

Министерство науки и высшего образования

Московский авиационный институт
(национальный исследовательский университет)

Институт компьютерных наук и прикладной математики

Кафедра вычислительной математики и программирования

Журнал по ознакомительной практике

Студент: Амурский В. А.

Группа: М8О-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 заносится в раздел «Общественно-политическая практика». Выполнение работы по оказанию практической помощи предприятию (участие в выполнении спецзаданий, работа сверхурочно и т.п.) заносится 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:

« » _____ 2022 г.
(дата)

Студент Амурский В. А. _____
(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. Объем практики 154 часов.

Руководитель практики от института:

« » _____ 2022 г.
(дата)

(подпись)

ТАБЕЛЬ ПРОХОЖДЕНИЯ ПРАКТИКИ

№	Дата	Название контеста	Время проведения	Место проведения	Решено задач	Дорешано задач	Подпись
1	17.09.2021	Основы C++ [1]	16:30 - 21:30	Дистанционно	0	12	
2	19.09.2021	Stage 2-B: Grand Prix of IMO, Div. 2	11:00 - 16:00	Дистанционно	2	0	
3	24.09.2021	Основы C++ [2]	16:30 - 21:30	Дистанционно	0	12	
4	26.09.2021	Stage 3-B: Grand Prix of XiAn, Div. 2	11:00 - 16:00	Дистанционно	2	0	
5	01.10.2021	Библиотека C++ [3]	16:30 - 21:30	Дистанционно	7	5	
6	08.10.2021	Библиотека C++ [4]	16:30 - 21:30	Дистанционно	5	7	
7	15.10.2021	Теория чисел [5]	16:30 - 21:30	Дистанционно	3	6	
8	22.10.2021	Основы ДП [6]	16:30 - 21:30	Дистанционно	6	6	
9	29.10.2021	Арифметика в кольце, комбинаторика, функция Эйлера [7]	16:30 - 21:30	Дистанционно	2	8	
10	05.11.2021	Префиксные суммы, сортировка событий, метод двух указателей [8]	16:30 - 21:30	Дистанционно	4	8	
11	12.11.2021	Двумерное ДП, задача о рюкзаке [9]	16:30 - 21:30	Дистанционно	3	2	
12	19.11.2021	Геометрия, тернарный поиск [10]	16:30 - 21:30	Дистанционно	5	1	
13	05.12.2021	Осенняя олимпиада первого курса	11:00 - 18:30	МАИ	5	0	
14	19.12.2021	ICPC. 1/4 финала	11:00 - 16:00	Дистанционно	3	0	
15	11.02.2022	Основы теории графов [11]	16:30 - 21:30	Дистанционно	3	5	
16	18.02.2022	Кратчайшие пути во взвешенных графах [12]	16:30 - 21:30	Дистанционно	6	2	
17	20.02.2022	Stage 10-B: Grand Prix of Kyoto, Div 2	11:00 - 16:00	Дистанционно	2	0	
18	25.02.2022	СНМ, минимальное остовное дерево [13]	16:30 - 21:30	Дистанционно	3	2	
19	04.03.2022	Деревья, наименьший общий предок [14]	16:30 - 21:30	Дистанционно	2	0	
20	11.03.2022	Паросочетания в двудольном графе, потoki в транспортной сети [15]	16:30 - 21:30	Дистанционно	1	0	
21	18.03.2022	Строки, Z-функция, хеши, префиксное дерево [16]	11:00 - 16:00	Дистанционно	5	0	
22	25.03.2022	ДП по подмножествам, ДП по профилю [17]	16:30 - 21:30	Дистанционно	3	0	
23	01.04.2022	Теория игр, функция Шпрага-Гранди [18]	16:30 - 21:30	Дистанционно	1	0	
24	08.04.2022	Дерево отрезков [19]	16:30 - 21:30	Дистанционно	5	0	
25	15.04.2022	Дерево отрезков с отложенными обновлениями [20]	16:30 - 21:30	Дистанционно	2	0	
26	22.04.2022	Декартово дерево [21]	16:30 - 21:30	Дистанционно	5	0	
27	24.04.2022	Rucode	10:00 - 15:00	Дистанционно	1	0	
28	01.05.2022	Grand Prix of BSUIR, Div 2	11:00 - 16:00	Дистанционно	1	0	
29	15.05.2022	Весенняя олимпиада первого курса	16:30 - 21:30	МАИ	2	0	
30	12.07.2022	Оформление журнала. Защита практики	9:00 - 18:00	МАИ			
		Итого часов	154				

Отзывы цеховых руководителей практики

Принято участие в 29 контестах, прослушаны установочные лекции и разборы задач, дорешаны задачи контестов, оформлен журнал практики. Задание практики выполнено.

Тренер Инютин М. А. _____
(подпись)

Работа в помощь предприятию

Встречи с представителями ИТ-компаний, сотрудничающих с МАИ.

ТЕХНИЧЕСКИЙ ОТЧЁТ ПО ПРАКТИКЕ

Основы C++ [1]

A. A + B

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

Вам заданы два целых числа A и B . Выведите $A + B$.

Входные данные

Единственная строка входных данных содержит два целых числа A и B ($|a|, |b| \leq 100$).

Выходные данные

Выведите $A + B$.

Примеры

входные данные	Скопировать
1 2	
выходные данные	Скопировать
3	

Идея решения

Идея проста: считать два числа, сложить их и вывести. Асимптотика $O(1)$

Исходный код

```
1 | #include <iostream>
2 | #include <string>
3 | #include <unordered_map>
4 | #include <unordered_set>
5 | #include <algorithm>
6 | #include <vector>
7 | #include <cmath>
8 | using namespace std;
9 | typedef long long ll;
10 |
11 | int main()
12 | {
13 |     ios_base::sync_with_stdio(false);
14 |     cin.tie(0);
15 |     cout.tie(0);
16 |     ll a, b;
17 |     cin >> a >> b;
18 |     cout << a + b;
19 |     return 0;
20 | }
```

Фрагмент турнирной таблицы контеста

№	Кто	=	Штраф	А	В	С	D	Е	Е	Г	Н	І	Ј	К	І
	* Амурский Василий Андреевич M80-101B-21	12		+	+1	+	+	+	+1	+2	+	+	+	+1	+

Выводы

Задача дорешана. Проблем не возникло, мог решить на конетсте, если бы его не проспал.

Основы C++ [2]

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	

Идея решения

Сначала заведем массив чисел и введем его через консоль с помощью цикла. Потом так же с помощью цикла просуммируем все числа в массиве и выведем ответ. Асимптотика $O(n)$, где n - кол-во элементов.




Исходный код

```
1 | #include <iostream>
2 | #include <string>
3 | #include <unordered_map>
4 | #include <unordered_set>
5 | #include <algorithm>
6 | #include <vector>
7 | #include <cmath>
8 | #include <iomanip>
9 | using namespace std;
```



```
10 typedef long long ll;
11 int main()
12 {
13     ios_base::sync_with_stdio(false);
14     cin.tie(0);
15     cout.tie(0);
16     int n;
17     cin >> n;
18     ll ans = 0;
19     for (int i = 0; i < n; i++) {
20         ll x;
21         cin >> x;
22         ans += x;
23     }
24     cout << ans;
25     return 0;
26 }
```

Фрагмент турнирной таблицы контеста

Положение 			Совпадений: 1   Амур												
№	Кто	=	Штраф	А	В	С	D	Е	Е	С	Н	І	Ј	К	Љ
	* Амурский Василий Андреевич M8O-101Б-21	12		+1	+	+	+	+1	+1	+	+	+	+2	+4	+1

Выводы

Задача дорешана. Основные события отладки: неправильный ответ на претесте 1, забыл считать кол-во элементов в массиве.

G. Объединение

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

Василий создал онлайн игру в которой игроки могут объединяться в кланы ограниченного размера. Его игра стала довольно популярной и когда кланы стали достигать максимального размера, игроки попросили Василия увеличить максимальный размер кланов. К сожалению Василий не очень хороший программист и код написал так, что теперь невозможно изменить максимальный размер клана. Вместо этого он решил позволить кланам объединяться в союзы, которые не будут иметь ограничения на размер.

После внедрения новой системы игроки начали объединяться в союзы, причём Василий заметил один любопытный момент: объединялись всегда два наименее многочисленных клана. Так как игроки договариваются довольно медленно, Василий решил попробовать предсказать в каком порядке кланы будут объединяться далее. Реализуйте программу, которая определит порядок объединения кланов, если они будут действовать как предполагает Василий.

Входные данные

В первой строке вам дано единственное число N ($1 \leq N \leq 10^5$) — количество кланов в игре. В следующей строке через пробел даны сами размеры кланов a_i ($1 \leq a_i \leq 10^5$).

Выходные данные

В $N - 1$ строке выведите размеры кланов, которые будут объединяться. Сначала выводите размер меньшего клана, затем размер большего клана.

Пример

входные данные	Скопировать
6 1 2 3 4 5 6	
выходные данные	Скопировать
1 2 3 3 4 5 6 6 9 12	

Идея решения



Будем имитировать численность каждого клана в отсортированном состоянии благодаря типу `multiset`. Далее заведем цикл с $n - 1$ итераций, где будем брать два минимальных элемента, выводить их и класть сумму в `multiset`. Асимптотика $O(n \log n)$.

Исходный код

```
1 #include <iostream>
2 #include <string>
3 #include <unordered_map>
4 #include <unordered_set>
5 #include <map>
6 #include <set>
7 #include <algorithm>
8 #include <vector>
9 #include <cmath>
10 #include <iomanip>
11 using namespace std;
12 typedef long long ll;
```

```
13
14
15 int main()
16 {
17     ios_base::sync_with_stdio(false);
18     cin.tie(0);
19     cout.tie(0);
20     ll n;
21     cin >> n;
22     multiset<ll> a;
23     for (int i = 0; i < n; i++) {
24         ll x;
25         cin >> x;
26         a.insert(x);
27     }
28     for (int i = 0; i < n - 1; i++) {
29         ll x1, x2;
30         x1 = *a.begin();
31         a.erase(a.begin());
32         x2 = *a.begin();
33         a.erase(a.begin());
34         cout << x1 << ' ' << x2 << endl;
35         a.insert(x1 + x2);
36     }
37     return 0;
38 }
```

Фрагмент турнирной таблицы контеста

Положение 											Совпадений: 2 					<input type="text" value="Аму"/>	
№	Кто	=	Штраф	А	В	С	Д	Е	Е	Г	Н	І	І	К	Л		
13	Амурский Василий Андреевич М80-1015-21	5	498	+1 00:07	+		+1 01:43	+	-1 01:32	+	02:00						
	* Амурский Василий Андреевич М80-1015-21	8				+	+		+4		+1	+2	+	+	+2		

Выводы

Задача решена, проблем не возникло.

Арифметика в кольце, комбинаторика, функция Эйлера [7]

А. Степень

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

Вычислите результат возведения a в степень n по модулю $10^9 + 7$.

Входные данные

В первой строке вам задано единственное число T ($1 \leq T \leq 10^5$) — количество тестов. В следующих T строках вам заданы тесты в виде пар целых чисел a и n ($0 \leq a \leq 10^9, 0 \leq n \leq 10^{18}; a + n > 0$).

Выходные данные

Для каждого теста в отдельной строке выведите результат вычисления a^n по модулю $10^9 + 7$.

Пример

входные данные	Скопировать
3 2 1 2 2 2 10	
выходные данные	Скопировать
2 4 1024	

Идея решения

Реализуем логарифмическое возведение в степень: от степени n мы переходим, если она чётна, к $n/2$, а иначе — к $n - 1$. Понятно, что всего будет не более $2 \log n$ переходов, прежде чем мы придём к $n = 0$. К тому же, после каждого перемножения будем брать остаток по $m = 10^9 + 7$. Асимптотика $O(\log n)$

Исходный код

```
1 | #include <iostream>
2 | #include <string>
3 | #include <unordered_map>
4 | #include <unordered_set>
5 | #include <map>
6 | #include <set>
7 | #include <algorithm>
8 | #include <vector>
9 | #include <cmath>
10 | #include <numeric>
11 | #include <iomanip>
12 | #include <stack>
13 | #include <fstream>
14 | using namespace std;
15 | typedef long long ll;
16 | ll mod = 1000000007;
17 | int main()
18 | {
19 |     ios_base::sync_with_stdio(false);
```

```
20 cin.tie(0);
21 cout.tie(0);
22 ll t;
23 cin >> t;
24 for (int _ = 0; _ < t; _++) {
25     ll res = 1;
26     ll a, n;
27     cin >> a >> n;
28     while (n > 0) {
29         if (n % 2 == 1) {
30             res = (res * a) % mod;
31         }
32         a = (a * a) % mod;
33         n /= 2;
34     }
35     cout << res << endl;
36 }
37 return 0;
38 }
```

Фрагмент турнирной таблицы контеста

Положение										Совпадений: 2				амур	
№	Кто	=	Штраф	А	В	С	Д	Е	Е	С	Н	І	І		
13	Амурский Василий Андреевич М80-1015-21	2	15	<div><div></div><div>+</div><div>00:02</div></div>	<div><div></div><div>+</div><div>00:13</div></div>	-2	-1								
	* Амурский Василий Андреевич М80-1015-21	8				+3	+	+3	+	+3	+1	+1	+		

Выводы

Задача решена, проблем не возникло.

Префиксные суммы, сортировка событий, метод двух указателей [8]

В. Отрезки — 1

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

На числовую прямую накладываются отрезки, сколько точек числовой прямой будет накрыто хотя бы одним отрезком.

Входные данные

В первой строке вам дано число N ($1 \leq N \leq 2 \cdot 10^5$) — количество отрезков, уложенных на прямую. В следующих N строках заданы сами отрезки в виде пар чисел разделённых пробелом l_i и r_i ($|l_i|, |r_i| \leq 10^9, l_i \leq r_i$) (отрезок накрывает все точки с l_i по r_i включительно).

Выходные данные

Выведите единственное число — количество точек числовой прямой накрытых хотя бы одним отрезком.

Примеры

входные данные	Скопировать
3 -2 2 -1 1 0 1	
выходные данные	Скопировать
5	

Идея решения

Реализуем метод сканирующей прямой: создадим массив, в котором каждом элементе будет храниться два числа - первое число координата точки, второе число приоритет точки (0 - открывающая точка, 1 - закрывающая точка). К тому же будем хранить две переменные ans и q , которые будут отвечать за кол-во точек не накрытых хотя бы одним отрезком и кол-во "открытых" отрезков. Далее будем обрабатывать сами отрезки: если текущая точка открывающая, то добавим единицу к переменной q и проверим равенство $q = 1$, если равенство справедливо, то добавим к переменной ans разницу между текущей точкой и предыдущей, если точка закрывающая, то вычтем из q единицу. Ответ равен $last - first - ans + 1$, где $last$ - координата крайней правой точки, $first$ - координата крайней левой точки. Асимптотика $O(n \log n)$.

Исходный код

```
1 #include <iostream>
2 #include <cmath>
3 #include <algorithm>
4 #include <vector>
5 #include <iomanip>
6 #include <unordered_map>
7 #include <unordered_set>
8 #include <map>
9 #include <set>
10 #include <string>
11 #include <stack>
12 #include <deque>
13
14 using namespace std;
```

```

15 typedef long long ll;
16
17 int main() {
18     ios_base::sync_with_stdio(false);
19     cin.tie(0);
20     cout.tie(0);
21     ll n;
22     cin >> n;
23     vector<pair<ll, ll>> a(2*n);
24     for (int i = 0; i < 2*n; i+=2) {
25         ll l, r;
26         cin >> l >> r;
27         a[i] = { l, 0 };
28         a[i + 1] = { r, 1 };
29     }
30     sort(a.begin(), a.end());
31     ll ans = 0;
32     ll q = 0;
33     for (int i = 0; i < 2 * n; i++) {
34         if (a[i].second == 0) {
35             ++q;
36             if (q == 1 and i) {
37                 ans += a[i].first - a[i - 1].first - 1;
38             }
39         }
40         else {
41             --q;
42         }
43     }
44     cout << a[2 * n - 1].first - a[0].first - ans + 1;
45     return 0;
46 }

```

Фрагмент турнирной таблицы контеста

№	Кто	=	Штраф	A	B	C	D	E	F	G	H	I	J	K	L
10	Амурский Василий Андреевич M8O-101B-21	4	127	+1 00:05	+ 00:18	+ 00:24	+ 01:00	-3							
	* Амурский Василий Андреевич M8O-101B-21	8						+8	+12	+5	+	+2	+2	+2	+

Выводы

Задача решена, проблем не возникло.

Двумерное ДП, задача о рюкзаке [9]

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 предоставляло 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.

Пример

входные данные	Скопировать
ABCBDAВ BDCABA	
выходные данные	Скопировать
4	

Идея решения

Переборное решение будет работать за $O(n^2 * m^2)$, что при заданном условии не будет проходить ограничение по времени. При таком ограничении по времени пройдет только $O(n * m)$. Реализуем задачу о наибольшей общей возрастающей последовательности. $d[i][j]$ — это длина наибольшей общей возрастающей подпоследовательности префиксов $a[1..i]$ и $b[1..j]$, причем элемент $b[j]$ — последний представитель НОВП массива b , а $a[i]$ может не быть последним в массиве a . Вычислять d будем всё так же: сначала по увеличению i , а при равенстве — по увеличению j . Тогда для очередного значения $d[i][j]$ есть два варианта:

- $a[i]$ не входит в НОВП. Тогда $d[i][j] = d[i-1][j]$: значение динамики уже посчитано на префиксе $a[1..i-1]$
- $a[i]$ входит в НОВП. Это значит, что $a[i] = b[j]$, то есть для подсчёта $d[i][j]$ нужно пробежать циклом по b в поисках элемента $b[k] < b[j]$ с наибольшим значением $d[i-1][k]$. Но мы считаем d сначала по увеличению i , поэтому будем считать $a[i]$ фиксированным. Чтобы не запускать цикл при каждом равенстве $a[i]$ элементу $b[k]$, в дополнительной переменной $best$ будем хранить "лучший" элемент (и его индекс ind в массиве b) такой, что этот элемент строго меньше $a[i]$ (а

также меньше $b[k]$) и значение динамики для него максимально: $b[ind] < a[i] = b[j]$ и $best = d[i - 1][ind] - > max$

Исходный код

```

1 #include <iostream>
2 #include <cmath>
3 #include <algorithm>
4 #include <vector>
5 #include <iomanip>
6 #include <unordered_map>
7 #include <unordered_set>
8 #include <map>
9 #include <set>
10 #include <string>
11 #include <stack>
12 #include <deque>
13
14 using namespace std;
15 typedef long long ll;
16
17 int main() {
18     ios_base::sync_with_stdio(false);
19     cin.tie(0);
20     cout.tie(0);
21     string a, b;
22     cin >> a >> b;
23     ll n = a.size(), m = b.size();
24     vector<vector<ll>> dp(n + 1, vector<ll>(m + 1));
25     for (int i = 1; i <= n; i++) {
26         for (int j = 1; j <= m; j++) {
27             if (a[i - 1] == b[j - 1]) {
28                 dp[i][j] = dp[i - 1][j - 1] + 1;
29             }
30             else {
31                 dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
32             }
33         }
34     }
35     cout << dp[n][m];
36     return 0;
37 }
```

Фрагмент турнирной таблицы контеста

№	Кто	=	Штраф	А	В	С	Д	Е	Е	Г	Н	І	І
9	Амурский Василий Андреевич М80-101Б-21	3	221	⁺ 00:06	⁺ 01:36	⁺ 01:59				-5			
	* Амурский Василий Андреевич М80-101Б-21	2					+		+1				

Выводы

Задача решена, проблем не возникло.

Геометрия, тернарный поиск [10]

В. Угол между векторами

ограничение по времени на тест: 1 секунда

ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

Определите наименьший неориентированный угол между парой ненулевых векторов.

Входные данные

В первой строке вам дано число N ($1 \leq N \leq 1000$) — количество тестов в файле. Далее каждый тест задаётся в отдельной строке в виде пар координат трёх точек $P_0 = (x_0, y_0)$, $P_1 = (x_1, y_1)$ и $P_2 = (x_2, y_2)$ ($0 \leq |x_i|, |y_i| \leq 10^4$), разделённых пробелом.

Выходные данные

Для каждого теста в отдельной строке выведите число из отрезка $[0, \pi]$ — угол между векторами (P_0, P_1) и (P_0, P_2) . Абсолютная либо относительная погрешность результатов не должна превышать 10^{-6} .

Пример

входные данные	Скопировать
2 0 0 1 0 0 1 1 1 0 0 2 2	
выходные данные	Скопировать
1.570796 3.141593	

Идея решения

Реализуем векторы и операции над ними. Угол между двумя векторами находится с помощью функции взятия арктангенса между векторным произведением и скалярным произведением, но здесь угол имеет диапазон от $[-2\pi; 2\pi]$, а в условии сказано найти от $[0; \pi]$, поэтому возьмем модуль от полученного угла. Асимптотика $O(1)$.

Исходный код

```
1 | #include <iostream>
2 | #include <sstream>
3 | #include <fstream>
4 | #include <iomanip>
5 | #include <string>
6 | #include <cstdlib>
7 | #include <cstdio>
8 | #include <cstring>
9 | #include <cmath>
10 | #include <ctime>
11 | #include <climits>
12 | #include <cassert>
13 | #include <vector>
14 | #include <queue>
15 | #include <stack>
16 | #include <deque>
17 | #include <set>
18 | #include <map>
19 | #include <bitset>
```

```

20 #include <utility>
21 #include <algorithm>
22 #include <unordered_map>
23 using namespace std;
24
25 const double eps = 10e-3;
26 struct point {
27     double x, y;
28     point() {}
29     point(double a, double b) : x(a), y(b) {}
30 };
31
32 point operator -(const point& a, const point& b) {
33     return point(a.x - b.x, a.y - b.y);
34 }
35
36 double scal(point a, point b) {
37     return a.x * b.x + a.y * b.y;
38 }
39
40 double vec(point a, point b) {
41     return a.x * b.y - b.x * a.y;
42 }
43
44 double ug(point a, point b) {
45     return (atan2(vec(a, b), scal(a, b)));
46 }
47
48 double len(const point a, const point b) {
49     return (sqrt((b.x - a.x) * (b.x - a.x) + (b.y - a.y) * (b.y - a.y)));
50 }
51
52 istream& operator >>(istream& in, point& p) {
53     in >> p.x >> p.y;
54     return in;
55 }
56
57 bool eql(double a, double b) {
58     if (abs(a - b) < eps) {
59         return true;
60     }
61     return false;
62 }
63
64 int main() {
65     ios_base::sync_with_stdio(false);
66     cin.tie(0);
67     cout.tie(0);
68     int n;
69     cin >> n;
70     for (int i = 0; i < n; i++) {
71         point a, b;
72         int x0, y0, x1, y1, x2, y2;
73         cin >> x0 >> y0 >> x1 >> y1 >> x2 >> y2;
74         a = point(x1 - x0, y1 - y0);
75         b = point(x2 - x0, y2 - y0);
76         cout << fixed << setprecision(6) << abs(ug(a, b)) << '\n';
77     }
78     return 0;
79 }

```

Фрагмент турнирной таблицы контеста

№	Кто	=	Штраф	А	В	С	Д	Е	Е	Г	Н	І	Ј
6	Амурский Василий Андреевич М80-101Б-21	5	79	+ 00:08	+ 00:11	+ 00:12	+ 00:20	+ 00:28	-1				
	* Амурский Василий Андреевич М80-101Б-21	1							-2	+			

Выводы

Задача решена, проблем не возникло.

Основы теории графов [11]

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 ($1 \leq k \leq n \leq 100000$, $0 \leq m \leq \min\left(\frac{n(n-1)}{2}, 300000\right)$) — 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 ($1 \leq a, b \leq n$)

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	

Идея решения

Поиском в глубину называется рекурсивный алгоритм обхода дерева или графа, начинающий в корневой вершине (в случае графа её может быть выбрана произвольная вершина) и рекурсивно обходящий весь граф, посещая каждую вершину ровно один раз. Для того чтобы вывести вершины в порядке обхода в глубину заведем переменную счетчик s и массив ans , где будем отмечать под каким значением счетчика мы вошли в вершину. В конце выведем массив ans .

Исходный код

```
1 || #include <iostream>
2 || #include <string>
```

```

3 #include <unordered_map>
4 #include <unordered_set>
5 #include <map>
6 #include <set>
7 #include <algorithm>
8 #include <vector>
9 #include <cmath>
10 #include <numeric>
11 #include <iomanip>
12 #include <stack>
13
14 using namespace std;
15 typedef long long ll;
16
17 ll mod = 1000000007;
18 ll inf = INT64_MAX;
19
20 vector<vector<ll>> gr;
21 vector<bool> used;
22 vector<ll> ans;
23 ll c = 0;
24 void dfs( ll p) {
25     used[p] = 1;
26     ans[p] = c++;
27     for (int i = 0; i < gr[p].size();i++) {
28         if (!used[gr[p][i]]) {
29             dfs(gr[p][i]);
30         }
31     }
32 }
33 int main() {
34     ios_base::sync_with_stdio(false);
35     cin.tie(0);
36     cout.tie(0);
37     ll n, m, k;
38     cin >> n >> m >> k;
39     gr.resize(n);
40     used.resize(n);
41     ans.assign(n, -1);
42     for (int i = 0; i < m; i++) {
43         ll a, b;
44         cin >> a >> b;
45         gr[a-1].push_back(b-1);
46         gr[b - 1].push_back(a - 1);
47     }
48     dfs(k - 1);
49     for (auto t : ans) {
50         cout << t << ' ';
51     }
52     return 0;
53 }

```

Фрагмент турнирной таблицы конкурса

№	Кто	=	Штраф	А	В	С	Д	Е	Е	Г	Н
11	Амурский Василий Андреевич М80-101Б-21	3	120	+ 00:05	+ 00:18	-2		+ 01:37			
	* Амурский Василий Андреевич М80-101Б-21	5				+4	+		+3	+2	+6

Выводы

Задача решена, проблем не возникло.

Stage 10-B: Grand Prix of Kyoto, Div 2

Problem A. Announcements

Input file: *standard input*
Output file: *standard output*
Time limit: 2 seconds
Memory limit: 1024 mebibytes

There are N billboards with announcements near Kyoto University.

The i -th billboard appears at day S_i . However, at each T -th day, all billboards installed before this day are removed. You may assume that, on those days, no new billboards will appear.

Find the minimal number of times you need to visit the university to see each billboard at least once.

Input

The first line of input contains one integer N ($1 \leq N \leq 2 \cdot 10^5$). The second line contains N integers S_1, S_2, \dots, S_N . Here, S_i is the day when the i -th billboard appears ($1 \leq S_i \leq 10^9$). The last line contains one integer T ($2 \leq T \leq 10^9$, S_i is not divisible by T for any i): the interval between successive deletions. This means the billboards are removed on days $T, 2T, 3T$, and so on.

Output

Print one integer: the minimum number of visits you need to do to see each billboard at least once.

Examples

standard input
3 1 2 5 3
standard output
2

Идея решения

Пусть одна итерация - это момент от предыдущего удаления листовок до следующего. Тогда мы можем узнать в какую итерацию была удалена каждая из листовок, поделив день вывешивания листовки на период их удаления. Нам надо узнать кол-во различных итераций. Это легко сделать, если мы результат деления положим в объект типа `set` и узнаем размер полученного множества.

Исходный код


```
1 | #include <iostream>
2 | #include <cmath>
3 | #include <vector>
4 | #include <algorithm>
5 | #include <set>
6 | #include <queue>
7 | #include <tuple>
8 | #include <string>
9 | #include <list>
10 | typedef long long ll;
11 | using namespace std;
```

```

12 | const ll inf = INT64_MAX;
13 |
14 | ll n, t;
15 | vector<ll> a;
16 |
17 | int main() {
18 |     ios_base::sync_with_stdio(false);
19 |     cin.tie(0);
20 |     cout.tie(0);
21 |     cin >> n;
22 |     a.resize(n);
23 |     for (int i = 0; i < n; i++) cin >> a[i];
24 |     cin >> t;
25 |     set<ll> ans;
26 |     for (int i = 0; i < n; i++) {
27 |         ans.insert(a[i] / t);
28 |     }
29 |     cout << ans.size();
30 |     return 0;
31 | }

```

Фрагмент турнирной таблицы контеста

#	Participant  Y	A	B	C	D	E	F	G	H	K	N	O	P	Q	Score	Penalty
		52/79	17/34	3/13	5/25	5/14	0/3	2/11	0/0	0/0	52/78	28/81	43/99	16/49		
47	MAI #33 Amursky, Lyugge, Chesnov :	+ 00:05	-2 02:10	—	—	—	—	—	—	—	+2 02:17	-3 02:28	-1 03:15	—	2	183

Выводы

Задача решена, проблем не возникло.

Деревья, наименьший общий предок [14]

А. Запросы о предках

ограничение по времени на тест: 1 секунда
ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод
вывод: стандартный вывод

Вам дано дерево из n вершин.

Требуется ответить на q запросов: «Является ли вершина u предком v ?»

Входные данные

Первая строка содержит одно целое число t ($1 \leq t \leq 10^4$) — количество наборов входных данных.

Первая строка каждого набора содержит одно целое число n ($2 \leq n \leq 2 \cdot 10^5$) — количество вершин в дереве.

Следующие $n - 1$ строка содержат по одному целому числу x . Число x на строке i означает, что x — предок вершины i ($1 \leq x \leq i \leq n$).

Следующая строка содержит одно целое число q ($1 \leq q \leq 2 \cdot 10^5$) — число запросов в наборе.

Следующие q строк содержат по два целых числа u, v ($1 \leq u, v \leq n, u \neq v$) — вершины дерева, характеризующие запрос.

Суммы n и q по всем наборам входных данных не превосходит $2 \cdot 10^5$.

Выходные данные

Для каждого запроса в наборе входных данных выведите «YES» (без кавычек), если вершина u является предком v .

Выведите «NO» (без кавычек) в противном случае.

Вы можете выводить YES и NO в любом регистре (например, строки yEs, yes, Yes и YES будут распознаны как положительный ответ).

Идея решения

Введем массивы tin и $tout$ для подсчёта времени входа и выхода соответственно. Вершина u является предком v тогда и только тогда, когда $tin_v \in [tin_u; tout_u]$. Это условие мы и проверим и выведем ответ.

Исходный код

```
1 | #include <iostream>
2 | #include <sstream>
3 | #include <fstream>
4 | #include <iomanip>
5 | #include <string>
6 | #include <cstdlib>
7 | #include <cstdio>
8 | #include <cstring>
9 | #include <cmath>
10 | #include <vector>
11 | #include <queue>
12 | #include <algorithm>
13 | using namespace std;
14 | typedef int ll;
15 | const long long size = 10e5 * 2 + 228;
16 | ll n, m;
```



```

17 ll qq = 0;
18 ll q, v, u;
19 ll x;
20 int tin[202222], tout[202222];
21 vector<vector<ll>> gr;
22 void dfs(long long v, long long p = -1) {
23     tin[v] = qq++;
24     for (int i = 0; i < gr[v].size(); i++) {
25         dfs(gr[v][i]);
26     }
27     tout[v] = qq;
28     return;
29 }
30 int main() {
31     ios_base::sync_with_stdio(false);
32     cin.tie(0);
33     cout.tie(0);
34     ll t;
35     cin >> t;
36     for (int _ = 0; _ < t; _++) {
37         cin >> n;
38         gr.assign(n, vector<ll>());
39         /* tin.assign(n, 0), tout.assign(n, 0); */
40         for (int i = 1; i <= n - 1; i++) {
41             cin >> x;
42             gr[x - 1].push_back(i);
43         }
44         dfs(0);
45         cin >> q;
46         for (int i = 0; i < q; i++) {
47             cin >> u >> v;
48             --u, --v;
49             if (tin[v] >= tin[u] and tin[v] < tout[u]) {
50                 cout << "YES";
51             }
52             else cout << "NO";
53             cout << '\n';
54         }
55     }
56     return 0;
57 }
58 }

```

Фрагмент турнирной таблицы конкурса

№	Кто	=	Штраф	A	B	C	D	E	F
18	Амурский Василий Андреевич М80-101Б-21	2	435	+9 01:48	+3 01:27				

Выводы

Задача решена. Основные события процесса отладки: ошибка исполнения на тесте 2, вместо resize использовал assign; превышено ограничение времени на тесте 10, добавил команды cin.tie(0) и cout.tie(0);