

**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: Филимонов Н. Н.

Группа: М80-101Б-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 руководителями 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:

29 июня 2022 г.
(дата)

Студент Ф и л и м о н о в Н. Н. _____
(подпись)

ЗАДАНИЕ

Принять участие в учебно-тренировочных конкурсах по олимпиадному программированию для студентов первого курса в течение 9 дней: посетить и проработать установочные лекции, решать и дорешивать конкурсные задания, принять участие в разборах конкурсов. Составить отчет в форме журнала установленной формы и пройти процедуру защиты практики.

Объем практики 108 часов в течение 12 учебных дней.

Руководитель практики от института:

29 июня 2022 г.

(дата)

(подпись)

ТАБЕЛЬ ПРОХОЖДЕНИЯ ПРАКТИКИ

№	Дата	Наименование работы или контеста	Время проведения	Место проведения	Решено задач	Дорешано задач	Подпись
1	29.06.2022	Организационное собрание. Выдача задания	10:30 - 19:30	МАИ			
2	30.06.2022	Основы C++	9:00 - 18:00	Дистанционно	3	2	
3	01.07.2022	Библиотека C++	9:00 - 18:00	Дистанционно	0	2	
4	02.07.2022	Динамическое программирование	9:00 - 18:00	Дистанционно	2	0	
5	04.07.2022	Префиксные суммы, сортировка событий, два указателя	9:00 - 18:00	Дистанционно	0	1	
6	05.07.2022	ДП, задача о рюкзаке	9:00 - 18:00	Дистанционно	0	2	
7	06.07.2022	Длинная арифметика	9:00 - 18:00	Дистанционно	0	0	
8	07.07.2022	Основы теории графов	9:00 - 18:00	Дистанционно	0	2	
9	08.07.2022	Кратчайшие пути во взвешенных графах	9:00 - 18:00	Дистанционно	4	0	
10	09.07.2022	Алгоритмы на строках	9:00 - 18:00	Дистанционно	2	0	
11	11.07.2022	Оформление журнала с электронным приложением	9:00 - 18:00	Дистанционно			
12	12.07.2022	Защита практики	11:00 - 20:00	МАИ			
		Итого часов	108				

Отзывы цеховых руководителей практики

Принято участие в 8 конкурсах, прослушаны установочные лекции и разборы задач, решено 11 и дорешано 9 задач конкурсов, 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: Филимоновым Николаем Николаевичем

Слушали

Постановили

Отчёт практиканта

Считать
практику
выполненной
и
защищённой
на

Общая оценка: _____

Председатель:

Зайцев В. Е. _____

Члены:

Артемьев Д. И. _____

Инютин М. А. _____

Дата: 12 июля 2022 г.

ТЕХНИЧЕСКИЙ ОТЧЁТ ПО ПРАКТИКЕ

Первое соревнование: Основы C++

Фрагмент турнирной таблицы контеста

Standings															
#	Who	=	Penalty	A	B	C	D	E	F	G	H	I	J	K	L
28	Филимонов Николай Николаевич M8O-101B-21	3	373	+4 00:48			-1	+2 02:11		+ 01:14					
	* Филимонов Николай Николаевич M8O-101B-21	2			+4		+								

А. Сумма

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

У Васи очень сложная работа, он складывает числа. Казалось бы, все умеют складывать числа, но задача Васи сложна количеством и длиной чисел которые нужно складывать. В данный момент он умеет складывать по 13 шестизначных чисел в секунду, однако этого недостаточно. Вася очень не хочет терять столь интересную работу, помогите ему, напишите программу, которая будет складывать для него числа быстрее.

Входные данные

В первой строке дано число n ($0 \leq n \leq 10^6$), в следующей строке дано n целых чисел x_i ($|x_i| \leq 10^9$), которые нужно сложить.

Выходные данные

Выведите единственное число — итоговую сумму.

Пример

входные данные	Скопировать
3 1 2 3	
выходные данные	Скопировать
6	

Решение: суммируем всё, убрал задержку ввода. Сложность $O(n)$

Код программы:

```
1. #include <iostream>
2. using namespace std;
3. int main(){
4.     ios::sync_with_stdio(false);
5.     cin.tie(0);
6.     long long a, b, c, i;
7.     b=0;
8.     cin >> c;
9.     for (i=0; i<c; i++){
10.        cin >> a;
11.        b=b+a;
12.    }
13.    cout << b << endl;
14. }
```

Вывод: Задача решена. Была ошибка “превышено ограничение времени на тесте 3”, не учел задержку ввода и вышел за данный временной интервал.

Е. В последний момент

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

Авторы не успели придумать условие этой задачи, поэтому вместо условия здесь дырка от бублика.

Входные данные

Формат ввода описать тоже не успели.

Выходные данные

Как и формат вывода.

Решение: нужно найти количество дырок в вводимом числе. Читаем символ, считаем в нем количество дырок, прибавляем к счетчику. Сложность $O(n)$

Код программы:

```
1. #include <iostream>
2. using namespace std;
3. int main() {
4.     ios::sync_with_stdio(false);
5.     cin.tie(0);
6.     string str;
7.     cin >> str;
8.     long long a, k, l, s;
9.     s=0;
10.    k=a;
11.    for (int i=0; i<str.size(); i++){
12.        if (str[i]=='0' || str[i]=='4' || str[i]=='6' || str[i]=='9'){
13.            s=s+1;
14.        }
15.        if (str[i]=='8'){
16.            s=s+2;
17.        }
18.    }
19.    cout << s << endl;
20. }
```

Выводы: Задача решена.

Второе соревнование: Библиотека C++

Фрагмент турнирной таблицы контеста

Standings															
#	Who	=	Penalty	<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>	<u>E</u>	<u>F</u>	<u>G</u>	<u>H</u>	<u>I</u>	<u>J</u>	<u>K</u>	<u>L</u>
	* Филимонов Николай Николаевич M80-1015-21	2		+1	+										

А. 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 $A, A + 1, A + 2, \dots, A + N - 1$, 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.

Решение: Сортируем массив. Проходим по нему, проверяя, что каждый следующий элемент больше предыдущего на 1. Сложность $O(n)$

Код программы:

```
1. #include <iostream>
2. #include <vector>
3. #include <algorithm>
4. using namespace std;
5.
6. int main() {
7.     ios::sync_with_stdio(false);
8.     cin.tie(0);
9.     int n;
10.    long long x;
11.    cin >> n;
12.    vector<long long> deck;
13.
14.    for (int i = 0; i < n; ++i) {
15.        cin >> x;
16.        deck.push_back(x);
17.    }
18.
19.    sort(deck.begin(), deck.end());
20.
```

```

21.     for (int i = 1; i < n; ++i) {
22.         if (deck[i] != deck[i - 1] + 1) {
23.             cout << "Scammed";
24.             exit(0);
25.         }
26.     }
27.     cout << "Deck looks good";
28. }

```

Выводы: Задача решена.

В. Анаграммы

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

Два слова называются анаграммами, если одно можно получить из другого путём перестановки букв. Например, «кот — ток» и «равновесие — своенравие» являются парами анаграмм, а «кот — кит» не являются анаграммами. У Пети есть большая коллекция различных слов, так как она очень разрослась и её уже тяжело просмотреть всю за один раз, он хочет её как-нибудь уменьшить. Он решил, что меньше всего его коллекция пострадает если он оставит такое подмножество слов, что среди них не будет анаграмм и при этом невозможно будет больше взять ни одного слова из изначальной коллекции так, чтобы предыдущее условие не нарушилось. Помогите ему определить новый размер коллекции, чтобы он мог решить стоит ли всё это затевать или от этого его коллекция уменьшится недостаточно.

Входные данные

В первой строке дано единственное число N ($1 \leq N \leq 2 \cdot 10^5$) — количество слов в коллекции. В следующих N строках даны слова s_i ($1 \leq |s_i| \leq 20$) состоящие из маленьких латинских букв — изначальная коллекция Пети.

Выходные данные

Выведите единственное число — размер новой коллекции, которая останется у Пети, если он будет следовать своему плану.

Решение: создать множество. Читая из ввода по 1 слову, сортируем буквы в нём в лексикографическом порядке и вставляем во множество. Если слово является анаграммой уже внесённого во множество слово, то во множество не вносим.

Сложность $O(n)$

Код программы:

```

1. #include <bits/stdc++.h>
2. using namespace std;
3.
4. bool isBigger(char a, char b) {
5.     return a - 'a' < b - 'a';
6. }
7.
8. string lexGr(string str) {
9.     short length = str.length();
10.    for (int i = 0; i < length; ++i) {
11.        for (int j = 0; j < length - 1; ++j) {
12.            if (!isBigger(str[j], str[j + 1])) {
13.                swap(str[j], str[j + 1]);
14.            }
15.        }
16.    }
17.    return str;
18. }
19.
20.
21. int main() {

```

```

22. ios::sync_with_stdio(false);
23. cin.tie(0);
24.
25. long n, counter = 0;
26. string word;
27. set<string> book;
28.
29. cin >> n;
30. for (int i = 0; i < n; ++i) {
31.     cin >> word;
32.     string newWord = lexGr(word);
33.     if (book.find(newWord) == book.end()) {
34.         book.insert(newWord);
35.         counter++;
36.     }
37. }
38.
39. cout << counter << endl;
40. return 0;
41. }

```

Выводы: Задача решена

Третье соревнование: Динамическое программирование

Фрагмент турнирной таблицы контеста

Standings															
#	Who	=	Penalty	A	B	C	D	E	F	G	H	I	J	K	L
28	Филимонов Николай Николаевич M80-1015-21	2	555	+ 03:58		+1 04:57									

А. 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, ..., 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 ($1 \leq n, k \leq 2 \cdot 10^4$) — пункт назначения и максимальная длина прыжка кузнечика.

Выходные данные

Выведите единственное число — ответ на задачу.

Решение: создаём вектор, в первую ячейку которого ставим число 1, а далее, в каждую последующую ячейку ставим сумму чисел, стоящих в k предыдущих ячейках. Сложность $O(n \cdot k)$

Код программы:

```

1. #include <iostream>
2. #include <vector>
3.
4. using namespace std;
5.
6.

```

```

7. int main() {
8. ios::sync_with_stdio(false);
9. cin.tie(0);
10. int n, k;
11. cin >> n >> k;
12. vector<long long> v(n + 1);
13. v[0] = 1;
14. for (int i = 1; i <= n; i++) {
15. for (int j = 1; j <= min(i, k); j++) {
16. v[i] = (v[i] + v[i - j]) % 1000000007;
17. }
18. }
19. cout << v[n];
20. return 0;
21. }

```

Выводы: задача решена

Четвертое соревнование: Префиксные суммы, сортировка событий, два указателя

Фрагмент турнирной таблицы контеста

Standings															
#	Who	=	Penalty	A	B	C	D	E	F	G	H	I	J	K	L
	* Филимонов Николай Николаевич M80-101B-21	1		+											

А. 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.

Решение:

создаём вектор, в который записываем все числа из ввода. Создаём ещё один вектор, в котором в каждую ячейку $an[i]$ записываем сумму чисел записанных в $v[i-1]$ и $s[i-1]$. В результате в векторе an в каждой ячейке $s[i]$ будет записана сумма чисел от начала вектора v до $v[i-1]$. Когда нужно будет найти сумму чисел на подотрезке вектора, можно будет просто вычесть из числа в $s[r]$, число $an[l-1]$. Где r — индекс правой границы подотрезка, а l — левой. Сложность $O(n)$

Код программы:

```
1. #include <iostream>
```

```

2. #include <vector>
3. using namespace std;
4. int main()
5. {
6.     ios::sync_with_stdio(false);
7.     cin.tie(0);
8.
9.     int n, q, l, r;
10.    cin >> n;
11.    vector<long long> v(n+1);
12.    for (int i = 0; i < n; i++)
13.    {
14.        cin >> v[i];
15.    }
16.    vector<long long> s(n + 1);
17.    s[0] = 0;
18.    for (int i = 1; i < n + 1; i++)
19.    {
20.        s[i] = s[i - 1] + v[i - 1];
21.    }
22.    cin >> q;
23.    for (int i = 0; i < q; i++)
24.    {
25.        cin >> l >> r;
26.        cout << s[r] - s[l-1] << endl;
27.    }
28. }

```

Вывод: задача решена

Пятое соревнование: ДП, задача о рюкзаке

Фрагмент турнирной таблицы контеста

Standings													
#	Who	=	Penalty	A	B	C	D	E	F	G	H	I	J
	* Филимонов Николай Николаевич M8O-1015-21	2		+	+								

А. Старинный шифр

ограничение по времени на тест: 1 секунда

ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

Рассматривая шифровки, полученные от русской пианистки и шведского профессора, Мюллер вдруг вспомнил, как он с двумя стариками, Рафке и Хилли, распутывал в двадцатые годы одно дело. Там тоже была весьма сложная шифровка, которая пересылалась в двух различных экземплярах. Содержимое каждого экземпляра по отдельности не представляло интереса, однако вместе эти шифровки обозначали конкретную информацию о канале, через который связывались заговорщики НСДАП. Чтобы получить ее, нужно было выделить в обеих шифровках некоторую одинаковую последовательность (которая могла разделяться любым количеством любых символов, как внутри одной части шифровки, так и внутри другой) и вычислить ее длину. При этом заговорщики были настолько хитры, что считали только максимальную длину всех возможных последовательностей, совпадающих в шифровках. Тогда Мюллер разгадал шифр. Сейчас годы брали свое — он видел два листа, в которых, очевидно, было зашифровано одно и то же сообщение, но расшифровать его не мог.

Вашей задачей будет не разгадать хитроумный код полковника Исаева, а попытаться расшифровать код заговорщиков НСДАП, так легко раскрытый Мюллером.

Входные данные

В двух строках даны соответственно первая и вторая шифровки, перехваченные у НСДАП. Обе шифровки состоят из латинских букв в верхнем регистре. Длина обеих шифровок не превосходит 500.

Выходные данные

В единственной строке вывода должна содержаться максимальная длина совпадающей подпоследовательности символов в двух шифровках.

Решение: создаём двумерный вектор, размерами $(n+1) \times (m+1)$, где n – длина первой строки, а m – второй. Запускаем 2 цикла, один из которых вложен в другой, с итераторами i и j , идущими от 1 до $n+1$ и $m+1$ соответственно, и заполняем двумерный вектор таким образом, что в ячейку, на которой мы в данный момент находимся, вносится максимум из 2 чисел, стоящих либо в ячейке слева, либо в ячейке сверху. В данном векторе в любой ячейке с индексом $[i][j]$ будут стоять числа, показывающие, какова длина максимальной совпавшей последовательности букв в двух частях слов, где первая часть, это часть первого слова от 1 буквы до i -ой буквы, а вторая часть, это часть второго слова от 1 до j -ой буквы. Сложность $O(n*m)$

Код программы:

```
1. #include <bits/stdc++.h>
2.
3.
4. using namespace std;
5.
6. using pii = pair<int, int>;
7.
8.
9. int main() {
10.     ios::sync_with_stdio(false);
11.     cin.tie(0);
12.
13.     string first, second;
14.     cin >> first >> second;
15.     long long n = first.size(), m = second.size();
16.     vector<vector<long long>> dp(m + 1, vector<long long>(n + 1, 0));
17.
18.     for (int i = 1; i <= m; ++i) {
19.         for (int j = 1; j <= n; ++j) {
20.             if (first[j - 1] == second[i - 1]) {
21.                 dp[i][j] = 1 + dp[i - 1][j - 1];
22.             }
23.             else {
24.                 dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
25.             }
26.         }
27.     }
28.     /*for (int i = 0; i < m + 1; ++i) {
29.         cout << endl;
30.         for (int j = 0; j < n + 1; ++j) {
31.             cout << dp[i][j] << ' ';
32.         }
33.     } */
34.     cout << dp[m][n] << endl;
35.
36.     return 0;
37. }
```

Выводы: Задача решена

Седьмое соревнование: Основы теории графов

Фрагмент турнирной таблицы контеста

Standings											
#	Who	=	Penalty	A	B	C	D	E	F	G	H
	* Филимонов Николай Николаевич M80-1015-21	2		+	+						

А. 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 ($1 \leq k \leq n \leq 100000, 0 \leq m \leq \min(\frac{n(n-1)}{2}, 300000)$) — количество вершин и рёбер в графе и номер вершины, с которой следует начинать обход, соответственно. Далее в m строках описаны рёбра графа в виде пар соединяемых ими вершин.

Выходные данные

Выведите n чисел — номера вершин в порядке обхода в глубину от заданной вершины, если добраться из какой-либо вершины до заданной невозможно, то вместо номера выведите -1.

Решение: строим матрицу смежности, потом вызываем функцию поиска в глубину dfs и запускаем счетчик, который инкрементируется каждый раз, когда функция dfs вызвана в вершине, в которой мы ещё не были. В каждой такой вершине вносим нынешнее значение счетчика. После полного прохода в глубину, выводим массив when. Сложность $O(n+m)$

Код программы:

```
1. #include <bits/stdc++.h>
2.
3. using namespace std;
4.
5. using pii = pair<int, int>;
6. using graph = vector<vector<int>>>;
7.
8. void dfs(int u, const graph & g, vector<bool> & visited, int & counter, vector<int>
  & when) {
9.     if (visited[u]) {
10.         return;
11.     }
12.     when[u] = counter;
13.     counter++;
14.     visited[u] = true;
15.     for (int v : g[u]) {
16.         //cout << v << ' ';
17.         dfs(v, g, visited, counter, when);
18.     }
19. }
20.
21. int main() {
22.     ios::sync_with_stdio(false);
23.     cin.tie(0);
24.
25.     int n, m, k;
26.     cin >> n >> m >> k;
```

```

27.     graph g(n);
28.     for (int i = 0; i < m; ++i) {
29.         int u, v;
30.         cin >> u >> v;
31.         --u;
32.         --v;
33.         g[u].push_back(v);
34.         g[v].push_back(u);
35.     }
36.     vector<bool> visited(n + 1);
37.     vector<int> when(n + 1, -10);
38.     int counter = 0;
39.     dfs(--k, g, visited, counter, when);
40.     for (int i = 0; i < n; ++i) {
41.         if (visited[i]) {
42.             cout << when[i] << ' ';
43.         }
44.         else {
45.             cout << -1 << ' ';
46.         }
47.     }
48.     return 0;
49. }

```

Выводы: Задача решена

В. 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	Сору
3 3 3 1 2 2 3 3 1	
выходные данные	Сору
1 1 0	
входные данные	Сору
3 1 1 1 2	
выходные данные	Сору
0 1 -1	

Решение: строим матрицу смежности, вызываем функцию поиска в ширину, в зависимости от условия выводим либо длину пути, либо -1 при отсутствии пути. Сложность $O(n+m)$

Код программы:

```
1. #include <bits/stdc++.h>
2.
3. using namespace std;
4. using pii = pair<int, int>;
5.
6. const long long int INF = 1e15;
7.
8. void bfs(int start, vector<vector<int>> &g, vector<long long int> &d, vector<int>
    &p)
9. {
10.     d[start] = 0;
11.     queue<int> q;
12.     q.push(start);
13.     while (!q.empty())
14.     {
15.         int u = q.front();
16.         q.pop();
17.         for (int v : g[u])
18.         {
19.             if (d[v] == INF)
20.             {
21.                 d[v] = d[u] + 1;
22.                 p[v] = u;
23.                 q.push(v);
24.             }
25.         }
26.     }
27. }
28.
29. int main()
30. {
31.     ios::sync_with_stdio(false);
32.     cin.tie(0);
33.
34.     int n, m, vertex;
35.     cin >> n >> m >> vertex;
36.     vector<vector<int>> g(n);
37.     vertex--;
38.     for (int i = 0; i < m; i++)
39.     {
40.         int u, v;
41.         cin >> u >> v;
42.         u--;
43.         v--;
44.         g[u].push_back(v);
45.         g[v].push_back(u);
46.     }
47.     int start = vertex - 1;
48.     vector<long long int> d(n, INF);
49.     vector<int> p(n, -1);
50.     bfs(start, g, d, p);
51.     for (int i = 0; i < n; i++)
52.     {
53.         if (d[i] == INF)
54.         {
55.             cout << -1 << " ";
56.         }
57.         else
58.         {
59.             cout << d[i] << " ";
60.         }
61.     }
62.     return 0;
```

Выводы: Задача решена

Восьмое соревнование: Кратчайшие пути во взвешенных графах

Фрагмент турнирной таблицы конкурса

Standings											
#	Who	=	Penalty	A	B	C	D	E	F	G	H
14	Филимонов Николай Николаевич M8O-101B-21	4	908	+ 01:33	+ 03:49	+ 04:52	+ 04:54				

А. 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 ($1 \leq u, v \leq n, |w| \leq 10^5$) — 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 .

Если граф содержит отрицательный цикл, выведите «Negative cycle» без кавычек.

Пример

входные данные		Copy
<pre>5 4 1 1 4 3 1 3 1 3 4 1 5 4 2</pre>		
выходные данные		Copy
<pre>0 -1 1 2 4</pre>		

Решение: Реализуем алгоритм Форда-Беллмана, создавая двумерный массив.
Сложность $O(n \cdot m)$

Код программы:

```
1. #include <bits/stdc++.h>
2.
3. using namespace std;
4.
5. using pii = pair<int, int>;
6. const int64_t INF = 1e18;
7. struct wedge {
8.     int u, v;
9.     int64_t w;
10. };
11.
12.
13. int main()
```

```

14. {
15.     ios::sync_with_stdio(false);
16.     cin.tie(0);
17.
18.     int n, m, start;
19.     cin >> n >> m >> start;
20.
21.     vector<wedge> g;
22.
23.     for (int i = 0; i < m; i++) {
24.         int u, v;
25.         int64_t w;
26.         cin >> u >> v >> w;
27.         --v;
28.         --u;
29.         g.push_back({ u,v,w });
30.         g.push_back({ v,u,w });
31.     }
32.
33.     start--;
34.
35.     vector<int64_t> d(n, INF);
36.     d[start] = 0;
37.     bool flag = false;
38.     for (int i = 0; i < n; i++)
39.     {
40.         flag = false;
41.         for (wedge elem : g) {
42.             int u = elem.u;
43.             int v = elem.v;
44.             int64_t w = elem.w;
45.             if (d[u] + w < d[v]) {
46.                 flag = true;
47.                 d[v] = d[u] + w;
48.             }
49.         }
50.         if (!flag) {
51.             break;
52.         }
53.     }
54.
55.     if (flag) {
56.         cout << "Negative cycle";
57.     }
58.
59.     else {
60.         for (int64_t elem : d)
61.         {
62.             if (elem == INF) cout << "-1" << " ";
63.             else cout << elem << ' ';
64.         }
65.         cout << "\n";
66.     }
67.
68. }

```

Выводы: Задача решена

В. 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** ($1 \leq n \leq 500$) — количество вершин в графе. В следующих **n** строках вам даны по **n** чисел **a_{ij}** ($0 \leq a_{ij} \leq 10^9$, $a_{ii} = 0$) — 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	Copy
1 0	
выходные данные	Copy
0	

входные данные	Copy
3 0 10 1 1 0 1 1 1 0	
выходные данные	Copy
0 2 1 1 0 1 1 1 0	

Решение: Реализуем алгоритм Флойда-Уоршелла, выводим двумерный массив, который представляет таблицу наименьших путей между вершинами графа. Сложность ($n*n$)

Код программы:

```
1. #include <bits/stdc++.h>
2.
3. using namespace std;
4.
5. using pi = pair<int, int>;
6.
7. const int64_t INF = 1e18;
8.
9. int main(){
10.     ios::sync_with_stdio(false);
11.     cin.tie(0);
12.     int n, m;
13.     cin >> n;
14.     vector< vector<int64_t> > d(n+1, vector<int64_t>(n+1, INF));
15.     for (int i = 0; i<n; ++i){
16.         for (int j = 0; j<n; ++j){
17.             int u, v;
18.             int64_t w;
19.             cin >> w;
20.             d[i][j]=w;
```

```

21.     }
22.     }
23.     for (int k = 0; k<n; ++k){
24.         for (int u = 0; u<n; ++u){
25.             for (int v= 0; v<n; ++v){
26.                 d[u][v]=min(d[u][v],d[u][k] + d[k][v]);
27.             }
28.         }
29.     }
30.     for (int u = 0; u<n; ++u){
31.         for (int v = 0; v<n; ++v){
32.             cout << d[u][v] << ' ';
33.         }
34.         cout << endl;
35.     }
36.
37. }

```

Выводы: Задача решена

Контеcт C++ [Кратчайшие пути во взвешенных графах] (08.07.2022)

С. Алгоритм Дейкстры

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 ($1 \leq n \leq 10^4, 1 \leq m \leq 5 \cdot 10^5, 1 \leq s \leq n$) — 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 .

Решение:

По входным данным мы строим список смежности и после запускаем поиск в ширину (BFS) из стартовой вершины. Так как граф взвешенный, расстоянием до вершины будем считать совокупный вес рёбер на пути к ней от стартовой вершины. Находясь в какой-либо вершине, мы проверяем, куда мы можем попасть из неё. Если сопряжённая вершина не посещена, то мы добавляем её в план. А если она уже посещена, мы проверяем, будет ли путь через вершину в которой мы находимся, короче того пути, которым мы добирались до этой сопряжённой вершины ранее. Если это так, то мы просто заменяем значение в счётчике пути для сопряжённой вершины и добавляем её в план, ведь если путь до неё стал короче, то и путь «через» неё тоже. После того, как мы нашли наикратчайшие пути до всех достижимых вершин, мы проверяем достигли ли мы всех из списка победителей и находим расстояние до самого дальнего из них, выводим в зависимости от результата.э
Сложность $O(n^2)$

Код программы:

```
1. #include <bits/stdc++.h>
2.
3. using namespace std;
4.
5. struct wedge {
6.     int u, v;
7.     int64_t w;
8. };
9.
10. using pii = pair<int, int>;
11. using item = pair<int64_t, int>;
12. using wgraph = vector<vector<wedge>>;
13.
14. const int64_t INF = 1e18;
15.
16. vector<int64_t> deikstra(int start, const wgraph & g) {
17.     int n = g.size();
18.     vector<int64_t> d(n, INF);
19.     vector<bool> visited(n);
20.     set<item> s;
```

```

21.     d[start] = 0;
22.     for (int i = 0; i < n; ++i) {
23.         s.insert({ d[i], i });
24.     }
25.     while (!s.empty()) {
26.         int u = s.begin()->second;
27.         s.erase(s.begin());
28.         if (visited[u]) {
29.             continue;
30.         }
31.         visited[u] = true;
32.         for (wedge elem : g[u]) {
33.             int v = elem.v;
34.             int64_t w = elem.w;
35.             if (d[u] + w < d[v]) {
36.                 s.erase({ d[v], v });
37.                 d[v] = d[u] + w;
38.                 s.insert({ d[v], v });
39.             }
40.         }
41.     }
42.     return d;
43. }
44.
45.
46.
47. int main() {
48.     ios::sync_with_stdio(false);
49.     cin.tie(0);
50.     int n, m, start;
51.     cin >> n >> m >> start;
52.     --start;
53.     wgraph g(n);
54.     for (int i = 0; i < m; ++i) {
55.         int64_t w;
56.         int u, v;
57.         cin >> u >> v >> w;
58.         --u;
59.         --v;
60.         g[u].push_back({ u, v, w });
61.         g[v].push_back({ v, u, w });
62.     }
63.
64.     vector<int64_t> d = deikstra(start, g);
65.     for (int64_t el : d) {
66.         if (el == INF) {
67.             cout << -1 << " ";
68.         }
69.         else {
70.             cout << el << " ";
71.         }
72.     }
73.     cout << endl;
74.     return 0;
75. }

```

Выводы: Задача решена

Девятое соревнование: Алгоритмы на строках

Фрагмент турнирной таблицы конкурса

Standings													
#	Who	=	Penalty	A	B	C	D	E	F	G	H	I	J
14	Филимонов Николай Николаевич М80-1015-21	2	351	+2 01:22	+2 03:09								

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.

Решение:

Вводим строку. Посимвольно обрабатываем ее по формуле, записываем полученные значения в m . Выводим раз за разом полученные числа.

Сложность $O(n)$ n – длина строки S

Код программы:

```
1. #include <bits/stdc++.h>
2.
3.
4. using namespace std;
5.
6. int main() {
7.     string s;
8.     cin >> s;
9.     int n = (int) s.length();
10.    vector<int> z (n);
11.    cout << n << " ";
12.    for (int i=1, l=0, r=0; i<n; ++i) {
13.        if (i <= r){
14.            z[i] = min (r-i+1, z[i-l]);
15.        }
16.        while (i+z[i] < n && s[z[i]] == s[i+z[i]])
17.            ++z[i];
18.        if (i+z[i]-1 > r){
19.            l = i, r = i+z[i]-1;
20.        }
21.        cout << z[i] << " ";
22.    }
23.
24. }
25.
```

Выводы: Задача решена