Министерство науки и высшего образования
Российской Федерации

Московский авиационный институт
(национальный исследовательский университет)

Институт компьютерных наук и прикладной математики

Кафедра вычислительной математики и программирования

Журнал по ознакомительной практике

Студент: Попов Н. А.
Группа: М8О-108Б-21

Оценка:

Дата:

Подпись:

Москва, 2022

ИНСТРУКЦИЯ

о заполнении журнала по производственной практике

Журнал по производственной практике студентов имеет единую форму для всех видов практик.

Задание в журнал вписывается руководителем практики от института в первые три-пять дней пребывания студентов на практике в соответствии с тематикой, утверждённой на кафедре до начала практики. Журнал по производственной практике является основным документом для текущего и итогового контроля выполнения заданий, требований инструкции и программы практики.

Табель прохождения практики, задание, а также технический отчёт выполняются каждым студентом самостоятельно.

Журнал заполняется студентом непрерывно в процессе прохождения всей практики и регулярно представляется для просмотра руководителям практики. Все их замечания подлежат немедленному выполнению.

В разделе «Табель прохождения практики» ежедневно должно быть указано, на каких рабочих местах и в качестве кого работал студент. Эти записи проверяются и заверяются цеховыми руководителями практики, в том числе мастерами и бригадирами. График прохождения практики заполняется в соответствии с графиком распределения студентов по рабочим местам практики, утверждённым руководителем предприятия. В разделе «Рационализаторские предложения» должно быть приведено содержание поданных в цехе рационализаторских предложений со всеми необходимыми расчётами и эскизами. Рационализаторские предложения подаются индивидуально и коллективно.

Выполнение студентом задания по общественно-политической практике заносятся в раздел «Общественно-политическая практика». Выполнение работы по оказанию практической помощи предприятию (участие в выполнении спецзаданий, работа сверхурочно и т.п.) заносятся в раздел журнала «Работа в помощь предприятию» с последующим письменным подтверждением записанной работы соответствующими цеховыми руководителями. Раздел «Технический отчёт по практике» должен быть заполнен особо тщательно. Записи необходимо делать чернилами в сжатой, но вместе с тем чёткой и ясной форме и технически грамотно. Студент обязан ежедневно подробно излагать содержание работы, выполняемой за каждый день. Содержание этого раздела должно отвечать тем конкретным требованиям, которые предъявляются к техническому отчёту заданием и программой практики. Технический отчёт должен показать умение студента критически оценивать работу данного производственного участка и отразить, в какой степени студент способен применить теоретические знания для решения конкретных производственных задач.

Иллюстративный и другие материалы, использованные студентом в других разделах журнала, в техническом отчёте не должны повторяться, следует ограничиваться лишь ссылкой на него. Участие студентов в производственно-технической конференции, выступление с докладами, рационализаторские предложения и т.п. должны заноситься на свободные страницы журнала.

Примечание. Синьки, кальки и другие дополнения к журналу могут быть сделаны только с разрешения администрации предприятия и должны подшиваться в конце журнала.

Руководители практики от института обязаны следить за тем, чтобы каждый цеховой руководитель практики перед уходом студентов из данного цеха в другой цех вписывал в журнал студента отзывы об их работе в цехе.

Текущий контроль работы студентов осуществляется руководители практики от института и цеховыми руководителями практики заводов. Все замечания студентам руководители делают в письменном виде на страницах журнала, ставя при этом свою подпись и дату проверки.

Результаты защиты технического отчёта заносятся в протокол и одновременно заносятся в ведомость и зачётную книжку студента.

Примечание. Нумерация чистых страниц журнала проставляется каждым студентом в своём журнале до начала практики.

С инструкцией о заполнении журнала ознакомлены:

« » _____ 2021 г. Студент Попов Н. А. _____

(дата) (подпись)

ЗАДАНИЕ

Принять участие в тренировках и соревнованиях по олимпиадному программированию для студентов первого курса в 2021/2022 учебном году: посетить и проработать установочные лекции, решать и дорешивать конкурсные задания, принять участие в разборе. Объём практики 108 часов.

Руководитель практики от института:

« » _____ 2021 г.
(дата)

_____
(подпись)

ТАБЕЛЬ ПРОХОЖДЕНИЯ ПРАКТИКИ

№	Дата	Название контеста	Время проведения	Место проведения	Решено задач	Дорешано задач	Подпись
1	30.06.2022	Основы C++	15:00 - 19:00	Дистанционно	11	1	
2	01.07.2022	Библиотека C++	12:00 - 17:00	Дистанционно	7	x	
3	02.07.2022	Динамическое программирование	12:00 - 17:00	Дистанционно	0	x	
4	04.07.2022	Префиксные суммы, сортировка событий, два указателя	12:00 - 17:00	Дистанционно	0	x	
5	05.07.2022	ДП, задача о рюкзаке	12:45 - 17:45	Дистанционно	0	x	
6	06.07.2022	Длинная арифметика	12:45 - 17:45	Дистанционно	2	x	
7	07.07.2022	Основы теории графов	12:00 - 17:00	Дистанционно	2	x	
8	08.07.2022	Кратчайшие пути во взвешенных графах	12:15 - 17:15	Дистанционно	5	3	
9	09.07.2022	Алгоритмы на строках	12:00 - 17:00	Дистанционно	5	0	
10	12.07.2022 Защита практики	Оформление журнала.	9:00 - 18:00	Дистанционно			
		Итого часов	108				

Отзывы цеховых руководителей практики

Принято участие в N контестах, прослушаны установочные лекции и разборы задач, решено M и дорешано K задач контестов, оформлен журнал практики с электронным приложением. Задание практики выполнено. Рекомендую оценку

Тренер Инютин М. А. _____
(подпись)

Работа в помощь предприятию

Встречи с представителями ИТ-компаний, сотрудничающих с МАИ.

ПРОТОКОЛ

ЗАЩИТЫ ТЕХНИЧЕСКОГО ОТЧЁТА

по *ознакомительной практике*

студентом: Поповым Николаем Александровичем

Слушали:	**Постановили:**
Отчёт практиканта	Считать практику выполненной и защищённой на Общая оценка: _____

Председатель: Зайцев В. Е. _____
Члены: Сорокин С. А. _____
Инютин М. А. _____

Дата: 12 июля 2022 г.

ТЕХНИЧЕСКИЙ ОТЧЁТ ПО ПРАКТИКЕ

Основы C++

B. Развороты

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Вам дан массив целых чисел и набор запросов. Каждый запрос задаётся парой целых чисел l и r, от вас требуют развернуть подмассив, начинающийся в позиции l и заканчивающийся в позиции r.

Входные данные
В первой строке вам задано единствееное число n $(1 \leqslant n \leqslant 1000)$ — число элементов в массиве.

В следующей строке вам заданы сами элементы массива a_i $(|a_i| \leqslant 10^9)$.

В следующих строках вам задаются запросы по одному в строке в виде пары чисел разделённой пробелом l_i и r_i $(1 \leqslant l_i \leqslant r_i \leqslant n)$.

Последний запрос состоит из пары нулей, и обрабатывать его не требуется. Количество запросов не превышает 1000.

Выходные данные
В единственной строке выведите массив, который будет получен после обработки всех запросов.

Пример

входные данные	Скопировать
5 1 2 3 4 5 1 5 2 3 0 0	

выходные данные	Скопировать
5 3 4 2 1	

Идея решения

В первую очередь мы вводим количество элементов в массиве, затем – последовательность в массив, а затем записываем порядки элементов, которые мы поменяем местами. Далее мы производим операцию swap нужных элементов (не забывая увеличивать индекс начала подполследовательности и уменьшать индекс ее конца). Далее выводим отсортированную последовательность. Сложность алгоритма $O(m \cdot n)$, где m - количество запросов, а n - длина разворачиваемой подпоследовательности

Исходный код

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

using namespace std;

// vector print function
void print_vector(vector <long long> v)
{
    for (int i = 0; i < v.size(); i++) {
```

```
        cout << v[i];
        if (i != v.size() - 1) {
            cout << " ";
        }
    }
}

int main(void)
{
    int n;
    cin >> n;
    // input vector
    vector<long long> a(n);
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }

    int l, r;
    while (1) {
        cin >> l >> r;
        if (l == 0 && r == 0) break;
        l--; r--;
        // Reversing a subarray from l to r
        while (l < r) {
            swap(a[l], a[r]);
            l++;
            r--;
        }
    }
    // result output
    print_vector(a);
    return 0;
}
```

Выводы

Задача решена.

C. Подстроки

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Родители решили подарить Васе на день рождения две строки, причём такие, чтобы одна была заметно меньше второй. Они знают, что Васе будет не интересно играть с этими строками, если одна будет слишком часто встречаться в другой. Ваша задача помочь родителям Васи подобрать подарок, для того чтобы определить качество подарка, они хотят знать, сколько раз меньшая строка встречается в большей.

Входные данные

В первой строке вам даны два целых числа n $(1 \leqslant n \leqslant 10^5)$ и m $(1 \leqslant m \leqslant 10^3, m \leqslant n)$ — длины строк.

В следующих двух строках даны сами строки, состоящие из маленьких английских букв.

Выходные данные

Выведите единственное число — число подстрок из первой строки, совпадающих со второй строкой.

Примеры

входные данные	Скопировать
4 2 aaaa aa	

выходные данные	Скопировать
3	

входные данные	Скопировать
5 4 abcab abcd	

выходные данные	Скопировать
0	

Идея решения

Для хранения строки и подстроки будеем использовать вектор типа *char* Далее используем алгоритм прямого поиска подстроки в строке.

Алгоритм прямого поиска подстроки в строке:

1. i = 1
2. сравнить i-й символ массива T с первым символом массива W
3. совпадение → сравнить вторые символы и так далее
4. несовпадение → i:=i+1 и переход на пункт 2

Сложность алгоритма $O(n \cdot m)$, где m - длина строки, а n - длина подстроки

Исходный код

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

```

```cpp
using namespace std;

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

    int n1, n2;
    cin >> n1 >> n2;

    // first string
    vector<char> str1(n1);
    // second string
    vector<char> str2(n2);

    // first string input
    for (int i = 0; i < str1.size(); i++) {
        cin >> str1[i];
    }
    // second string input
    for (int i = 0; i < str2.size(); i++) {
        cin >> str2[i];
    }

    int p1 = 0, ch = 0;
    int p2 = str2.size();
    bool flag = false;

    // direct search
    for (int i = 0; i < str1.size() - str2.size() + 1; i++) {
        flag = true;
        for (int j = 0; j < str2.size(); ++j)
            if (str1[j + p1] != str2[j])
                flag = false;

        if (flag) {
            ch++;
        }
        p1++;
    }
    // result output
    cout << ch << endl;
    return 0;
}
```

Выводы

Задача решена.

Фрагмент турнирной таблицы контеста

4	Попов Николай Александрович М8О-108Б-21	11	1627	+1 01:27	+1 02:32	+ 01:50	+ 01:35	+2 01:47	+ 02:04	+ 02:13	+ 02:58	+ 02:37	+ 02:51	+1 03:33	-2

Библиотека C++

A. Никогда не играйте с незнакомцами

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
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 $(1 \leqslant N \leqslant 10^5)$ размер колоды. В следующей строке даны N чисел: силы карт в колоде c_i $(1 \leqslant c_i \leqslant 10^9)$.

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	Скопировать
3 1 2 3	
выходные данные	**Скопировать**
Deck looks good	

входные данные	Скопировать
3 10 100 1000	
выходные данные	**Скопировать**
Scammed	

Идея решения

В первую очередь вводим последовательность в вектор. Затем мы проверяем, заряжена ли колода в киосках: если в колоде одна карта или набор карт с значениями силы как в условии задачи (разница между соседними картами равна единице), то с колодой всё в порядке. В противном случае Федю обманули.

Исходный код

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
```

```cpp

using namespace std;

int main()
{
    int n;
    cin >> n;
    vector<int> a(n);
    // input
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }

    // sorting vector
    sort(a.begin(), a.end());
    for (int i = 1; i < n; i++) {
        if (a[i] != a[i - 1] + 1) {
            // result output
            cout << "Scammed";
            return 0;
        }
    }
    // result output
    cout << "Deck looks good";
    return 0;
}
```

Выводы

Задача решена.

Фрагмент турнирной таблицы контеста

4	Попов Николай Александрович М8О-108Б-21	11	1627	+1 01:27	+1 02:32	+ 01:50	+ 01:35	+2 01:47	+ 02:04	+ 02:13	+ 02:58	+ 02:37	+ 02:51	+1 03:33	-2

Динамическое программирование

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 $1, 2, \ldots, k$. 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	Скопировать
10 2	
выходные данные	Скопировать
89	

входные данные	Скопировать
20000 2	
выходные данные	Скопировать
437241455	

Идея решения

Чтобы решить эту задачу, нам нужно составить таблицу результатов при движении на определённые координаты до значения k. Значение ячейки в 0 равно единице, Для дальнейших ячеек необходимо добавить следующий шаг в цикле (i - j), а затем взять остаток от деления на m. После цикла выводим результат из последней ячейке массива.

Исходный код

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

using namespace std;

const long long MOD = (int) 1e9 + 7;

int main()
{
    int n, k;
    cin >> n >> k;

    vector<long long> dp(n + 1);
    dp[0] = 1;
    for (int i = 1; i <= n; i++) {
```

```
        int j = 1;
        while (j <= k && i - j >= 0) {
            dp[i] += dp[i - j];
            // Avoiding overflow of number by saving modulo number
            dp[i] %= MOD;
            j++;
        }
    }
    // result output
    cout << dp[n] << endl;
    return 0;
}
```

Выводы

Задача решена.

G. Преобразуй число

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Имеется натуральное число N. За один ход можно вычесть из него единицу, поделить на два или поделить на три. Делить можно только нацело. Цена каждого хода — само число, над которым производится операция. Ваша задача — преобразовать число N в единицу за минимальную стоимость.

Входные данные

В единственной строке находится натуральное число N ($2 \leqslant N \leqslant 2 \cdot 10^7$)

Выходные данные

Выведите единственное число — ответ на задачу.

Пример

входные данные	Скопировать
82	
выходные данные	Скопировать
202	

Примечание

В первом тестовом примере следует сначала вычесть единицу, а потом делить на три. Получим $82 + 81 + 27 + 9 + 3 = 202$.

Идея решения

Чтобы решить эту задачу составим таблицу из n чисел. В i-ой ячейке массива будем хранить количество способов получить из числа единицу. Заполняем массив с конца, динамически добавляя к i-ой ячейке значения массива с индексами $i \cdot 2$, $i \cdot 3$, $i \cdot 1$. После заполнения массива выводим результат - dp[1]. Сложность алгоритма - $O(n)$.

Исходный код

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

```cpp

using namespace std;

int main()
{
    // input data
    int n;
    cin >> n;

    vector<long long> dp(n + 1, INT_MAX);
    dp[n] = 0;
    for (int i = n - 1; i >= 0; i--) {
        // add to the cell the number of ways from cells i*2, i*3, i+1
        if (dp[i + 1] + i + 1 < dp[i]) {
            dp[i] = dp[i + 1] + i + 1;
        }
        if (i * 3 <= n) {
            dp[i] = min(dp[i * 3] + i * 3, dp[i]);
        }
        if (i * 2 <= n) {
            dp[i] = min(dp[i * 2] + i * 2, dp[i]);
        }
    }
    // result output
    cout << dp[1] << endl;
    return 0;
}
```

Выводы

Неправильный ответ на тесте 35. Ошибка из-за переполнения типа *int*. Изменил тип на *long long*. Задача решена.

Фрагмент турнирной таблицы контеста

4	Попов Николай Александрович М8О-108Б-21	11	1627	+1 01:27	+1 02:32	+ 01:50	+ 01:35	+2 01:47	+ 02:04	+ 02:13	+ 02:58	+ 02:37	+ 02:51	+1 03:33	-2

Префиксные суммы, сортировка событий, два указателя

A. Суммы подотрезков

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт
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 ($1 \leqslant N \leqslant 2 \cdot 10^5$) — количество элементов в массиве. В следующей строке даны N чисел разделённых пробелом a_i ($|a_i| \leqslant 10^9$) — элементы входного массива. В следующей строке дано число Q ($1 \leqslant Q \leqslant 2 \cdot 10^5$) — количество запросов к вашей программе. В следующих Q строках заданы сами запросы в виде пар чисел разделённых пробелом l_i и r_i ($1 \leqslant l_i \leqslant r_i \leqslant n$) — левая и правая граница запроса соответственно.

Выходные данные

Выведите Q чисел — ответы на запросы.

Пример

входные данные	Скопировать
3 1 -1 3 3 1 1 2 3 1 3	

выходные данные	Скопировать
1 2 3	

Идея решения

Вводим число n. Затем создаем массив a длины n и заполняем его. Далее составляем массив префиксных сумм $pref$ в котором будем хранить перфиксные суммы для введенного массива.

$$pref[0] = 0,$$
$$pref[j] = pref[j-1] + a[j]$$

Затем вводим q пар чисел - левая и правая границы запроса. Для ответа на запрос пользуемся формулой:

$$\text{Сумма подотрезка } [l, r] = pref[r] - pref[l-1]$$

Сложность заполнения масива $pref$: $O(n)$
Общая сложность алгоритма: $O(n+q)$

Исходный код

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

using namespace std;
```

```cpp

// function for printing vector in one line
void print_vector(vector<int> v) {
    for (int i = 0; i < v.size(); i++) {
        cout << v[i] << " ";
    }
    cout << endl;
}

int main()
{
    // fast input
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    int n, q;
    cin >> n;

    vector<long long> a(n);
    for (int i = 0; i < n; i++) {
        cin >> a[i];
    }
    // vecotor for containing prefix sum
    vector<long long> pref(n);
    pref[0] = a[0];

    for (int j = 0; j < n; j++) {
        pref[j] = pref[j - 1] + a[j];
    }

    // number of queris
    cin >> q;
    for (int i = 0; i < q; i++) {
        int l, r;
        cin >> l >> r;
        l--; r--;
        // output query result
        cout << pref[r] - pref[l - 1] << endl;
    }
    return 0;
}
```

Выводы

Задача решена.

Фрагмент турнирной таблицы контеста

4	Попов Николай Александрович М8О-108Б-21	11	1627	+1 01:27	+1 02:32	+ 01:50	+ 01:35	+2 01:47	+ 02:04	+ 02:13	+ 02:58	+ 02:37	+ 02:51	+1 03:33	-2

Длинная арифметика

B. Умножение

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт
ввод: стандартный ввод
вывод: стандартный вывод

Вам заданы два целых неотрицательных числа A и B. Выведите $A \cdot B$.

Входные данные
Первая строка входных данных содержит целое неотрицательное число A.

Вторая строка входных данных содержит целое неотрицательное число B.

Число разрядов в числах не превышает $3 \cdot 10^5$.

Выходные данные
Выведите $A \cdot B$.

Примеры

входные данные	Скопировать
2 2	
выходные данные	**Скопировать**
4	

входные данные	Скопировать
12345678987654321 98765432123456789	
выходные данные	**Скопировать**
1219326320073159566072245112635269	

Идея решения

Храним числа, которые невозможно вместить в стандартный целочисленный тип в виде строк. Для умножения используем быстрое преобразование Фурье.

Сложность алогоритма: $O(n \cdot log_2 n)$

Исходный код

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

using complex = std::complex<double>;
using vc = std::vector<complex>;

const int BASE = 10;
const double PI = std::acos(-1);

int rev_bits(int x, int lg_n) {
    int y = 0;
    for (int i = 0; i < lg_n; ++i) {
        y = y << 1;
```

```
        y ^= (x & 1);
        x = x >> 1;
    }
    return y;
}

void fft(vc & a, bool invert)
{
    int n = a.size();
    int lg_n = 0;

    while ((1 << lg_n) < n)
        ++lg_n;

    for (int i = 0; i < n; ++i)
        if (i < rev_bits(i, lg_n))
            swap(a[i], a[rev_bits(i, lg_n)]);

    for (int layer = 1; layer <= lg_n; ++layer) {
        int cluster = 1 << layer;
        double phi = (2.0 * PI) / cluster;
        if (invert)
            phi *= -1;

        complex wn = complex(std::cos(phi), std::sin(phi));

        for (int i = 0; i < n; i += cluster) {
            complex w(1);
            for (int j = 0; j < cluster / 2; ++j) {
                complex u = a[i + j];
                complex v = a[i + j + cluster / 2] * w;
                a[i + j] = u + v;
                a[i + j + cluster / 2] = u - v;
                w *= wn;
            }
        }
    }

    if (invert)
        for (int i = 0; i < n; ++i)
            a[i] /= n;
}


std::string fft_mult(const std::string & a, const std::string & b) {
    size_t max_size = std::max(a.size(), b.size());
    size_t n = 1;

    while (n < max_size)
        n *= 2;

    n *= 2;
    vc fa(n), fb(n);

    for (size_t i = 0; i < a.size(); ++i)
        fa[a.size() - i - 1] = complex(a[i] - '0');

    for (size_t i = 0; i < b.size(); ++i)
        fb[b.size() - i - 1] = complex(b[i] - '0');

```

```
    fft(fa, false);
    fft(fb, false);

    for (size_t i = 0; i < n; ++i)
        fa[i] = fa[i] * fb[i];

    fft(fa, true);

    std::vector<int> res(n);
    for (size_t i = 0; i < n; ++i)
        res[i] = (int64_t)round(fa[i].real());

    for (size_t i = 0; i < n - 1; ++i) {
        res[i + 1] += res[i] / BASE;
        res[i] %= BASE;
    }

    while (res.size() > 1 and res.back() == 0)
        res.pop_back();

    std::reverse(res.begin(), res.end());
    std::string ab;

    for (int64_t elem : res)
        ab.push_back('0' + elem);

    return ab;
}

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

    std::string a, b;
    std::cin >> a >> b;
    std::string fft_res = fft_mult(a, b);
    std::cout << fft_res << std::endl;

    return 0;
}
```

Выводы

Задача решена.

Фрагмент турнирной таблицы контеста

7	Попов Николай Александрович М8О-108Б-21	2	489	+2 03:36	+ 03:53		-5	

Основы теории графов

A. Поиск в глубину

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 64 мегабайта
ввод: стандартный ввод
вывод: стандартный вывод

Вам дан простой неориентированный граф, выведите для каждой вершины её номер в порядке обхода в глубину. Рёбра, исходящие из вершины, следует перебирать в порядке, в котором они заданы во входном файле.

Входные данные

В первой строке даны n, m и k $\left(1 \le k \le n \le 100000,\ 0 \le m \le \min\left(\frac{n(n-1)}{2}, 300000\right)\right)$ — количество вершин и рёбер в графе и номер вершины, с которой следует начинать обход соответственно. Далее в m строках описаны рёбра графа в виде пар соединяемых ими вершин a и b $(1 \le a, b \le n)$

Выходные данные

Выведите n чисел — номера вершин в порядке обхода в глубину от заданной вершины. Стартовая вершина имеет номер 0. Если добраться из какой-либо вершины до заданной невозможно, то вместо номера выведите -1.

Примеры

входные данные	Скопировать
3 3 3 1 2 2 3 3 1	

выходные данные	Скопировать
2 1 0	

входные данные	Скопировать
3 1 1 1 2	

выходные данные	Скопировать
0 1 -1	

Идея решения

Храним граф в виде вектора векторов целых чисел. Заполняем граф на основе входных данных. Создаем дополнительный вектор *depth* размера n, в котором будем хранить глубину каждой вершины. При обходе в глубину прежде чем переходить к следующей вершине инкрементируем текущую глубину и присваиваем ее соответсвующей вершине.

Сложность алгоритма: $O(n + m)$, где n - число вершин, а m - число ребер графа.

Исходный код

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

using namespace std;
using graph = vector< vector<int> >;

// depth-first search
void dfs(int v, const graph &g, vector<int> &depth, int &curr)
```

```
{
    if (depth[v] != -1)
        return;
    depth[v] = ++curr;
    for (int u: g[v]) {
        dfs(u, g, depth, curr);
    }
}

int main()
{
    // fast input
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);

    int n, m, k;
    cin >> n >> m >> k;
    k--;

    graph g(n);
    // input graph
    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);
    }

    // vector that contains depht of each vertex
    vector<int> depth(n, -1);
    int curr = -1;

    dfs(k, g, depth, curr);

    // result output
    for (int d : depth) {
        cout << d << " ";
    }
    cout << endl;
    return 0;
}
```

Выводы

Задача решена.

Фрагмент турнирной таблицы контеста

16	Попов Николай Александрович М8О-108Б-21	2	241	+2 01:34	+ 01:47	-2		-1			

Кратчайшие пути во взвешенных графах

B. Алгоритм Флойда — Уоршелла

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 между всеми парами вершин.

Входные данные

В первой строке вам дано число n $(1 \le n \le 500)$ — количество вершин в графе. В следующих n строках вам даны по n чисел a_{ij} $(0 \le a_{ij} \le 10^9,\ a_{ii} = 0)$ — длины рёбер из i-й вершины в j-ю.

Выходные данные

Выведите n строк по n чисел d_{ij} — длины кратчайших путей из вершины i в вершину j.

Примеры

входные данные	Скопировать
1 0	

выходные данные	Скопировать
0	

входные данные	Скопировать
3 0 10 1 1 0 1 1 1 0	

выходные данные	Скопировать
0 2 1 1 0 1 1 1 0	

Идея решения

Создадим специальную структуру для хранения ребер графа. В ней опишем пару вершин, которую это ребро соединяет, а также вес этого ребра.

Храним граф в виде вектора векторов ребер. Заполняем граф на основе входных данных. Создаем таблицу для хранения кратчайших расстояний между вершинами. Заполняем ее, используя алгоритм Флойда-Уоршелла. Затем выводим n строк по n чисел d_{ij} — длины кратчайших путей из вершины i в вершину j.

Временная сложность: $O(n^3)$
Пространственная сложность: $O(n^2)$

Исходный код

```
1 #include <iostream>
2 #include <vector>
3
```

```cpp
using namespace std;

const int64_t INF = 1e18;

// weighted edge
struct wedge {
    int u, v;
    int64_t w;
};

int main(void)
{
    int n;
    cin >> n;
    vector<wedge> g;
    // table of distances between all vertices
    vector< vector <int64_t> > d(n, vector<int64_t> (n, INF));
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (i == j) {
                d[i][j] = 0;
            }
            int64_t w;
            cin >> w;
            d[i][j] = w;
        }
    }
    // Floyd-Warshall algorithm
    for (int k = 0; k < n; ++k) {
        for (int i = 0; i < n; ++i) {
            for (int j = 0; j < n; ++j) {
                d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
            }
        }
    }
    // result output
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << d[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

Выводы

Задача решена.

Фрагмент турнирной таблицы контеста

5	Попов Николай Александрович М8О-108Б-21	5	736	+ 01:30	+ 01:48	+ 02:37	+ 03:00	+ 03:21	-4	

Алгоритмы на строках

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 $(1 \le |S| \le 10^5)$ состоящая из маленьких латинских букв.

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

Примеры

входные данные	Скопировать
abacaba	
выходные данные	Скопировать
7 0 1 0 3 0 1	

входные данные	Скопировать
abababa	
выходные данные	Скопировать
7 0 5 0 3 0 1	

Идея решения

В процессе вычисления Z-функции поддерживать последнее ненулевое найденное значение в виде границ отрезка [l;r], равного соответствующему префиксу. Под “последним” значением понимается отрезок с наибольшим r.

Таким образом, в качестве начального значения $z[i]$ можно использовать:

$$z[i] = min(z[i-l], r-i+1)$$

После чего запускаем наивный алгоритм, пытаясь увеличить $z[i]$. Это возможно, если правая граница текущего отрезка совпадения превышает r, или если i не входит в $[l;r]$, и вычислять $z[i]$ необходимо с нуля.

Если в результате правая граница текущего отрезка $(i + z[i] - 1)$ превысила r, обновляем значения l и r.

Сложность такого алгоритма равна $O(n)$.

Исходный код

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

using namespace std;

```

```cpp
// z-function of string calculation function
vector<int> z_func(const string &s)
{
    int n = s.size();
    vector<int> z(n);
    z[0] = n;
    int l = 0;
    int r = 0;

    for (int i = 1; i < n; i++) {
        if (i <= r) {
            z[i] = min(z[i - l], r - i + 1);
        }
        while (i + z[i] < n and s[z[i]] == s[i + z[i]]) {
            ++z[i];
        }
        if (i + z[i] >= r) {
            l = i;
            r = i + z[i] - 1;
        }
    }
    return z;
}

int main(void)
{
    string s;
    cin >> s;
    vector<int> z = z_func(s);
    // result output
    for (int i = 0; i < s.size(); i++) {
        cout << z[i] << " ";
    }
    cout << endl;
    return 0;
}
```

Выводы

Задача решена.

Фрагмент турнирной таблицы контеста

28	Попов Николай Александрович М8О-108Б-21	1	293	+ 04:53	-1								