Московский авиационный институт
(национальный исследовательский университет)

Институт информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Журнал по ознакомительной практике

Студент: Попов М. Р.
Учебная группа: М8О-108Б-20
Руководитель: Зайцев В. Е.
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021 год

Задание: посетить лекции по олимпиадному программированию, принять участие в соревнованиях, решить или дорешать не менее двенадцати задач, сдать зачёт по темам олимпиадного программирования, составить журнал.

Инструктаж по технике безопасности: _____/_____

Начало: 29.06.2021
Окончание: 12.07.2021

Протокол защиты

Слушали:	Постановили:
	Общая оценка_____

Председатель: _____ _____
(фамилия, имя и отчество) (подпись)

Члены: _____ _____
(фамилия, имя и отчество) (подпись)

_____ _____
(фамилия, имя и отчество) (подпись)

Сводная таблица

Дата	Название	Время проведения	Место проведения	Решено задач	Дорешено задач	Всего задач на соревновании
29.06.2021	Вводная лекция. Выдача задач	09:00 – 18:00	Дистанционно	—	—	—
30.06.2021	Ознакомление с практикой. Анализ полученной работы	09:00 – 18:00	Дистанционно	—	—	—
01.07.2021	Тестовая тренировка. Знакомство с системой	09:00 – 18:00	Дистанционно	—	—	—
02.07.2021	Основы C++	09:00 – 18:00	Дистанционно	10	0	10
03.07.2021	Стандартная библиотека C++	09:00 – 18:00	Дистанционно	10	0	12
05.07.2021	Динамическое программирование	09:00 – 18:00	Дистанционно	1	0	10
06.07.2021	Жадные алгоритмы	09:00 – 18:00	Дистанционно	1	0	10
07.07.2021	Основы теории графов, алгоритмы обхода	09:00 – 18:00	Дистанционно	1	0	8
08.07.2021	Кратчайшие пути во взвешенных графах	09:00 – 18:00	Дистанционно	1	0	8
09.07.2021	Алгоритмы на строках	09:00 – 18:00	Дистанционно	1	0	8
10.07.2021	Зачёт	09:00 – 18:00	Дистанционно	—	—	—
12.07.2021	Сдача журнала	13:00 – 18:00	МАИ ГУК 436 зона Б	—	—	—

Первое соревнование: Основы C++

Задача A: Числа

Ограничение по времени на тест: 1 секунда
Ограничение по памяти на тест: 256 мегабайт
Ввод: стандартный ввод
Вывод: стандартный вывод

Вам дан набор чисел, выведите их в отсортированном по возрастанию порядке.

Входные данные
В первой строке дано единственное число N ($1 \le N \le 2 \cdot 10^5$) — количество чисел в наборе. В следующей строке даны N чисел a_i ($1 \le a_i \le 10^9$) — числа, которые нужно отсортировать.

Выходные данные
Выведите заданные числа в отсортированном по возрастанию порядке.

Примеры

Входные данные	Выходные данные
10 4 5 8 2 9 6 1 3 7 10	1 2 3 4 5 6 7 8 9 10

Идея решения
Считаем число *N*, далее создадим вектор целых чисел размера *N* и считаем в него *N* чисел с помощью цикла *for*. Затем с помощью операции *sort*, содержащейся в заголовочном файле *algorithm* стандартной библиотеки языка программирования C++, отсортируем вектор и выведем все его элементы, начиная с нулевого, с помощью цикла *for*.

Код программы
```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main()
{
    int n;
    cin >> n;
    vector<int> v(n);
    for (int i = 0; i < n; i++)
```

```
        cin >> v[i];
    sort(v.begin(), v.end());
    for (int i = 0; i < n; i++)
        cout << v[i] << " ";
    cout << endl;
    return 0;
}
```

Сложность: $O(n \cdot \log n)$

Вердикт: полное решение.

Задача C: Рейтинг

Ограничение по времени на тест: 1 секунда
Ограничение по памяти на тест: 256 мегабайт
Ввод: стандартный ввод
Вывод: стандартный вывод

N команд приняли участие в соревновании. Известно, что i-я команда решила s_i задач и получила p_i штрафного времени. Ваша задача отсортировать все команды в порядке, в котором они должны быть представлены в турнирной таблице. Если количество задач, решённых командами, различно, то выше должна оказаться команда с большим количеством решённых задач. Если количество решённых задач у двух команд одинаково, то выше должна оказаться команда с меньшим штрафом. Если и количество решённых задач, и суммарный штраф у команд совпадают, то выше должна оказаться команда, название которой меньше в лексикографическом смысле.

Входные данные
В первой строке вам дано единственное число N $(1 \leq N \leq 10^5)$ — количество команд, участвовавших в соревновании. В следующих N строках вам даны строка $name$ и два числа s, p $(|name| \leq 20, 0 \leq s \leq 20, 0 \leq p \leq 3000)$ — название команды, состоящее из малых латинских букв, количество решённых задач и суммарный штраф.

Выходные данные
Выведите названия команд в порядке, в котором они должны быть представлены в турнирной таблице.

Примеры

Входные данные	Выходные данные
6 itsfine 8 422 redpanda 12 686 catsandunicorns 9 732 vinoizneudachnikov 10 1250 owo 0 0 nonames 8 459	redpanda vinoizneudachnikov catsandunicorns itsfine nonames owo
2 noone 0 0 someone 0 0	noone someone

Идея решения
Создадим структуру *team*, состоящую из строки и двух целых чисел, в которой будем хранить её название, количество решённых ей задач и количество штрафного времени. Далее считаем число N и создадим вектор, состоящий из

N структур типа *team*. С помощью цикла *for* считаем данные о всех командах. Затем создадим компаратор *cmp*, который будет сравнивать две структуры типа *team* и определять их положение в турнирной таблице относительно друг друга, и отсортируем структуры *team* в векторе с помощью операции *sort* и компаратора *cmp*. Наконец, с помощью цикла *for* выведем получившийся вектор, состоящий из структур типа *team*.

Код программы

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>

using namespace std;

struct team
{
    string name;
    int s;
    int p;
};

bool cmp(team &a, team &b)
{
    if (a.s > b.s)
        return 1;
    if (a.s < b.s)
        return 0;
    if (a.p < b.p)
        return 1;
    if (a.p > b.p)
        return 0;
    if (a.name < b.name)
        return 1;
    if (a.name >= b.name)
        return 0;
}

int main()
{
    int n;
    cin >> n;
    vector<team> v(n);
    for (int i = 0; i < n; i++)
        cin >> v[i].name >> v[i].s >> v[i].p;
    sort(v.begin(), v.end(), cmp);
```

```
    for (int i = 0; i < n; i++)
        cout << v[i].name << endl;
    return 0;
}
```

Сложность: $O(n \cdot \log n)$

Вердикт: полное решение.

Задача G: Ошибка в уставе

Ограничение по времени на тест: 1 секунда
Ограничение по памяти на тест: 256 мегабайт
Ввод: стандартный ввод
Вывод: стандартный вывод

Сидя в приемной у Гиммлера, Штирлиц начал читать различные выдержки из устава немецкой армии. «Ага, да здесь ошибка!» — подумал Штирлиц. И действительно, в одной из частей устава были неправильно расставлены скобки. Штирлиц решил поправить такой непорядок в немецкой армии и выписал все скобочные конструкции из устава с целью проверить их. Но тут его срочно вызвал Шелленберг, вследствие чего ошибки так и не были найдены. Помогите Штирлицу, чтобы его представили к награде рейхсфюрера СС.

Входные данные
В первой строке дано число N — количество скобок в части устава ($1 \leq N \leq 100000$). Затем дана строка из N символов, в которой содержатся лишь символы "(", ")", "{", "}", "[", "]".

Выходные данные
В единственной строке должно содержаться слово «Ja», если последовательность правильная, и «Nein» в противном случае. Ответ выводить без кавычек.

Примеры

Входные данные	Выходные данные
2 ()	Ja
2)(	Nein

Идея решения
Нетрудно догадаться, в каких случаях последовательность будет являться правильной. Во-первых, правильная последовательность не должна начинаться с закрывающей скобки. Во-вторых, правильная последовательность не должна заканчиваться открывающей скобкой. Помимо этого, необходимо учесть, что в правильной последовательности невозможен случай, когда за открывающей скобкой одного типа следует закрывающая скобка другого типа, например «(]» или «{)». Наконец, следует помнить, что в правильной последовательности количество открывающих и закрывающих скобок одного типа должно быть равным. С помощью оператора *if* и цикла *for* проверим введённую строку на все вышеперечисленные условия и определим, является ли последовательность правильной.

Код программы
```
#include <iostream>
```

```
#include <string>

using namespace std;

int main()
{
    int n;
    cin >> n;
    string s;
    cin >> s;
    if ((s[0] == ')')||(s[0] == '}')||(s[0] == ']'))
    {
        cout << "Nein";
        return 0;
    }
    if ((s[n - 1] == '(')||(s[n - 1] == '{')||(s[n - 1]
== '['))
    {
        cout << "Nein";
        return 0;
    }
    for (int i = 0; i < n; i++)
    {
        if ((s[i] == '(')&&((s[i + 1] == ']')||(s[i + 1]
== '}')))
        {
            cout << "Nein";
            return 0;
        }
        if ((s[i] == '[')&&((s[i + 1] == '}')||(s[i + 1]
== ')')))
        {
            cout << "Nein";
            return 0;
        }
        if ((s[i] == '{')&&((s[i + 1] == ']')||(s[i + 1]
== ')')))
        {
            cout << "Nein";
            return 0;
        }
    }
    int amcl = 0, amsl = 0, amfl = 0, amcr = 0, amsr =
0, amfr = 0;
    for (int i = 0; i < n; i++)
```

```
    {
        if (s[i] == '(')
            amcl++;
        if (s[i] == '[')
            amsl++;
        if (s[i] == '{')
            amfl++;
        if (s[i] == ')')
            amcr++;
        if (s[i] == ']')
            amsr++;
        if (s[i] == '}')
            amfr++;
    }
    if ((amcl != amcr)||(amsl != amsr)||(amfl != amfr))
    {
        cout << "Nein";
        return 0;
    }
    cout << "Ja";
    return 0;
}
```

Сложность: $O(n)$

Вердикт: полное решение.

Задача H: Магазин

Ограничение по времени на тест: 2 секунды
Ограничение по памяти на тест: 256 мегабайт
Ввод: стандартный ввод
Вывод: стандартный вывод

У Билла большая семья: трое сыновей, девять внуков. И всех надо кормить. Поэтому Билл раз в неделю ходит в магазин.
Однажды Билл пришел в магазин и увидел, что в магазине проводится акция под названием «каждый k-й товар бесплатно». Изучив правила акции, Билл выяснил следующее. Пробив на кассе товары, покупатель получает чек. Пусть в чеке n товаров, тогда n/k округленное вниз самых дешевых из них достаются бесплатно.
Например, если в чеке пять товаров за 200, 100, 1000, 400 и 100 рублей, соответственно, и $k = 2$, то бесплатно достаются оба товара по 100 рублей, всего покупатель должен заплатить 1600 рублей.
Билл уже выбрал товары, и направился к кассе, когда сообразил, что товары, которые он хочет купить, можно разбить на несколько чеков, и благодаря этому потратить меньше денег.
Помогите Биллу выяснить, какую минимальную сумму он сможет заплатить за выбранные товары, возможно разбив их на несколько чеков.

Входные данные
Первая строка входного файла содержит два целых числа n, k $(1 \le n \le 100000, 2 \le k \le 100)$ — количество товаров, которые хочет купить Билл и параметр акции «каждый k-й товар бесплатно». Следующая строка содержит n целых чисел a_i $(1 \le a_i \le 10000)$ — цены товаров, которые покупает Билл.

Выходные данные
Выведите в выходной файл одно число — минимальную сумму, которую должен заплатить Билл за товары.

Пример

Входные данные	Выходные данные
5 2 200 100 1000 400 100	1300

Идея решения
Создадим вектор целых чисел размера n и с помощью цикла *for* считаем в него цены товаров. Далее отсортируем вектор от большего числа к меньшему с помощью операции *sort* и заменим каждое k-ое число в этом векторе на 0. Найдём сумму вектора с помощью цикла *for*, которая будет являться ответом к задаче.

Код программы

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main()
{
    int n, k, ans = 0;
    cin >> n >> k;
    vector<int> v(n);
    for (int i = 0; i < n; i++)
        cin >> v[i];
    sort(v.rbegin(), v.rend());
    for(int i = k - 1; i < n; i += k)
        v[i] = 0;
    for (int i = 0; i < n; i++)
        ans += v[i];
    cout << ans;
    return 0;
}
```

Сложность: $O(n \cdot \log n)$

Вердикт: полное решение.

Второе соревнование: Стандартная библиотека C++

Задача A: Никогда не играйте с незнакомцами

Ограничение по времени на тест: 2 секунды
Ограничение по памяти на тест: 64 мегабайта
Ввод: стандартный ввод
Вывод: стандартный вывод

Фёдор очень любит гулять неподалёку от Патриарших прудов. Во время очередной прогулки к нему подошёл незнакомец и предложил сыграть в карты. Фёдор согласился, и незнакомец объяснил ему правила.
Игра происходит колодой из карт на каждой из которых написано некоторое число — сила карты, известно, что правильный набор карт включает в себя N карт с силами $A, A+1, A+2, \ldots, A+N-1$, где A выбирается произвольным образом. На каждом ходу атакующий игрок выбирает некоторый набор карт из своей руки и выкладывает их на стол, защищающийся игрок либо выкладывает некоторый набор карт с суммарной силой строго большей чем сила карт атакующего игрока и отбивает эту атаку, либо берёт атакующий набор карт себе. После этого игроки добирают из колоды случайные карты, таким образом чтобы в руке каждого стало не менее 10 карт. Если защищающийся отбил атаку, то игроки меняются ролями и начинается новый раунд, побеждает игрок, у которого в руке не осталось карт.
Фёдор крайне азартен и, к сожалению, проиграл кучу денег. Но он не готов признавать свои ошибки, по какой-то странной причине его оппонент оставил игральную колоду Фёдору, и Фёдор решил проверить, а действительно ли колода является правильной колодой для этой игры или же незнакомец его обманул.

Входные данные
В первой строке вам дано число N $(1 \leq N \leq 10^5)$ размер колоды. В следующей строке даны N чисел: силы карт в колоде $c_i (1 \leq c_i \leq 10^9)$.

Выходные данные
Если колода является корректной колодой для игры выведите "Deck looks good" без кавычек, в противном случае выведите "Scammed" без кавычек.

Примеры

Входные данные	Выходные данные
3 1 2 3	Deck looks good
3 10 100 1000	Scammed

Идея решения

Из условия задачи становится очевидным, что в отсортированной корректной колоде значение силы каждой карты, начиная со второй, ровно на 1 больше значения силы предыдущей карты. Чтобы проверить это, создаём вектор целых чисел размера N и считываем в него значения сил N карт. Далее сортируем полученный вектор и проверяем каждое его значение, кроме нулевого, на условие, описанное выше, с помощью цикла *for* и оператора *if*. Если нашли значение, при котором условие не выполняется, выводим сообщение о некорректности колоды, иначе выводим сообщение о корректности колоды.

Код программы

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int main()
{
                long long a, b, c, d;
                cin >> a;
                vector<int> m(a);
                for (int i = 0; i < a; i++)
                        cin >> m[i];
                sort (m.begin(), m.end());
                if (a == 1)
                {
                        cout << "Deck looks good";
                        return (0);
                }
                for (int i = 1; i < a; i++)
                {
                        if (m[i] - m[i - 1] != 1)
                        {
                         cout << "Scammed";
                         return(0);
                        }

                }
                cout << "Deck looks good";
                return 0;
}
```

Сложность: $O(n \cdot \log n)$

Вердикт: полное решение.

Задача E: Постфиксная запись

Ограничение по времени на тест: 0.25 секунд
Ограничение по памяти на тест: 256 мегабайт
Ввод: стандартный ввод
Вывод: стандартный вывод

Однажды в очереди в столовую Вася услышал разговор старшекурсников, в котором один из них упомянул постфиксную запись выражений. Нашему герою стало очень интересно, что это за такая запись. Поэтому он решил поискать про неё в интернете и вообще разобраться в вопросе.
Оказалось, постфиксная запись (или обратной польской записи) выражения - это такая запись, в которой операция записывается после двух операндов. Например, сумма двух чисел A и B записывается как $A\ B\ +$. Запись $B\ C\ +\ D\ *$ обозначает привычное нам $(B + C) * D$, а запись $A\ B\ C\ +\ D\ *\ +$ означает $A + (B + C) * D$.
Затем Вася узнал, что достоинство постфиксной записи в том, что она не требует скобок и дополнительных соглашений о приоритете операторов для своего чтения, и решил попробовать вычислить несколько выражений в таком виде. Для проверки своих ответов, он просит написать вас программу, которая будет вычислять написанные им выражения.

Входные данные
В единственной строке записано правильное выражение в постфиксной записи, содержащее цифры и операции +, -, *. Цифры и операции разделяются пробелами. Количество цифр в выражении не превышает 1000 .

Выходные данные
Выведите значение выражения, которое написал Вася. Результат и промежуточные вычисления не превышает 10^{18}.

Пример

Входные данные	Выходные данные
7 1 2 + * 3 9 - *	-126

Идея решения
Создадим стек, состоящий из строк, в который будем считывать данные. Если на ввод подаётся знак арифметической операции, достаём из стека две последние строки и переводим их в целые числа с помощью операции *stoll,* после чего совершаем над этими числами операцию, соответствующую поданному знаку, результат операции преобразуем в строку с помощью операции *to_string* и кладём эту строку в стек. Если в постфиксной записи записано правильное

выражение, в стеке останется единственная строка, в которой будет записано число, являющееся результатом выражения. Выведем эту строку и закончим работу программы.

Код программы

```
#include <iostream>
#include <stack>
#include <string>

using namespace std;

int main ()
{
    string a;
    string x, y, z;
    long long c, d, e;
    stack<string> s;
    while (cin >> a)
    {
        s.push(a);
        if (a == "+")
        {
            s.pop();
            y = s.top();
            s.pop();
            x = s.top();
            s.pop();
            c = stoll(y);
            d = stoll(x);
            e = d + c;
            z = to_string(e);
            s.push(z);
        }
        if (a == "-")
        {
            s.pop();
            y = s.top();
            s.pop();
            x = s.top();
            s.pop();
            c = stoll(y);
            d = stoll(x);
```

```
            e = d - c;
            z = to_string(e);
            s.push(z);
        }
        if (a == "*")
        {
            s.pop();
            y = s.top();
            s.pop();
            x = s.top();
            s.pop();
            c = stoll(y);
            d = stoll(x);
            e = d * c;
            z = to_string(e);
            s.push(z);
        }
    }
    cout << s.top();
    return 0;
}
```

Сложность: $O(n)$

Вердикт: полное решение.

Третье соревнование: Динамическое программирование

Задача A: Кузнечик

Ограничение по времени на тест: 2 секунды
Ограничение по памяти на тест: 64 мегабайта
Ввод: стандартный ввод
Вывод: стандартный вывод

Кузнечик Пётр живёт на числовой прямой и ему нужно попасть из точки 0 в точку n, он может прыгать только в сторону увеличения координат не более чем на k шагов, то есть первый прыжок он может осуществить только в точки $1,2,\dots,k$. Помогите ему определить сколькими путями он сможет это сделать, так как ответ может быть очень большой выведите его по модулю 10^9+7.

Входные данные
В единственной строке вам даны два числа n и $k(1 \le n,k \le 2 \cdot 10^4)$ — пункт назначения и максимальная длина прыжка кузнечика.

Выходные данные
Выведите единственное число — ответ на задачу.

Примеры

Входные данные	Выходные данные
10 2	89
20000 2	437241455

Идея решения
Чтобы определить количество путей из точки 0 в точку n, необходимо знать, сколько путей существует из точки 0 во все точки, лежащие перед точкой n. Для этого заведём вектор беззнаковых целых чисел, состоящий из $n+1$ элементов, примем нулевой и первый элемент за 1, так как в точки 0 и 1 существует только 1 путь из точки 0 при любых значениях k. Чтобы найти количество путей из точки 0 в точку i, необходимо сложить значения предыдущих k значений вектора. Вычислив k значений перед точкой n, найдём значение в самой точке n, это и будет являться количеством путей из точки 0 в точку n, то есть ответом к задаче.

Код программы
```
#include <iostream>
#include <vector>

using namespace std;
```

```
const int a = 1000000007;

int main()
{
    int n, k;
    unsigned long long s;
    cin >> n >> k;
    vector<unsigned long long> dp(n + 1);
    dp[0] = 1;
    dp[1] = 1;
    for (int i = 2; i <= n; i++)
    {
        for (int j = 1; j <= min(k, i); j++)
        {
            dp[i] += dp[i - j];
            if (dp[i] >= a)
                dp[i] %= a;
        }
    }
    s = dp[n];
    cout << s;
    return 0;
}
```

Сложность: $O(n \cdot k)$

Вердикт: полное решение.

Четвёртое соревнование: Жадные алгоритмы

Задача A: Суммы подотрезков

Ограничение по времени на тест: 2 секунды
Ограничение по памяти на тест: 64 мегабайта
Ввод: стандартный ввод
Вывод: стандартный вывод

Для заданного массива ответьте на запросы суммы на подотрезке массива.

Входные данные
В первой строке дано единственное число $N(1 \leq N \leq 2 \cdot 10^5)$ — количество элементов в массиве. В следующей строке даны N чисел, разделённых пробелом $a_i(|a_i| \leq 10^9)$ — элементы входного массива. В следующей строке дано число $Q(1 \leq Q \leq 2 \cdot 10^5)$ — количество запросов к вашей программе. В следующих Q строках заданы сами запросы в виде пар чисел, разделённых пробелом, l_i и $r_i(1 \leq l_i \leq r_i \leq n)$ — левая и правая граница запроса соответственно.

Выходные данные
Выведите Q чисел — ответы на запросы.

Пример

Входные данные	Выходные данные
3 1 -1 3 3 1 1 2 3 1 3	1 2 3

Идея решения
Создадим вектор целых чисел размером n и заполним его вводимыми числами с помощью цикла *for*. Далее для эффективного решения необходимо найти сумму каждого префикса введённого вектора. Для этого создадим вектор префиксных сумм и вычислим такую сумму для каждого значения введённого вектора с помощью цикла *for* (для вычисления i-го элемента вектора префиксных сумм сложим $(i-1)$-ое значение введённого вектора и $(i-1)$-ое значение вектора префиксных сумм). Затем считаем количество запросов Q, и с помощью цикла *for* ответим на каждый запрос. Чтобы найти сумму подотрезка с левыми и правыми границами l и r соответственно, из r-го значения вектора префиксных сумм вычтем $(l-1)$-ое значение вектора префиксных сумм.

Результат вычитания будет являться ответом на текущий запрос, который мы выведем на экран.

Код программы

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

using namespace std;

int main ()
{
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);
    int n, q, l, r;
    cin >> n;
    vector<long long> v(n);
    for (int i = 0; i < n; i++)
        cin >> v[i];
    vector<long long> prev(n+1);
    prev[0] = 0;
    for (int i = 1; i < n+1; i++)
        prev[i] = prev[i-1] + v[i-1];
    cin >> q;
    for (int i = 0; i < q; i++)
    {
        cin >> l >> r;
        cout << prev[r] - prev[l-1] << endl;
    }
    return 0;
}
```

Сложность: $O(n)$

Вердикт: полное решение.

Пятое соревнование: Основы теории графов, алгоритмы обхода

Задача B: Поиск в ширину

Ограничение по времени на тест: 1 секунда
Ограничение по памяти на тест: 256 мегабайт
Ввод: стандартный ввод
Вывод: стандартный вывод

Вам дан простой неориентированный граф, найдите в нём длины кратчайших путей от всех вершин до заданной.

Входные данные
В первой строке даны n, m и k $(1 \leq k \leq n \leq 100000, 0 \leq m \leq min(\frac{n(n-1)}{2}, 300000))$ — количество вершин и рёбер в графе, и номер вершины, расстояния до которой нужно найти, соответственно. Далее в m строках описаны рёбра графа в виде пар соединяемых ими вершин.

Выходные данные
Выведите n чисел — расстояния от каждой вершины до заданной вершины, если добраться из какой-либо вершины до заданной невозможно, то вместо расстояния выведите -1.

Примеры

Входные данные	Выходные данные
3 3 3 1 2 2 3 3 1	1 1 0
3 1 1 1 2	0 1 -1

Идея решения
Создадим вектор векторов целых чисел, в котором будем хранить рёбра графа в виде пар соединяемых ими вершин. Создадим вектор целых чисел, в котором будем хранить длины кратчайших путей от всех вершин до вершины k, причём $(k-1)$-ому значению этого вектора присвоим значение 0 (изначально вектор заполнен значениями -1, так как это значение необходимо вывести, если в данную вершину из заданной попасть невозможно). Далее создадим очередь, в которую будем класть номер вершины, в которую предстоит попасть при обходе на следующем этапе, и удалять номер этой вершины, когда мы в неё

попали при обходе. Сам обход будем осуществлять с помощью цикла *for*, при этом для каждой новой вершины будем определять длину кратчайшего пути, прибавляя 1 к длине кратчайшего пути от той вершины, из которой мы в неё попали. Обход графа будет закончен тогда, когда в очереди больше не останется вершин. С помощью цикла *for* выведем на экран длины кратчайших путей от всех вершин до заданной.

Код программы

```
#include <iostream>
#include <vector>
#include <queue>

using namespace std;

using graph = vector<vector<int>>;

int main()
{
    int n, m, k;
    cin >> n >> m >> k;
    k--;
    graph g(n);
    for (int i = 0; i < m; i++)
    {
        int a, b;
        cin >> a >> b;
        a--;
        b--;
        g[a].push_back(b);
        g[b].push_back(a);
    }
    queue<int> q;
    vector<int> lvl(n, -1);
    q.push(k);
    lvl[k] = 0;
    while (!q.empty())
    {
        int t = q.front();
        q.pop();
        for (int i = 0; i < g[t].size(); i++)
        {
            int h = g[t][i];
```

```
            if (lvl[h] == -1)
            {
                lvl[h] = lvl[t] + 1;
                q.push(h);
            }
        }
    }
    for (int i = 0; i < n; i++)
        cout << lvl[i] << " ";
    return 0;
}
```

Сложность: $O(n + m)$

Вердикт: полное решение.

Шестое соревнование: Кратчайшие пути во взвешенных графах

Задача B: Алгоритм Форда — Беллмана

Ограничение по времени на тест: 1 секунда
Ограничение по памяти на тест: 256 мегабайт
Ввод: стандартный ввод
Вывод: стандартный вывод

Задан неориентированный взвешенный граф, вершины которого пронумерованы от 1 до n. Ваша задача найти длины кратчайших путей от заданной вершины до всех остальных.

Входные данные
В первой строке вам дано числа n, m и $s (1 \leq n, m \leq 10^4, 1 \leq s \leq n)$ — количество вершин в графе, количество рёбер в графе, стартовая вершина. В следующих m строках вам даны рёбра в виде троек чисел $u, v, w (1 \leq u, v \leq n, 0 \leq w \leq 10^5)$ — пара вершин, соединяемых ребром, и его длина.

Выходные данные
Выведите n чисел d_i — длины кратчайших путей из заданной стартовой вершины до вершины i. Если пути до i не существует, выведите -1.

Пример

Входные данные	Выходные данные
5 4 1 1 4 3 1 3 1 3 4 1 5 4 2	0 -1 1 2 4

Идея решения
Создадим структуру для хранения информации о рёбрах графа (две вершины и вес ребра). Далее создадим вектор, состоящий из таких структур, и заполним его m рёбрами. Далее создадим вектор целых чисел, в котором будем хранить длины кратчайших путей от заданной вершины до всех остальных. Изначально s-ое значение этого вектора будет равно 0, а все остальные будут равны бесконечности (в нашей реализации вместо бесконечности возьмём число 10^9). Далее с помощью цикла *for* будем пытаться уменьшить длину кратчайшего пути для каждой вершины, зная, что длина пути будет кратчайшей, если в предпоследнюю вершину пути лежит также путь наименьшей длины. Чтобы

выйти из цикла создадим флаг типа *bool*, принимающий значение 1, если для хотя бы одной вершины был найден путь короче, чем при предыдущей итерации. Как только произойдёт итерация, после которой длина кратчайшего пути не изменится ни у одной вершины, флаг примет значение 0, что будет свидетельствовать о том, что алгоритм завершён и все длины кратчайших путей найдены. Выведем вектор длин кратчайших путей на экран, при этом заменяя значения 10^9 на -1 (для вершин, в которые не существует путей из заданной).

Код программы

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

using namespace std;

const int C = 1000000000;

struct edge
{
    int f;
    int t;
    int w;
};

int main()
{
    int n, m, s;
    cin >> n >> m >> s;
    s--;
    vector<edge> e;
    for (int i = 0; i < m; i++)
    {
        int u, v, w;
        cin >> u >> v >> w;
        u--;
        v--;
        e.push_back({u, v, w});
        e.push_back({v, u, w});
    }
    vector<long long> d(n, C);
    d[s] = 0;
    bool ch = 1;
    for (int i = 0; (i < n)&&(ch == 1); i++)
```

```
    {
        ch = 0;
        for (size_t j = 0; j < e.size(); j++)
        {
            edge cur_edge = e[j];
            int u = cur_edge.f;
            int v = cur_edge.t;
            int w = cur_edge.w;
            if (d[u] + w < d[v])
            {
                ch = 1;
                d[v] = d[u] + w;
            }
        }
    }
    for (int i = 0; i < n; i++)
    {
        if (d[i] == C)
            cout << "-1" << " ";
        else
            cout << d[i] << " ";
    }
    return 0;
}
```

Сложность: $O(n \cdot m)$

Вердикт: полное решение.

Седьмое соревнование: Алгоритмы на строках

Задача A: Z-функция

Ограничение по времени на тест: 1 секунда
Ограничение по памяти на тест: 256 мегабайт
Ввод: стандартный ввод
Вывод: стандартный вывод

Выведите $z-$функцию для заданной строки.

Входные данные
В первой строке дана строка $S(1 \leq |S| \leq 10^5)$ состоящая из маленьких латинских букв.

Выходные данные
В единственной строке выведите $|S|$ чисел через пробел — значения $z-$функции для заданной строки.

Примеры

Входные данные	Выходные данные
abacaba	7 0 1 0 3 0 1
abababa	7 0 5 0 3 0 1

Идея решения
Создадим функцию, принимающую на вход строку и возвращающую массив значений $z-$функции для этой строки. Инициализируем переменные l и r, в которых будем хранить начало и конец крайнего правого префикса, тогда используем цикл *for* и оператор *if* для вычисления $z-$функции i-ой позиции. Если $i \in [l..r]$, рассмотрим значение $z-$функции для позиции $j = i - l$, если $i + Z[j] \leq r$, то $Z[i] = Z[j]$, иначе найдём $Z[i]$ с помощью цикла *while*, перебирая все символы. Если $i \notin [l..r]$, найдём $Z[i]$ с помощью цикла *while*, перебирая все символы.

Код программы
```
#include <iostream>
#include <vector>
#include <string>

using namespace std;
```

```
vector<int> z_function (string s)
{
    int n = (int) s.length();
    vector<int> z(n);
    for (int i = 1, l = 0, r = 0; i < n; i++)
    {
        if (i <= r)
            z[i] = min(r - i + 1, z[i - l]);
        while ((i + z[i] < n)&&(s[z[i]] == s[i + z[i]]))
            z[i]++;
        if (i + z[i] - 1 > r)
        {
            l = i;
           r = i + z[i] - 1;
        }
    }
    return z;
}

int main()
{
    string s;
    cin >> s;
    vector<int> r = z_function(s);
    r[0] = s.size();
    for (int a: r)
       cout << a << " ";
    return 0;
}
```

Сложность: $O(|S|)$

Вердикт: полное решение.