A. Железный трон

Условие.

В Семи Королевствах расположено N городов, соединенных M железнодорожными путями. Каждый путь соединяет два различных города и позволяет перемещаться между ними в обоих направлениях.

Новый король Вестероса решил создать себе величественный Железный трон, для чего планирует переплавить некоторые железнодорожные пути. Однако, чтобы не нарушить транспортную доступность, король хочет переплавить только "лишние" пути.

Путь считается "лишним", если после его удаления всё ещё существует способ добраться из одного города в другой.

Помогите королю определить, сколько железнодорожных путей он может переплавить для создания своего трона, не нарушая возможности перемещения между городами, которые были связаны ранее.

Формат ввода.

В первой строке содержатся два целых числа N и M $(1 \leq N \leq 2 * 10^5, 0 \leq M \leq 2 * 10^5)$ — количество городов и железнодорожных путей.
Следующие M строк содержат описание путей. Каждая строка содержит два целых числа u и v $(1 \leq u, v \leq n, u \neq v)$, означающих, что существует двунаправленный путь между городами u и v.

Формат вывода.

Выведите одно целое число — максимальное количество путей, которые можно переплавить.

Пример.

Пример 1

Ввод	Вывод
3 1 1 2	0

Пример 2

Ввод	Вывод
3 3 1 2 2 3 3 1	1

Примечание

Считайте, что пути переплавляются последовательно. Так, во втором примере после удаления одного из путей остальные перестают быть "лишними".

Система оценивания

При прохождение всех тестов, ваше решение получит 100 баллов.

Решения

Язык C++

```
#include <chrono>
#include <iostream>
#include <random>
#include <vector>

#define sz(a) ((int)((a).size()))
#define make_unique(x)
\
  sort((x).begin(), (x).end());
\
  (x).erase(unique((x).begin(), (x).end()), (x).end())

using namespace std;
mt19937 rnd(chrono::steady_clock::now().time_since_epoch().count());

typedef long long ll;
typedef long double ld;

void dfs(int v, auto &used, auto &g) {
  used[v] = true;
  for (auto u : g[v])
    if (!used[u])
      dfs(u, used, g);
}

void solve() {
  int n, m;
```

```
  cin >> n >> m;
  vector<vector<int>> g(n);
  for (int i = 0; i < m; ++i) {
    int u, v;
    cin >> u >> v;
    --u, --v;
    g[u].emplace_back(v);
    g[v].emplace_back(u);
  }
  vector<bool> used(n);
  int cnt = 0;
  for (int i = 0; i < n; ++i) {
    if (!used[i]) {
      ++cnt;
      dfs(i, used, g);
    }
  }
  cout << m - (n - cnt) << endl;
}

int32_t main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  int t = 1;
  while (t--) {
    solve();
  }
}
```

Язык Python

```
def dfs(v, used, g):
    used[v] = True
    for u in g[v]:
        if not used[u]:
            dfs(u, used, g)


def solve():
    n, m = map(int, input().split())

    g = [[] for _ in range(n)]
    for _ in range(m):
        u, v = map(int, input().split())
        u -= 1
        v -= 1
```

```
        g[u].append(v)
        g[v].append(u)

    used = [False] * n
    cnt = 0
    for i in range(n):
        if not used[i]:
            cnt += 1
            dfs(i, used, g)

    print(m - (n - cnt))


if __name__ == "__main__":
    solve()
```

B. V

Условие.

При разработке современных технологий для работы с ядерными материалами корпорация «Р» столкнулась с интересным явлением. Оказалось, что радиационная активность некоторых материалов сначала имеет тенденцию к убыванию, достигая определённого минимума, после чего вновь начинает нарастать. Специалисты корпорации решили обозначить такие материалы специальными числовыми метками. Особенностью этих меток является следующая закономерность: цифры в числовой метке при движении от старшего разряда к младшему сначала строго убывают, а затем строго возрастают. Например, числа 3214 или 95479 подходят под эти условия (3→2→1 убывают, затем 1→4 возрастают; 9→5→4 убывают, затем 4→7→9 возрастают). Важно заметить, что число должно обязательно иметь и убывающую, и возрастающую часть, а длина каждой такой части должна быть строго более одной цифры.

Для решения важной задачи по систематизации ядерных материалов корпорация «Р» поручает вам создать универсальную программу, которая подсчитает количество подобных чисел среди всех натуральных чисел, не больше заданного числа N.

Формат ввода.

В первой строке вводится одно натуральное число N ($1 \leq N \leq 10^{16}$) – ограничение для чисел.

Формат вывода.

Выведите одно число — количество натуральных чисел, не превышающих `N` и удовлетворяющих описанному выше условию.

Пример

Ввод	Вывод
120	9

Примечание

Например:

- Число `3214`: 3 → 2 → 1 убывает, далее 1 → 4 возрастает — подходит;
- Число `543456`: 5 → 4 → 3 убывает, далее сразу следует 4 (больше 3), 4 → 5 → 6 возрастает — подходит;
- Число `12345`: только возрастает, не имеет убывающей части — не подходит;
- Число `54321`: только убывает, не имеет возрастающей части — не подходит;
- Число `543215678`: 5 → 4 → 3 → 2 → 1 убывает, далее 1 → 5 → 6 → 7 → 8 возрастает — подходит;
- Число `32123`: 3 → 2 → 1 убывает, затем 1 → 2 → 3 возрастает — подходит;
- Число `212`: 2 → 1 убывание, затем 2 возрастает — подходит.

Для примера подходящие цифры: 101, 102, 103, 104, 105, 106, 107, 108, 109.

Система оценивания

Каждая группа будет тестироваться только при успешном прохождении предыдущей.

Группа	N	Баллы
1	$1 \le N \le 10^3$	10
2	$1 \le N \le 10^6$	15
3	$1 \le N \le 10^9$	20
4	$1 \le N \le 10^{12}$	25
5	$1 \le N \le 10^{16}$	30

Решения

Язык C++

```
#include <cstring>
#include <iostream>
```

```cpp
using namespace std;

typedef long long ll;

ll dp[20][11][3][2][2];

ll countNumbers(const string &num) {
  memset(dp, 0, sizeof(dp));
  int len = num.size();

  for (int prev = 0; prev <= 10; prev++) {
    for (int state = 0; state < 3; state++) {
      for (int has_desc = 0; has_desc <= 1; has_desc++) {
        for (int tight = 0; tight <= 1; tight++) {

          dp[len][prev][state][has_desc][tight] =
              (has_desc && state == 1) ? 1 : 0;
        }
      }
    }
  }

  for (int pos = len - 1; pos >= 0; pos--) {
    for (int prev = 0; prev <= 10; prev++) {
      for (int state = 0; state <= 2; state++) {
        for (int has_desc = 0; has_desc <= 1; has_desc++) {
          for (int tight = 0; tight <= 1; tight++) {
            ll ans = 0;
            int limit = tight ? (num[pos] - '0') : 9;
            for (int digit = 0; digit <= limit; digit++) {
              int new_tight = (tight && digit == limit);
              if (prev == 10 && digit == 0) {
                // Skip leading zeros
                ans += dp[pos + 1][prev][state][has_desc][new_tight];
              } else if (prev == 10 && digit != 0) {
                // First non-zero digit
                ans += dp[pos + 1][digit][2][0][new_tight];
              } else {
                if (state == 2) {
                  if (digit < prev)
                    ans += dp[pos + 1][digit][0][1][new_tight];
                  // equal or greater invalid at first nonzero position
                } else if (state == 0) {
                  if (digit < prev)
                    ans += dp[pos + 1][digit][0][has_desc][new_tight];
                  else if (digit > prev)
                    ans += dp[pos + 1][digit][1][has_desc][new_tight];
                } else if (state == 1) {
                  if (digit > prev)
                    ans += dp[pos + 1][digit][1][has_desc][new_tight];
```

```
                }
              }
            }
            dp[pos][prev][state][has_desc][tight] = ans;
          }
        }
      }
    }
  }

  return dp[0][10][2][0][1];
}

int main() {
  string N;
  cin >> N;
  ll result = countNumbers(N);
  cout << result << "\n";
  return 0;
}
```

Язык Python

```
def count_numbers(num: str) -> int:
    dp = [[[[[0 for _ in range(2)] for _ in range(2)]
             for _ in range(3)] for _ in range(11)] for _ in range(20)]

    length = len(num)

    for prev in range(11):
        for state in range(3):
            for has_desc in range(2):
                for tight in range(2):
                    dp[length][prev][state][has_desc][tight] = 1 if
(has_desc and state == 1) else 0

    for pos in range(length - 1, -1, -1):
        for prev in range(11):
            for state in range(3):
                for has_desc in range(2):
                    for tight in range(2):
                        ans = 0
                        limit = int(num[pos]) if tight else 9
                        for digit in range(limit + 1):
                            new_tight = tight and (digit == limit)
                            if prev == 10 and digit == 0:
                                # Skip leading zeros
                                ans += dp[pos + 1][prev][state][has_desc]
```

```
[new_tight]
                            elif prev == 10 and digit != 0:
                                # First non-zero digit
                                ans += dp[pos + 1][digit][2][0][new_tight]
                            else:
                                if state == 2:
                                    if digit < prev:
                                        ans += dp[pos + 1][digit][0][1]
[new_tight]
                                elif state == 0:
                                    if digit < prev:
                                        ans += dp[pos + 1][digit][0]
[has_desc][new_tight]
                                    elif digit > prev:
                                        ans += dp[pos + 1][digit][1]
[has_desc][new_tight]
                                elif state == 1:
                                    if digit > prev:
                                        ans += dp[pos + 1][digit][1]
[has_desc][new_tight]
                        dp[pos][prev][state][has_desc][tight] = ans

    return dp[0][10][2][0][1]

if __name__ == "__main__":
    N = input()
    result = count_numbers(N)
    print(result)
```

C. Задача от Большого Брата

Условие.

В мире, где реальность и истина стали манипулируемыми инструментами, правит могущественная корпорация "Океания Инкорпорейтед". Её глава — таинственный и всевидящий Большой Брат, чье лицо всегда наблюдает за каждым сотрудником. Большой Брат — это не просто лидер, это символ абсолютной власти и контроля над мыслями и действиями каждого.

Океания Инкорпорейтед — это колоссальная структура, пронизывающая все аспекты жизни. Каждый её сотрудник связан сложной сетью иерархий, где у всех, кроме Большого Брата, есть непосредственный начальник. Все подчиняются Великой Управляющей Системе, которая гарантирует порядок и контроль. Таким образом, все сотрудники, прямо или косвенно, служат Великому Брату.

Внутри корпорации действует строгая иерархия, которая обеспечивает полный контроль сверху вниз. Если сотрудник А является подчиненным сотрудника Б, то все подчиненные сотрудника А также автоматически считаются подчиненными сотрудника Б. Таким образом, каждая линия подчиненности охватывает весь спектр сотрудников, спускаясь по всей иерархической цепочке. Все сотрудники, за исключением несравненного главы корпорации — Большого Брата, находятся в структуре подчиненных ему, таким образом обеспечивая его абсолютное влияние и контроль над всей организацией.

В корпорации также действует "Министерство Правды", которое ответственное за поддержание социального равновесия через инструмент под названием метрика несправедливости. Для ее вычисления рассматривается все сотрудники, подчиненные X. Выявляются двое сотрудников с наибольшими зарплатами и двое с наименьшими. Из суммы двух наибольших зарплат вычитается сумма двух наименьших. Полученное значение используется отделом как метрика несправедливости, при этом сам X во внимание не принимается. Такая система помогает контролировать стабильность внутри компании и предотвращать подрыв основ власти.

В компании также функционирует "Министерство Изобилия", ответственное за управление внутренними стимулами и поощрениями сотрудников. По указу этого министерства, зарплата определенного сотрудника X и всех его подчиненных может быть изменена на величину d. Эта корректировка выполняется с целью поддержания состояния рабочей "динамики" — обеспечения приверженности и гибкости сотрудников в соответствии с целями и ценностями корпорации.

Задача системного контроля в этом угнетенном мире — безоговорочно исполнять указания этих министерств. Они вычисляют несправедливость и регулируют оклады, препятствуя любым отклонениям от нормы. Так корпорация сохраняет пропасть между подчиненными и руководством, чтобы никто не смог усомниться в безграничной власти Большого Брата.

Формат ввода.

В первой строке вводится единственное число N $(2 \leq N \leq 250000)$ - число сотрудников в компании.
Во второй строке вводится $N-1$ число a_i $(1 \leq a_i \leq N)$ - непосредственный руководитель сотрудника номер $i+1$.
Руководитель компании имеет номер 1. Гарантируется, что каждый из сотрудников, кроме руководителя, является подчиненным руководителя.
В третьей строке вводится N чисел b_i $(b_1 = 10^{30}, 10^4 \leq b_2, \ldots, b_n \leq 10^9)$ - зарплаты сотрудников
В четвертой строке вводится число Q $(1 \leq Q \leq 4 * 10^5)$ - количество запросов, на которое необходимо ответить.
В следующих Q строках вводятся запросы в одном из двух форматов:

- 1 X ($1 \le X \le N$), запрос вычисления неравенства среди подчиненных X, вывести 0, если у X нет подчиненных или он один;
- 2 X d ($1 \le X \le N$, $-10^5 \le d \le 10^5$) запрос изменения зарплаты X и всех его подчиненных на заданную величину.
  Гарантируется, что будет хотя бы один запрос 1 типа.

Формат вывода.

На каждый запрос первого типа необходимо вывести строку с единственным числом - ответом на запрос.

Пример

Ввод	Вывод
6 1 2 1 4 4 10000000000000000000000000000000 10005 10004 10003 10002 10001 4 1 1 1 3 2 4 10 1 1	6 0 16

Примечание

Пояснения к тесту из условия:

- Первый запрос (1, 1). В подчинении у Большого Брата находятся все сотрудники: 2, 3, 4, 5, 6. Их зарплаты равны 10005, 10004, 10003, 10002, 10001. Две максимальных зарплаты 10005 и 10004. Две минимальных зарплаты: 10002 и 10001. Ответом на запрос будет 10005 + 10004 - 10002 - 10001 = 6.
- Второй запрос (1, 3). В подчинении у сотрудника никого нет, ответ на запрос 0.
- Третий запрос (2, 4, 10). В подчинении сотрудника 4 находятся сотрудники 5 и 6. После изменения зарплат сотрудников 4, 5, 6 их зарплаты станут 10013, 10012, 10011.
- Четвертый запрос (1, 1). В подчинении у Большого Брата находятся все сотрудники: 2, 3, 4, 5, 6. Их зарплаты равны 10005, 10004, 10013, 10012, 10011. Две максимальных зарплаты 10013 и 10012. Две минимальных зарплаты: 10005 и 10004. Ответом на запрос будет 10013 + 10012 - 10005 - 10004 = 16.

Система оценивания

Группа 1. Гарантируется, что $N \le 5000$, $Q \le 5000$. За эту группу вы получите 19 баллов.
Группа 2. Полные ограничения. За эту группу вы получите 81 балл. Будет

тестироваться только при прохождении группы 1.

Решения

Язык C++

```cpp
#include <iostream>
#include <vector>

using namespace std;
using ll = long long;

const int MAXN = 1 << 18;
const ll INF = 1e18;

struct Node {
  ll first_max = -INF;
  ll second_max = -INF;
  ll first_min = INF;
  ll second_min = INF;
  ll push = 0;
  Node get_clear() const {
    return Node{
        .first_max = first_max + push,
        .second_max = second_max + push,
        .first_min = first_min + push,
        .second_min = second_min + push,
    };
  }
};

Node segtree[MAXN << 1];

Node recalc(Node lhs_dirty, Node rhs_dirty) {
  Node lhs = lhs_dirty.get_clear();
  Node rhs = rhs_dirty.get_clear();
  Node res;
  res.first_max = max(lhs.first_max, rhs.first_max);
  res.first_min = min(lhs.first_min, rhs.first_min);
  if (lhs.first_max > rhs.first_max) {
    res.second_max = max(lhs.second_max, rhs.first_max);
  } else {
    res.second_max = max(lhs.first_max, rhs.second_max);
  }
  if (lhs.first_min < rhs.first_min) {
    res.second_min = min(lhs.second_min, rhs.first_min);
  } else {
    res.second_min = min(lhs.first_min, rhs.second_min);
  }
```

```
  return res;
}

void make_push(int nd) {
  segtree[nd + nd].push += segtree[nd].push;
  segtree[nd + nd + 1].push += segtree[nd].push;
  segtree[nd].push = 0;
}

void collect_nodes(int l_req, int r_req, vector<int> &nodes, int v = 1,
                   int lb = 0, int rb = MAXN) {
  if (l_req <= lb && rb <= r_req) {
    nodes.push_back(v);
    return;
  }
  if (r_req <= lb || rb <= l_req) {
    return;
  }
  make_push(v);
  collect_nodes(l_req, r_req, nodes, v + v, lb, (lb + rb) >> 1);
  collect_nodes(l_req, r_req, nodes, v + v + 1, (lb + rb) >> 1, rb);
  segtree[v] = recalc(segtree[v + v], segtree[v + v + 1]);
}

void add_segment(int l_req, int r_req, ll d, int v = 1, int lb = 0,
                 int rb = MAXN) {
  if (l_req <= lb && rb <= r_req) {
    segtree[v].push += d;
    return;
  }
  if (r_req <= lb || rb <= l_req) {
    return;
  }
  make_push(v);
  add_segment(l_req, r_req, d, v + v, lb, (lb + rb) >> 1);
  add_segment(l_req, r_req, d, v + v + 1, (lb + rb) >> 1, rb);
  segtree[v] = recalc(segtree[v + v], segtree[v + v + 1]);
}

vector<int> g[MAXN];
int tin[MAXN], tout[MAXN], timer = 0;

vector<pair<int, int>> my_stack;

void dfs(int v = 0) {
  my_stack.push_back({0, 0});
  tin[v] = timer++;
  while (my_stack.size()) {
    auto [v, cnt] = my_stack.back();
    my_stack.pop_back();
```

```
    if (cnt == g[v].size()) {
      tout[v] = timer;
    } else {
      my_stack.push_back({v, cnt + 1});
      my_stack.push_back({g[v][cnt], 0});
      tin[g[v][cnt]] = timer++;
    }
  }
}

ll first_query(int x) {
  vector<int> nodes;
  collect_nodes(tin[x] + 1, tout[x], nodes);
  if (nodes.empty()) {
    return 0;
  }
  Node res;
  for (auto id : nodes) {
    res = recalc(res, segtree[id]);
  }
  if (res.second_max < -(1e17)) {
    return 0;
  }
  return res.first_max + res.second_max - res.first_min - res.second_min;
}

void second_query(int x, ll d) { add_segment(tin[x], tout[x], d); }

void solve() {
  int n;
  { // reading graph
    cin >> n;
    for (int i = 1; i < n; ++i) {
      int par;
      cin >> par;
      g[par - 1].push_back(i);
    }
  }
  dfs();
  { // reading salaries into botton of segtree
    string useless;
    cin >> useless;
    for (int i = 1; i < n; ++i) {
      int salary;
      cin >> salary;
      int node_id = MAXN + tin[i];
      segtree[node_id].first_max = segtree[node_id].first_min = salary;
    }
  }
  { // building segtree
```

```
    for (int nd = MAXN - 1; nd > 0; --nd) {
      segtree[nd] = recalc(segtree[nd + nd], segtree[nd + nd + 1]);
    }
  }
  { // queries
    int q;
    cin >> q;
    while (q--) {
      int t;
      cin >> t;
      if (t == 1) {
        int x;
        cin >> x;
        cout << first_query(x - 1) << endl;
      } else {
        int x;
        ll d;
        cin >> x >> d;
        second_query(x - 1, d);
      }
    }
  }
}

signed main() {
  cin.tie(nullptr);
  cout.tie(nullptr);
  ios_base::sync_with_stdio(false);
  int t = 1;
  while (t--) {
    solve();
  }
  return 0;
}
```

Язык Python

```
class Node:
    def __init__(self):
        self.first_max = -(10**18)
        self.second_max = -(10**18)
        self.first_min = (10**18)
        self.second_min = (10**18)
        self.push = 0

    def get_clear(self):
        result = Node()
        result.first_max=self.first_max + self.push
```

```
        result.second_max=self.second_max + self.push
        result.first_min=self.first_min + self.push
        result.second_min=self.second_min + self.push
        return result

MAXN = 1 << 18

def recalc(lhs_dirty, rhs_dirty):
    lhs = lhs_dirty.get_clear()
    rhs = rhs_dirty.get_clear()
    res = Node()
    res.first_max = max(lhs.first_max, rhs.first_max)
    res.first_min = min(lhs.first_min, rhs.first_min)
    if lhs.first_max > rhs.first_max:
        res.second_max = max(lhs.second_max, rhs.first_max)
    else:
        res.second_max = max(lhs.first_max, rhs.second_max)
    if lhs.first_min < rhs.first_min:
        res.second_min = min(lhs.second_min, rhs.first_min)
    else:
        res.second_min = min(lhs.first_min, rhs.second_min)
    return res

def make_push(nd):
    segtree[nd*2].push += segtree[nd].push
    segtree[nd*2 + 1].push += segtree[nd].push
    segtree[nd].push = 0

def collect_nodes(l_req, r_req, nodes, v=1, lb=0, rb=MAXN):
    if l_req <= lb and rb <= r_req:
        nodes.append(v)
        return
    if r_req <= lb or rb <= l_req:
        return
    make_push(v)
    mid = (lb + rb) // 2
    collect_nodes(l_req, r_req, nodes, v*2, lb, mid)
    collect_nodes(l_req, r_req, nodes, v*2 + 1, mid, rb)
    segtree[v] = recalc(segtree[v*2], segtree[v*2 + 1])

def add_segment(l_req, r_req, d, v=1, lb=0, rb=MAXN):
    if l_req <= lb and rb <= r_req:
        segtree[v].push += d
        return
    if r_req <= lb or rb <= l_req:
        return
    make_push(v)
    mid = (lb + rb) // 2
    add_segment(l_req, r_req, d, v*2, lb, mid)
    add_segment(l_req, r_req, d, v*2 + 1, mid, rb)
```

```
    segtree[v] = recalc(segtree[v*2], segtree[v*2 + 1])

def dfs():
    global timer
    my_stack = []
    my_stack.append((0, 0))
    tin[0] = timer
    timer += 1
    while my_stack:
        v, cnt = my_stack.pop()
        if cnt == len(g[v]):
            tout[v] = timer
        else:
            my_stack.append((v, cnt + 1))
            child = g[v][cnt]
            my_stack.append((child, 0))
            tin[child] = timer
            timer += 1

def first_query(x):
    nodes = []
    collect_nodes(tin[x] + 1, tout[x], nodes)
    if not nodes:
        return 0
    res = Node()
    for id in nodes:
        res = recalc(res, segtree[id])
    if res.second_max < -(10**17):
        return 0
    return res.first_max + res.second_max - res.first_min - res.second_min

def second_query(x, d):
    add_segment(tin[x], tout[x], d)

def solve():
    n = int(input().strip())
    global segtree
    global g, tin, tout, timer

    g = [[] for _ in range(MAXN)]
    tin = [0] * MAXN
    tout = [0] * MAXN
    timer = 0

    par = list(map(int, input().strip().split()))
    for i in range(1, n):
        g[par[i - 1] - 1].append(i)

    dfs()
```

```
    salary = list(map(int, input().strip().split()))
    for i in range(1, n):
        node_id = MAXN + tin[i]
        segtree[node_id].first_max = salary[i]
        segtree[node_id].first_min = salary[i]

    for nd in range(MAXN - 1, 0, -1):
        segtree[nd] = recalc(segtree[2 * nd], segtree[2 * nd + 1])

    q = int(input().strip())
    for _ in range(q):
        t, *params = map(int, input().strip().split())
        if t == 1:
            x = params[0]
            print(first_query(x - 1))
        else:
            x, d = params
            second_query(x - 1, d)

if __name__ == "__main__":
    import sys
    segtree = [Node() for _ in range(MAXN << 1)]
    solve()
```

D. Ошибка стажёра

Условие.

Корпорация **P** — флагман атомных технологий, постоянно занимающийся исследованиями в сфере повышения эффективности ядерных реакций. В исследовательских лабораториях корпорации недавно была разработана революционная технология управления топливом — метод так называемого **двойного слияния потоков**.

Перед процессом запуска нового экспериментального реактора ученые формируют два исходных массива атомных элементов (назовем их массивы **A** и **B**). Согласно новому алгоритму Merge Sort, эти два массива соединяются в один общий поток элементов — массив **C**. Слияние происходит по простому правилу:

- Исследователь берет по одному элементу из каждого потока и выбирает из двух тот, у которого показатель радиоактивности меньше.
- Затем выбранный элемент добавляется в общий поток, а из исходного массива берется следующий элемент.

- Если один из массивов заканчивается, остальные элементы из другого массива добавляются в общий поток в исходном порядке без всяких сравнений.

Однако, как это часто бывает в инновационных исследованиях, начинающий лаборант Иван случайно перемешал элементы в итоговом массиве **C**. Теперь ученым необходимо срочно разобраться, можно ли восстановить исходные два массива элементов — **A** и **B**, чтобы произвести успешный запуск реактора. В противном случае запуск придется отложить, а демонстрация новых технологий станет невозможной.

Помогите экспертам "Р" оперативно понять, можно ли "распутать" перепутанный лаборантом массив **C** так, чтобы его можно было получить, применив описанный выше процесс слияния к двум исходным массивам элементов (не обязательно упорядоченным!).

Формат ввода.

В первой строке вводится целое число N — количество элементов в каждом из исходных массивов ($1 \leq N \leq 1000$).
Во второй строке вводится $2 \times N$ целых чисел C_i — элементы массива C, являющегося перестановкой чисел от 1 до $2 \times N$.

Формат вывода.

Если восстановить исходные массивы невозможно, выведите единственное число −1.
В противном случае, выведите сначала N элементов первого массива, а затем на следующей строке N элементов второго массива. Если решений несколько, выводите любое из них.

Пример.

Пример 1

Ввод	Вывод
10 9 3 14 6 5 16 11 1 18 13 4 8 12 15 2 20 10 19 17 7	16 11 1 18 13 4 8 12 15 2 9 3 14 6 5 20 10 19 17 7

Пример 2

Ввод	Вывод
7 9 8 2 10 14 5 11 12 1 7 4 3 6 13	-1

Примечание

Псевдокод алгоритма `merge`.

```
function merge(left,right)
    var _list_ result
    while length(left) > 0 and length(right) > 0
        if first(left) ≤ first(right)
            append first(left) to result
            left = rest(left)
        else
            append first(right) to result
            right = rest(right)
        end if
    while length(left) > 0
        append first(left) to result
        left = rest(left)
    while length(right) > 0
        append first(right) to result
        right = rest(right)
    return result
```

Допустим, исходные массивы имеют вид:

`A = [3, 1, 6] B = [4, 5, 2]`

Тогда итоговый массив C будет формироваться следующим образом:

- сравниваем 3 и 4 → выбираем 3, массив C = [3]
- сравниваем 1 и 4 → выбираем 1, массив C = [3, 1]
- сравниваем 6 и 4 → выбираем 4, массив C = [3, 1, 4]
- сравниваем 6 и 5 → выбираем 5, массив C = [3, 1, 4, 5]
- сравниваем 6 и 2 → выбираем 2, массив C = [3, 1, 4, 5, 2]
- массив B закончился, добавляем оставшиеся элементы из A → добавляем 6, массив C = [3, 1, 4, 5, 2, 6]

Система оценивания

Каждая группа будет тестироваться только при успешном тестировании предыдущей.

Группа	N	Баллы
1	$1 \leq N \leq 10$	10
2	$1 \leq N \leq 100$	20
3	$1 \leq N \leq 500$	30

Группа	N	Баллы
4	$1 \leq N \leq 1000$	40

Решения

Язык C++

```cpp
#include <algorithm>
#include <cmath>
#include <cstdio>
#include <cstring>
#include <vector>
using namespace std;

const int MAX_N = 1000;
const int MAX_2_SQRT_2N = 92; // Enough to cover constraints

typedef pair<int, int> PII;

int n, m, printed;
int a[2 * MAX_N + 1];
signed char mem[MAX_2_SQRT_2N + 1][2 * MAX_N + 1];
bool taken[2 * MAX_N + 1];
vector<int> vec;
vector<PII> item;
vector<vector<int>> group;
vector<vector<int>> group_len[2 * MAX_N + 1];

int main() {
  scanf("%d", &n);
  for (int i = 0; i < 2 * n; i++)
    scanf("%d", &a[i]);

  vec.push_back(a[0]);
  for (int i = 1; i < 2 * n; i++) {
    if (a[i] > vec[0]) {
      group.push_back(vec);
      vec.clear();
    }
    vec.push_back(a[i]);
  }
  group.push_back(vec);

  for (int i = 0; i < group.size(); i++) {
    vec.clear();
    vec.push_back(i);
    group_len[group[i].size()].push_back(vec);
```

```
  }

  for (int i = 1; i <= 2 * n; i++) {
    int k = group_len[i].size();
    while (k >= 3) {
      vec.clear();
      for (int x : group_len[i][k - 1]) {
        vec.push_back(x);
      }
      group_len[i].pop_back();
      k--;

      for (int x : group_len[i][k - 1]) {
        vec.push_back(x);
      }

      group_len[i].pop_back();
      k--;
      group_len[2 * i].push_back(vec);
    }
  }

  for (int i = 1; i <= 2 * n; i++) {
    for (int j = 0; j < group_len[i].size(); j++) {
      item.emplace_back(i, j);
    }
  }

  m = item.size();
  memset(mem, 0, sizeof mem);
  mem[m][0] = 1; // base case: 0 elements remaining at position m is true

  for (int pos = m - 1; pos >= 0; pos--) {
    for (int rem = 0; rem <= n; rem++) {
      mem[pos][rem] = mem[pos + 1][rem];
      if (item[pos].first <= rem) {
        mem[pos][rem] = max(mem[pos][rem], mem[pos + 1][rem -
item[pos].first]);
      }
    }
  }

  if (mem[0][n]) {
    int pos = 0, rem = n;
    while (pos < m && rem > 0) {
      if (mem[pos + 1][rem]) {
        pos++;
      } else {
        for (int idx : group_len[item[pos].first][item[pos].second]) {
          taken[idx] = true;
```

```
        }

        rem -= item[pos].first;
        pos++;
      }
    }

    for (int i = 0; i < group.size(); i++) {
      if (taken[i]) {
        for (int v : group[i]) {
          printf("%d%c", v, (++printed % n == 0 ? '\n' : ' '));
        }
      }
    }

    for (int i = 0; i < group.size(); i++) {
      if (!taken[i]) {
        for (int v : group[i]) {
          printf("%d%c", v, (++printed % n == 0 ? '\n' : ' '));
        }
      }
    }
  } else {
    printf("-1\n");
  }
}
```

Язык Python

```
import sys
from collections import defaultdict

MAX_N = 1000
MAX_2_SQRT_2N = 92  # Enough to cover constraints

def main():
    input = sys.stdin.read
    data = input().split()

    n = int(data[0])
    a = list(map(int, data[1:2*n+1]))

    vec = [a[0]]
    group = []

    for i in range(1, 2 * n):
        if a[i] > vec[0]:
            group.append(vec)
```

```
            vec = []
        vec.append(a[i])
    group.append(vec)

    group_len = defaultdict(list)

    for i in range(len(group)):
        group_len[len(group[i])].append([i])

    for i in range(1, 2 * n + 1):
        k = len(group_len[i])
        while k >= 3:
            vec = []
            for x in group_len[i][k - 1]:
                vec.append(x)
            group_len[i].pop()
            k -= 1

            for x in group_len[i][k - 1]:
                vec.append(x)
            group_len[i].pop()
            k -= 1
            group_len[2 * i].append(vec)

    item = []

    for i in range(1, 2 * n + 1):
        for j in range(len(group_len[i])):
            item.append((i, j))

    m = len(item)
    mem = [[0] * (2 * n + 1) for _ in range(MAX_2_SQRT_2N + 1)]
    mem[m][0] = 1  # base case: 0 elements remaining at position m is true

    for pos in range(m - 1, -1, -1):
        for rem in range(n + 1):
            mem[pos][rem] = mem[pos + 1][rem]
            if item[pos][0] <= rem:
                mem[pos][rem] = max(mem[pos][rem], mem[pos + 1][rem -
item[pos][0]])

    printed = 0
    taken = [False] * (2 * n + 1)

    if mem[0][n]:
        pos, rem = 0, n
        while pos < m and rem > 0:
            if mem[pos + 1][rem]:
                pos += 1
            else:
```

```
                for idx in group_len[item[pos][0]][item[pos][1]]:
                    taken[idx] = True

                rem -= item[pos][0]
                pos += 1

        for i in range(len(group)):
            if taken[i]:
                for v in group[i]:
                    end_char = '\n' if (printed + 1) % n == 0 else ' '
                    print(v, end=end_char)
                    printed += 1

        for i in range(len(group)):
            if not taken[i]:
                for v in group[i]:
                    end_char = '\n' if (printed + 1) % n == 0 else ' '
                    print(v, end=end_char)
                    printed += 1
    else:
        print("-1")

if __name__ == "__main__":
    main()
```

E. Максимум силы

Условие.

Корпорация "P", являясь мировым лидером в области атомной энергетики, активно занимается мониторингом и регулированием мощности своих ядерных реакторов. В каждом из реакторов находится специальный ряд из N контролирующих стержней, описываемых мощностью p_i ($1 \leq i \leq N$), каждый из которых отвечает за поддержание оптимального уровня мощности системы.

Изначально состояние каждого контролирующего стержня выставлено на нулевой уровень ($\forall i \; a_i == 0$). Для плавного и безопасного управления реактором необходимо периодически проводить операции по корректировке состояния этих стержней.

Известно, что инженеры корпорации "P" запланировали M последовательных операций, пронумерованных от 1 до M. Во время операции с номером i выбирается последовательность стержней от $Left_i$ до $Right_i$ включительно, и у каждого из них уровень мощности изменяется на величину X_i (эта величина может быть как положительной, повышающей мощность, так и отрицательной, снижающей её).

После планирования операций руководство поставило перед инженерами Q задач, которые требуется быстро решить для поддержки максимального уровня безопасности реактора.
Каждая такая задача характеризуется набором чисел $(Idx, Start, Finish)$. Она означает следующее: инженерам необходимо выбрать непрерывную последовательность операций с индексами от некоторого l до некоторого r (где $Start \leq l \leq r \leq Finish$), выполнить все эти операции и узнать, каким максимально возможным может оказаться итоговый уровень мощности у стержня с номером idx (p_{idx}) после применения выбранного набора операций.

Формат ввода.

В первой строке через проблем вводится два целых числа N, M ($1 \leq N, M \leq 10^5$).
В следующих M-строках вводится три целых числа $-$ $Left_i$, $Right_i$, X_i ($1 \leq Left_i \leq Right_i \leq N$; $|X_i| \leq 10^5$).
В следующих Q строках ($1 \leq Q \leq 10^5$) вводится три целых числа $-$ Idx, $Start$, $Finish$ ($1 \leq Idx \leq N$, $1 \leq Start \leq Finish \leq M$).

Формат вывода.

В Q строках выведите по одному числу $-$ ответ на вопрос.

Пример

Ввод	Вывод
7 1 1 7 13336 3 1 1 1 2 1 1 1 1 1	13336 13336 13336

Примечание

N = 4, M = 3
Операции:
(1, 3, +10) — прибавить +10 мощности стержням №1,2,3
(2, 4, -5) — убавить -5 мощности стержням №2,3,4
(3, 4, +7) — прибавить +7 мощности стержням №3,4

Q = 2 запроса:
(3, 1, 3) — стержень №3, операции с 1 по 3
(4, 2, 3) — стержень №4, операции со 2 по 3

Запрос 1: `(3, 1, 3)`
Берем стержень номер 3 и операции 1–3:

- Операция 1: стержень №3 получает +10
- Операция 2: стержень №3 получает -5
- Операция 3: стержень №3 получает +7

Теперь нам нужно выбрать непрерывный отрезок операций (от 1 до 3), чтобы максимизировать мощность стержня №3:

- Если возьмем только операцию (1): 10
- Если возьмем операции (1, 2): 10 + (-5) = 5
- Если возьмем операции (1, 2, 3): 10 - 5 + 7 = 12 **(максимум)**
- Если возьмем (2): -5
- Если возьмем (2, 3): -5 + 7 = 2
- Если возьмем только (3): 7

Максимум получается, если брать операции с 1 по 3: **ответ = 12**

Запрос 2: `(4, 2, 3)`
Теперь рассмотрим только стержень №4 и операции №2-3:

- Операция 2 для стержня №4: -5
- Операция 3 для стержня №4: +7
  Возможные варианты подряд:
- (2) : -5
- (2,3): -5 + 7 = 2
- (3): 7 **(максимум)**
  Максимальная мощность будет, если возьмем только операцию 3: **ответ = 7**

Система оценивания

Каждая группа будет тестироваться только при успешном тестировании предыдущей.

Группа	Баллы	Ограничения
1	10	N,M,Q≤10
2	15	N,M,Q≤100
3	10	$Left_i = Right_i$, отдельно точки
4	20	$N \leq 100$, $M, Q \leq 10^5$
5	15	Маленький разброс $0 \leq X_i \leq 100$, крупные отрезки
6	10	Запросы на единичное событие Start=Finish
7	10	Широкий диапазон запросов: $Start_i = 1, Finish_i = M$

Группа	Баллы	Ограничения
8	10	Без дополнительных ограничений

Решения

Язык C++

```cpp
#include <iostream>
#include <tuple>
#include <vector>

using namespace std;
#define fast_io
\
  ios::sync_with_stdio(false);
\
  cin.tie(nullptr);
\
  cout.tie(nullptr);
#define endl '\n'

typedef long long ll;

const int MAXN = 100005;
const ll MIN_INF = -1e15;

struct Node {
  ll mx_left, mx_right, mx_sum, total_sum;
};

vector<pair<int, int>> updates[MAXN];
vector<tuple<int, int, int>> queries[MAXN];
Node segtree[4 * MAXN];
ll answer[MAXN];

Node merge(const Node &left, const Node &right) {
  Node res;
  res.total_sum = left.total_sum + right.total_sum;
  res.mx_left = max(left.mx_left, left.total_sum + right.mx_left);
  res.mx_right = max(right.mx_right, right.total_sum + left.mx_right);
  res.mx_sum =
      max(max(left.mx_sum, right.mx_sum), left.mx_right + right.mx_left);
  return res;
}

void build(int node, int l, int r) {
  segtree[node] = {0, 0, 0, 0};
```

```
  if (l == r)
    return;
  int mid = (l + r) / 2;
  build(2 * node, l, mid);
  build(2 * node + 1, mid + 1, r);
}

void point_update(int node, int l, int r, int pos, ll val) {
  if (l == r) {
    segtree[node] = {val, val, val, val};
    return;
  }
  int mid = (l + r) / 2;
  if (pos <= mid)
    point_update(2 * node, l, mid, pos, val);
  else
    point_update(2 * node + 1, mid + 1, r, pos, val);
  segtree[node] = merge(segtree[2 * node], segtree[2 * node + 1]);
}

Node range_query(int node, int l, int r, int ql, int qr) {
  if (qr < l || r < ql)
    return {MIN_INF, MIN_INF, MIN_INF, 0};
  if (ql <= l && r <= qr)
    return segtree[node];
  int mid = (l + r) / 2;
  Node left = range_query(2 * node, l, mid, ql, qr);
  Node right = range_query(2 * node + 1, mid + 1, r, ql, qr);
  return merge(left, right);
}

int main() {
  fast_io;
  int n, m, q;
  cin >> n >> m;

  for (int i = 1; i <= m; i++) {
    int l, r, x;
    cin >> l >> r >> x;
    updates[l].emplace_back(i, x);
    updates[r + 1].emplace_back(i, 0);
  }

  cin >> q;
  for (int i = 1; i <= q; i++) {
    int k, s, t;
    cin >> k >> s >> t;
    queries[k].emplace_back(s, t, i);
  }
```

```
  build(1, 1, m);

  for (int i = 1; i <= n; i++) {
    for (auto &[id, val] : updates[i])
      point_update(1, 1, m, id, val);
    for (auto &[l, r, id] : queries[i]) {
      Node res = range_query(1, 1, m, l, r);
      answer[id] = res.mx_sum;
    }
  }

  for (int i = 1; i <= q; i++)
    cout << answer[i] << endl;
  return 0;
}
```

Язык Python

```
import sys
from typing import List, Tuple

MAXN = 100005
MIN_INF = -1e15

class Node:
    def __init__(self, mx_left=0, mx_right=0, mx_sum=0, total_sum=0):
        self.mx_left = mx_left
        self.mx_right = mx_right
        self.mx_sum = mx_sum
        self.total_sum = total_sum

updates = [[] for _ in range(MAXN)]
queries = [[] for _ in range(MAXN)]
segtree = [Node() for _ in range(4 * MAXN)]
answer = [0] * MAXN

def merge(left: Node, right: Node) -> Node:
    res = Node()
    res.total_sum = left.total_sum + right.total_sum
    res.mx_left = max(left.mx_left, left.total_sum + right.mx_left)
    res.mx_right = max(right.mx_right, right.total_sum + left.mx_right)
    res.mx_sum = max(max(left.mx_sum, right.mx_sum), left.mx_right +
right.mx_left)
    return res

def build(node: int, l: int, r: int) -> None:
    segtree[node] = Node(0, 0, 0, 0)
    if l == r:
```

```
        return
    mid = (l + r) // 2
    build(2 * node, l, mid)
    build(2 * node + 1, mid + 1, r)

def point_update(node: int, l: int, r: int, pos: int, val: int) -> None:
    if l == r:
        segtree[node] = Node(val, val, val, val)
        return
    mid = (l + r) // 2
    if pos <= mid:
        point_update(2 * node, l, mid, pos, val)
    else:
        point_update(2 * node + 1, mid + 1, r, pos, val)
    segtree[node] = merge(segtree[2 * node], segtree[2 * node + 1])

def range_query(node: int, l: int, r: int, ql: int, qr: int) -> Node:
    if qr < l or r < ql:
        return Node(MIN_INF, MIN_INF, MIN_INF, 0)
    if ql <= l and r <= qr:
        return segtree[node]
    mid = (l + r) // 2
    left = range_query(2 * node, l, mid, ql, qr)
    right = range_query(2 * node + 1, mid + 1, r, ql, qr)
    return merge(left, right)

def main():
    input = sys.stdin.readline
    print = lambda x: sys.stdout.write(str(x) + '\n')

    n, m = map(int, input().split())

    for i in range(1, m + 1):
        l, r, x = map(int, input().split())
        updates[l].append((i, x))
        updates[r + 1].append((i, 0))

    q = int(input())
    for i in range(1, q + 1):
        k, s, t = map(int, input().split())
        queries[k].append((s, t, i))

    build(1, 1, m)

    for i in range(1, n + 1):
        for id_val in updates[i]:
            id_num, val = id_val
            point_update(1, 1, m, id_num, val)
        for query in queries[i]:
            l, r, id_num = query
```

```
            res = range_query(1, 1, m, l, r)
            answer[id_num] = res.mx_sum

    for i in range(1, q + 1):
        print(answer[i])

if __name__ == "__main__":
    main()
```