }
44      // for (int k = 0; k < 4; ++k) {
45      // for (int j = 0; j < 9; ++j) {
46      // std::cout << k << ' ' << j + 1 << ' ' << dat[k][j] << std::endl;
47      // }
48      // }
49      int64_t ans = 0;
50      for (int i = 0; i < n; ++i) {
51          rm_num(a, i);
52          for (int j = i + 1; j < n; ++j) {
53              std::vector<bool> check(4);
54              rm_num(a, j);
55              // std::cout << a[i] << ' ' << a[j] << std::endl;
56              for (int k = 0; k < 4; ++k) {
57                  check[k] = (get_num(a[i], k) == get_num(a[j], k));
58              }
59              bs cur;
60              cur.flip();
61              for (int k = 0; k < 4; ++k) {
62                  if (check[k]) {
63                      cur &= dat[k][get_num(a[i], k)];
64                  } else {
65                      bs pos;
66                      for (int m = 1; m <= 9; ++m) {
67                          int digit = m - 1;
68                          if (digit != get_num(a[i], k) and digit != get_num(a[j], k)) {
69                              pos |= dat[k][digit];
70                          }
71                      }
72                      cur &= pos;
73                  }
74                  // std::cout << cur << std::endl;
75              }
76              // std::cout << "fin " << cur << std::endl;
77              ans += cur.count();
78          }
79          for (int j = i + 1; j < n; ++j) {
80              add_num(a, j);
81          }
82      }
83      std::cout << ans << '\n';
84  }
```

Положение команды

№	Участник Я	A 8/57	B 53/97	C 53/54	D 46/100	E 15/29	F 39/144	G 27/98	H 0/2	I 0/27	J 9/80	K 11/39	L 38/89	Очки	Штраф
43	Moscow AI #2 (Maxim Inyutin, Egor Belousov, Dmitry Artemyev)	—	+ 00:05	+ 00:20	-1 04:35	—	-2 01:31	—	—	—	—	—	+3 02:35	3	241

Grand Prix of Nanjing 12.12.2021

XXII Open Cup named after E.V. Pankratiev
Stage 9: Grand Prix of Nanjing, Division 1, Sunday, December 12, 2021

Problem H. Crystalfly

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Paimon is catching crystalflies on a tree, which are a special kind of butterflies in Teyvat. A tree is a connected graph consisting of n vertices and $(n-1)$ undirected edges.

Pixiv ID: 93964680

There are initially a_i crystalflies on the i-th vertex. When Paimon reaches a vertex, she can catch all the remaining crystalflies on the vertex immediately. However, the crystalflies are timid. When Paimon reaches a vertex, all the crystalflies on the adjacent vertices will be disturbed. For the i-th vertex, if the crystalflies on the vertex are disturbed for the first time at the beginning of the t'-th second, they will disappear at the end of the $(t'+t_i)$-th second.

At the beginning of the 0-th second, Paimon reaches vertex 1 and stays there before the beginning of the 1-st second. Then at the beginning of each following second, she can choose one of the two operations:

- Move to one of the adjacent vertices of her current vertex and stay there before the beginning of the next second (if the crystalflies in the destination will disappear at the end of that second she can still catch them).
- Stay still in her current vertex before the beginning of the next second.

Calculate the maximum number of crystalflies Paimon can catch in $10^{10^{10^{10}}}$ seconds.

Input

There are multiple test cases. The first line of the input contains an integer T indicating the number of test cases. For each test case:

The first line contains an integer n ($1 \le n \le 10^5$) indicating the number of vertices.

The second line contains n integers $a_1, a_2, \cdots, a_n$ ($1 \le a_i \le 10^9$) where a_i is the number of crystalflies on the i-th vertex.

The third line contains n integers $t_1, t_2, \cdots, t_n$ ($1 \le t_i \le 3$) where t_i is the time before the crystalflies on the i-th vertex disappear after disturbed.

For the next $(n-1)$ lines, the i-th line contains two integers u_i and v_i ($1 \le u_i, v_i \le n$) indicating an edge connecting vertices u_i and v_i in the tree.

Grand Prix of Nanjing 12.12.2021

It's guaranteed that the sum of n of all the test cases will not exceed 10^6.

Output

For each test case output one line containing one integer indicating the maximum number of crystalflies Paimon can catch.

Example

standard input	standard output
2 5 1 10 100 1000 10000 1 2 1 1 1 1 2 1 3 2 4 2 5 5 1 10 100 1000 10000 1 3 1 1 1 1 2 1 3 2 4 2 5	10101 10111

Note

For the first sample test case, follow the strategy below.

- During the 0-th second
  - Paimon arrives at vertex 1;
  - Paimon catches 1 crystalfly;
  - Crystalflies in vertices 2 and 3 are disturbed.
- During the 1-st second
  - Paimon arrives at vertex 3;
  - Paimon catches 100 crystalflies.
- During the 2-nd second
  - Paimon arrives at vertex 1;
  - Crystalflies in vertex 2 disappears.
- During the 3-rd second
  - Paimon arrives at vertex 2;
  - Crystalflies in vertices 4 and 5 are disturbed.
- During the 4-th second
  - Paimon arrives at vertex 5;
  - Paimon catches 10000 crystalflies;
  - Crystalflies in vertex 4 disappears.

Идея

Решим сперва чуть более простую задачу, в которой бабочки улетают через 1 или 2 секунды. Возьмем вершину 1 за корень дерева. Пусть мы зашли в какую-то вершину и испугали всех смежных бабочек, тогда в вершины, где бабочки улетают через 1 или 2 секунды после испуга, мы, если хотим собрать бабочек, должны заходить сразу же, так как, если мы зайдем в другую вершину дерева, то нам придется потратить по крайней мере 3 секунды. Тогда пусть dp_i обозначает максимальное кол-во бабочек, которые мы соберем, если начнем собирать в вершине i. Пересчет динамики таков: $dp_i = max_j\{\sum_{c \neq j}(dp_c - a_c) + dp_j\}$ где j и c являются детьми i в дереве, с корнем в 1. Для того, чтобы пересчитать динамику в вершинах с 3 секундами введем еще значение $dpvis_i$ - сколько бабочек мы соберем, начиная с вершины i, если мы не возьмем бабочек в вершине i, а также во всех детях i, но возьмем бабочек во всех детях детей i. Пересчет достаточно прост: $dpvis_i = \sum_c(dp[c] - a[c])$, где c - ребенок i. В конце концов пересчитаем динамику для i дополнительно по вершинам с испугом в 3 секунды: $dp_i = max_{j,k:j \neq k}\{\sum_{c \neq j, c \neq k}(dp_c - a_c) + dpvis_k + a_k + dp_j\}$, j вершина с испугом в 3 секунды. Итого, используя поиск в глубину и предпосчитав суммы через префиксные суммы, можно решить данную задачу за $O(n)$

Исходный код

```
#include <bits/stdc++.h>

using namespace std;

#define flush cout.flush

using ll = long long;
using ull = unsigned long long;
using ld = long double;
using pl = pair<ll, ll>;
const ll INF = 1e9 + 7;
const ll mod = 1e9 + 7;
const ll mod2 = 998244353;
const ld eps = 1e-9;
const ld PI = acos(-1);

vector<vector<ll>> g;
vector<ll> a;
vector<ll> t;
vector<ll> dp;
vector<ll> dp_vis;

void dfs(ll v, ll p) {
    dp[v] = 0;
```

```
25      dp_vis[v] = 0;
26      for (auto to : g[v]) {
27          if (to == p)continue;
28          dfs(to, v);
29      }
30      ll s = 0;
31      for (auto to : g[v]) {
32          if (to == p)continue;
33          dp_vis[v] += dp[to] - a[to];
34          s += dp[to] - a[to];
35      }
36      for (auto to : g[v]) {
37          if (to == p)continue;
38          dp[v] = max(dp[v], s - (dp[to] - a[to]) + dp[to]);
39      }
40      ll m = g[v].size();
41      vector<ll> d(m, 0);
42      vector<ll> suff(m, 0);
43      vector<ll> pref(m, 0);
44      for (ll i = 0; i < m; ++i) {
45          if (g[v][i] == p)continue;
46          ll to = g[v][i];
47          d[i] = dp_vis[to] + a[to] - (dp[to] - a[to]);
48      }
49      for (ll i = 0; i < m; ++i) {
50          pref[i] = max((i - 1 >= 0 ? pref[i - 1] : 0), d[i]);
51      }
52      for (ll i = m - 1; i >= 0; --i) {
53          suff[i] = max((i + 1 < m ? suff[i + 1] : 0), d[i]);
54      }
55      for (ll i = 0; i < m; ++i) {
56          if (g[v][i] == p)continue;
57          ll to = g[v][i];
58          if (t[to] != 3)continue;
59          dp[v] = max(dp[v],
60                      s - (dp[to] - a[to]) + dp[to] + max((i - 1 >= 0 ? pref[i - 1] : 0),
                            (i + 1 < m ? suff[i + 1] : 0)));
61      }
62      dp[v] += a[v];
63  }
64
65  void solve() {
66      g.clear();
67      dp.clear();
68      dp_vis.clear();
69      ll n;
70      cin >> n;
71      dp.resize(n);
72      dp_vis.resize(n);
```

```
73      g.resize(n);
74      a.resize(n);
75      t.resize(n);
76      for (ll &i : a)cin >> i;
77      for (ll &i : t)cin >> i;
78      for (ll i = 0; i < n - 1; ++i) {
79          ll u, v;
80          cin >> u >> v;
81          u--;
82          v--;
83          g[u].push_back(v);
84          g[v].push_back(u);
85      }
86      dfs(0, 0);
87      cout << dp[0] << "\n";
88  }
89
90  int main() {
91      ios::sync_with_stdio(false);
92      cin.tie(NULL);
93      ll t;
94      cin >> t;
95      while (t--)solve();
96      return 0;
97  }
```

Положение команды

№	Участник	A	B	C	D	E	F	G	H	I	J	K	L	M	Очки	Штраф
		62/74	5/14	58/142	37/107	24/92	5/25	21/85	50/105	45/126	45/179	1/3	11/33	59/119		
51	MAI #2 Artemiev, Belousov, Inyutin	+ 00:15	—	+5 01:10	—	—	—	—	+2 02:43	—	-19 04:58	—	—	+1 00:32	4	442

Problem E. Construct The Integer

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

For any positive integer y we define $A(y)$ as the set of all positive integers that are anagrams of y. Formally, we consider the sequence of all digits of y in decimal representation without leading zeroes, then we apply all possible permutations to this sequence. For any permutation we throw away leading zeroes and assemble digits back to an integer. All integers that can be obtained that way belong to $A(y)$. For example, for 2021 set $A(2021)$ consists of integers 122, 212, 221, 1220, 2120, 2210, 1202, 2102, 2201, 1022, 2012 and 2021. Note, that set $A(y)$ always contains y, thus it is never empty.

For any positive integer x we define $S(x)$ as the set of all positive integers z such that greatest common divisor of all integers in $A(z)$ equals x.

You are given the integer n, find the **minimum** integer z in $S(x)$, or determine that $S(x)$ is empty.

Input

The first line of the input contains one integer t ($1 \le t \le 50$) — the number of the test cases.

Each test case data consists of one integer n ($1 \le n \le 10^{18}$).

Output

For each test case, print one integer on a separate line. If the corresponding set $S(n)$ is empty, print `-1`. Otherwise, print minimum element of $S(n)$.

Example

standard input	standard output
2 12 2021	48 -1

Идея

Предпосчитаем gcd для первых 10000 чисел. Видно, что для больших чисел почти всегда gcd его анаграм равен 1. Но есть и исключения, которые мы обработаем отдельно. Наиболее интересны числа, состоящие из чётных цифр и сумма цифр которых делится на три. При перестановке цифр в таком числе делимость на 3 и 2 не изменится. Сложность решения $O(t + k)$, где k — константа предподсчёта $k \approx 10^4 \cdot 4! \cdot 4 \cdot \log 10^4$.

Исходный код

```
#include <bits/stdc++.h>

using pii = std::pair<int, int>;

std::map<int64_t, std::set<int64_t> > mp;

std::set<int64_t> anagrams(int64_t x) {
    std::string s = std::to_string(x);
    std::set<int64_t> res;
    std::sort(s.begin(), s.end());
    do {
        int64_t num = std::stoll(s);
        res.insert(num);
    } while (std::next_permutation(s.begin(), s.end()));
    return res;
}

void process_num(std::string s) {
    if (s.size() > 18) {
        return;
    } else {
        int64_t number = std::stoll(s);
        std::set<int64_t> nums = anagrams(number);
        int64_t gcd_res = *nums.begin();
        for (int64_t elem : nums) {
            gcd_res = std::__gcd(elem, gcd_res);
        }
        mp[gcd_res].insert(number);
    }
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(0);
    for (int i = 0; i <= 9; ++i) {
        std::string num;
        for (int j = 1; j <= 18; ++j) {
            num.push_back('0' + i);
```

```
            int64_t nnum = std::stoll(num);
            mp[nnum].insert(nnum);
        }
    }
    for (int i = 0; i <= 9; ++i) {
        std::string num;
        num.push_back('0' + i);
        num.push_back('0');
        for (int j = 1; j <= 18; ++j) {
            num.push_back('0' + i);
            process_num(num);
        }
    }
    for (int i = 1; i <= 9; ++i) {
        std::string num;
        for (int j = 1; j <= 18; ++j) {
            num.push_back('0' + i);
            std::string magic = num;
            for (int k = 1; k <= 18; ++k) {
                magic.push_back('8');
                process_num(magic);
            }
        }
    }
    for (int64_t i = 1; i <= 1e4; ++i) {
        process_num(std::to_string(i));
    }
    // for (auto elem : mp) {
    // std::cout << "gcd = " << elem.first << ", ";
    // std::cout << *elem.second.begin();
    // // for (int64_t num : elem.second) {
    // // std::cout << num << ' ';
    // // }
    // std::cout << std::endl;
    // }
    int t;
    std::cin >> t;
    while (t--) {
        int64_t n;
        std::cin >> n;
        if (!mp.count(n)) {
            std::cout << "-1\n";
        } else {
            std::cout << *mp[n].begin() << '\n';
        }
    }
}
```

Положение команды

№	Участник	A 278/352	B 0/8	C 11/27	D 134/681	E 87/849	F 239/511	G 175/322	H 154/972	I 12/623	J 51/109	K 3/8	L 37/104	M 71/345	Очки	Штраф
72	Moscow AI: MAI #2 (Dmitry Artemiev,Egor Belousov,Maksim Iniutin)	+ 00:02	—	—	+1 02:25	+3 02:54	+ 00:15	+ 01:41	+1 00:26	—	—	—	—	—	7	641