

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерных технологий

Продвинутые алгоритмы и структуры данных

Отчет о задаче № Е

Выполнила
студентка

Ершова А. И.

Группа № Р4115

Преподаватель: Косяков Михаил Сергеевич

г. Санкт-Петербург

2024

Условие задачи

Е. Фарид и XOR такси

Ограничение времени	1 секунда
Ограничение памяти	128.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

В ИТМО все знают про Фариду и его девушку. Он решил пойти с ней на свидание, но есть одна проблема — у него осталось мало денег.

Представим Санкт-Петербург в виде дерева из n узлов, где каждый узел обозначает место, а между некоторыми парами мест есть $n - 1$ двусторонних дорог, так что существует уникальный путь между любой парой мест.

У девушки Фариды есть странная особенность — ей нравятся цветы только из одного конкретного цветочного магазина, который находится в месте с номером f .

Недавно компания Тындекс запустила специальные машины, называемые "xor-cars". Особенность этих машин в том, что они предоставляют скидку, равную XOR-сумме весов дорог, по которым проходит маршрут. Иными словами, если вы едете из места x в место y по дорогам с длинами $w_0, w_1, w_2 \dots w_n$, то вы получите скидку, равную $(w_0 \text{ xor } w_1 \text{ xor } w_2 \text{ xor } \dots \text{ xor } w_n)$.

Фарид, будучи жадным, хочет узнать, какую максимальную скидку он может получить, если ему разрешено выбирать места, где он живет (x) и где живет его девушка (y), при этом соблюдаются следующие условия: $x \neq y$, $x \neq f$, $y \neq f$. Кроме того, ему нужно заехать в цветочный магазин в месте f , то есть сначала он берет Тындекс xor-car из места x и едет в место f , а затем из места f — в место y , где его ждет девушка.

Помогите Фариду найти максимальную скидку за такую поездку.

Формат ввода

Первая строка содержит два целых числа, разделенных пробелом, n и f , где n — это количество мест в Санкт-Петербурге, а f — номер места, где находится цветочный магазин.

Следующие $n - 1$ строк содержат по три целых числа $u[i]$, $v[i]$ и $w[i]$, которые обозначают, что между местом $u[i]$ и местом $v[i]$ существует двусторонняя дорога длиной $w[i]$.

$3 \leq n \leq 200,000$

$1 \leq f \leq n$

$1 \leq u[i], v[i] \leq n, (u[i] \neq v[i])$

$1 \leq w[i] \leq 1,000,000,000$


Формат вывода

Выведите одно целое число, максимальную скидку, которую может получить Фарид, если у него есть возможность выбрать места, в которых будут жить он и его девушка, и он поедет туда на машине Тындекс с функцией хог. (Не забудьте взять цветы :)))

Пример 1

Ввод 

```
5 1
2 1 4
2 3 9
1 4 8
1 5 5
```

Вывод 

```
13
```

Пример 2

Ввод 

```
3 2
2 1 60
2 3 50
```

Вывод 

```
14
```

Пример 1

Ввод 

17 3
HHHappyabNewQYear
Happy
New
Year

Вывод 

YES

Пример 2

Ввод 

9 2
HelloITMO
Hello
ITMO

Вывод 

NO

Решение:

```
#include <fstream>
#include <iostream>
#include <unordered_map>
#include <vector>

using namespace std;

vector<int> xorValues;
unordered_map<int, vector<pair<int, int>>> tree;

struct Road {
    int u, v, w;
};

class BinaryTrie {
private:
    struct Node {
        Node* child[2] = {nullptr, nullptr};
    };

    Node* rootNode;
```

```

public:
    void deleteTrie(Node* node) {
        if (!node)
            return;
        deleteTrie(node->child[0]);
        deleteTrie(node->child[1]);
        delete node;
    }

    BinaryTrie() {
        rootNode = new Node();
    }
    ~BinaryTrie() {
        deleteTrie(rootNode);
    }

    void addNum(int num) {
        Node* current = rootNode;
        for (int bitPosition = 31; bitPosition >= 0; --bitPosition) {
            int currentBit = (num >> bitPosition) & 1;
            if (!current->child[currentBit]) {
                current->child[currentBit] = new Node();
            }
            current = current->child[currentBit];
        }
    }

    int findMaxXor(int num) {
        Node* current = rootNode;
        int maxResult = 0;
        for (int bitPosition = 31; bitPosition >= 0; --bitPosition) {
            int currentBit = (num >> bitPosition) & 1;
            int preferredBit;
            if (currentBit == 1) {
                preferredBit = 0;
            } else
                preferredBit = 1;

            if (current && current->child[preferredBit]) {
                maxResult |= (1 << bitPosition);
                current = current->child[preferredBit];
            } else if (current) {
                current = current->child[currentBit];
            }
        }
        return maxResult;
    }
};

void dfs(int node, int parent, int xorSum) {

```

```

xorValues[node] = xorSum;
for (const auto& neighbor : tree.at(node)) {
    int nextNode = neighbor.first;
    int weight = neighbor.second;
    if (nextNode != parent) {
        int newXorSum = xorSum ^ weight;
        dfs(nextNode, node, newXorSum);
    }
}
}

int findMaxDiscount(int n, int f, const vector<Road>& roads) {
    for (const auto& road : roads) {
        tree[road.u].emplace_back(road.v, road.w);
        tree[road.v].emplace_back(road.u, road.w);
    }
    xorValues.resize(n + 1, -1);
    xorValues[f] = 0;

    dfs(f, -1, 0);

    BinaryTrie btrie;
    int maxDiscount = 0;

    for (int i = 1; i <= n; ++i) {
        if (i == f)
            continue;
        if (xorValues[i] != -1) {
            int maxXorForNode = btrie.findMaxXor(xorValues[i]);
            maxDiscount = max(maxDiscount, maxXorForNode);
            btrie.addNum(xorValues[i]);
        }
    }

    return maxDiscount;
}

int main() {
    int n;    // кол-во мест
    int f;    // номер места с цветочным
    cin >> n >> f;

    vector<Road> roads;
    for (int i = 0; i < n - 1; ++i) {
        int u;
        int v;
        int w;
        cin >> u >> v >> w;
        roads.push_back({u, v, w});
    }
}

```

```
int ans = findMaxDiscount(n, f, roads);

cout << ans << "\n";

return 0;
}
```

Сложность: $O(N)$.

Объяснение решения:

Сначала для обработки данных входные значения с информацией о доступных путях складываются в массив из структур Roads. Далее все входные значения попадают в основную функцию findMaxDiscount(). В ней значения сначала складываются в map, где ключом является номер места, а в качестве значения лежит массив со всеми доступными дорогами из этого места. Далее с помощью поиска в глубину заполняется массив xorValues, содержащий xor значения для всех элементов. Когда массив xorValues заполнен, начинается поиск максимальной скидки. С помощью класса BinaryTrie реализуется эффективный поиск максимального XOR-значения. Класс представляет собой префиксное дерево, где каждый узел имеет 2 потомка, соответствующих битам 0 и 1.

Для каждого места кроме цветочного (f) проверяется, есть ли его XOR-значение в массиве xorValues. Если оно существует, с помощью метода findMaxXor() находится максимальное XOR-значение между текущим местом и ранее добавленными значениями. Это достигается путём выбора на каждом уровне бита, который максимизирует текущий результат. Затем значение текущего места добавляется в дерево с помощью метода addNum(), чтобы использовать его при последующих вычислениях.

Методы addNum() и findMaxXor() работают на основе побитовой обработки чисел. Метод addNum() добавляет число в префиксное дерево, создавая узлы на каждом уровне, соответствующие битам числа. Метод findMaxXor() проходит по дереву, выбирая наиболее выгодный путь (побитовый XOR), чтобы максимизировать результат.

В результате прохождения всех мест вычисляется максимальная скидка.