

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: Постнов А. В.

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:

«17» сентября 2021 г.

Студент Постнов А. В. _____
(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 108 часов.

Руководитель практики от института:

« » _____ 2021 г.
(дата)

(подпись)

ТАБЕЛЬ ПРОХОЖДЕНИЯ ПРАКТИКИ

№	Дата	Название контекста	Время проведения	Место проведения	Решено задач	Дорешано задач	Подпись
1	17.09.2021	Выдача задания, Основы C++ [1]	16:30 - 21:30	МАИ	9	3	
2	24.09.2021	Основы C++ [2]	16:30 - 21:30	Дистанционно	7	5	
3	01.10.2021	Библиотека C++ [3]	16:30 - 21:30	Дистанционно	6	6	
4	08.10.2021	Библиотека C++ [4]	16:30 - 21:30	Дистанционно	5	7	
5	15.10.2021	Теория чисел [5]	16:30 - 21:30	Дистанционно	3	6	
6	22.10.2021	Основы ДП [6]	16:30 - 21:30	Дистанционно	4	8	
7	29.10.2021	Арифметика в кольце, комбинаторика, функция Эйлера [7]	16:30 - 21:30	Дистанционно	3	7	
8	05.11.2021	Префиксные суммы, сортировка событий, метод двух указателей [8]	16:30 - 21:30	Дистанционно	5	7	
9	12.11.2021	Двумерное ДП, задача о рюкзаке [9]	16:30 - 21:30	Дистанционно	2	5	
10	19.11.2021	Геометрия, тернарный поиск [10]	16:30 - 21:30	Дистанционно	5	5	
11	05.12.2021	Осенняя олимпиада первого курса	11:00 - 17:30	МАИ	3	2	
12	09.02.2022	Тренировочный конкурс на разные темы	15:00 - 18:00	Дистанционно	5	0	
13	11.02.2022	Основы теории графов [11]	16:30 - 21:30	Дистанционно	3	5	
14	18.02.2022	Кратчайшие пути во взвешенных графах [12]	16:30 - 21:30	Дистанционно	6	2	
15	25.02.2022	СНМ, минимальное остовное дерево [13]	16:30 - 21:30	Дистанционно	0	7	
16	04.03.2022	Деревья, наименьший общий предок [14]	16:30 - 21:30	Дистанционно	2	2	
17	11.03.2022	Паросочетания в двудольном графе, потоки в транспортной сети [15]	16:30 - 21:30	Дистанционно	2	4	
18	18.03.2022	Строки, Z-функция, хеши, префиксное дерево [16]	16:30 - 21:30	Дистанционно	5	2	
19	25.03.2022	ДП по подмножествам, ДП по профилю [17]	16:30 - 21:30	Дистанционно	3	4	
20	01.04.2022	Теория игр, функция Шпрага-Гранди [18]	16:30 - 21:30	Дистанционно	1	8	
21	08.04.2022	Дерево отрезков [19]	16:30 - 21:30	Дистанционно	3	4	
22	22.04.2022	Декартово дерево [21]	16:30 - 21:30	Дистанционно	3	0	
23	15.05.2022	Весенняя олимпиада первого курса	11:00 - 17:30	МАИ	1	3	
24	12.07.2022	Оформление журнала. Защита практики	09:00 - 18:00	МАИ			
		Итого часов	125		86	102	

Отзывы цеховых руководителей практики

Принято участие в 23 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]

I. Треугольник

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

Вам задан невырожденный треугольник координатами своих вершин. Выведите его площадь.

Входные данные

В трёх строках вам даны координаты трёх углов треугольника в виде пар чисел разделённых пробелом. Координаты по абсолютной величине не превышают 10^3 .

Выходные данные

Выведите единственное число — площадь треугольника с абсолютной или относительной погрешностью 10^{-9} .

Примеры

входные данные	Скопировать
0 0 1 0 0 1	
выходные данные	Скопировать
0.5	

Идея решения

Для того, чтобы определить площадь треугольника буду использовать формулу Герона:

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

где p - полупериметр треугольника, а a , b , c - длины его сторон. Длину сторон вычислю по координатам вершин.

Исходный код

```
1 #include <iostream>
2 #include <iomanip>
3 #include <cmath>
4 //2a1 + d(n - 1) / 2 * n
5
6 using namespace std;
7 int main(){
8     double x1, y1, x2, y2, x3, y3;
9     double d1, d2, d3;
10    cin >> x1 >> y1;
11    cin >> x2 >> y2;
12    cin >> x3 >> y3;
13    d1 = sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
14    d2 = sqrt((x3 - x1) * (x3 - x1) + (y3 - y1) * (y3 - y1));
15    d3 = sqrt((x3 - x2) * (x3 - x2) + (y3 - y2) * (y3 - y2));
16    double p = (d1 + d2 + d3) / 2;
17    cout << setprecision(10);
18    cout << sqrt(p * (p - d1) * (p - d2) * (p - d3));
```

```
19 ||
20 ||
21 ||     return 0;
22 || }
```

Фрагмент турнирной таблицы контеста

Результаты

Вы можете использовать двойной щелчок (или Ctrl + щелчок) по ячейкам таблицы для просмотра истории

Положение		Совпадений: 1												Постнов	
№	Кто	=	Штраф	A	B	C	D	E	F	G	H	I	J	K	L
30	Постнов Александр Вячеславович М8О-101Б-21	9	591	+ 00:03	+2 00:12	+ 00:16	+ 00:20	+ 00:36	+3 00:57	+3 01:43	+1 01:17	+ 01:27			

Выводы

Задача решена. Отладка не потребовалась.

Основы C++ [2]

L. Снова сумма

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

Задание очень простое, вам нужно сложить все данные числа.

Входные данные

В каждой строке входного файла вам дано одно число — x_i ($0 \leq x_i \leq 10^{15}$). Каждое из чисел x_i дано с точностью ровно 15 знаков после запятой. Количество строк не превышает 100.

Выходные данные

Выведите результат сложения с точностью **ровно 15** знаков после запятой, лидирующие нули у чисел выводить запрещено.

Примеры

входные данные	Скопировать
1.000000000000000	
выходные данные	Скопировать
1.000000000000000	

входные данные	Скопировать
1.000000000000000	
2.000000000000000	
выходные данные	Скопировать
3.000000000000000	

Идея решения

В переменную типа *double* не поместится такое длинное число, поэтому буду считать отдельно целую часть и вещественную часть. Ответом будет конкатенация этих строк.

Исходный код

```
1 | #include <iostream>
2 | #include <string>
3 |
4 | using namespace std;
5 |
6 | int main(){
7 |     ios::sync_with_stdio(false);
8 |     cin.tie(0);
9 |     string s;
10 |
11 |     long long chislo1 = 0;
12 |     long long chislo2 = 0;
13 |     string answer;
14 |     int counter = 0;
15 |     while (cin >> s){
16 |         counter++;
17 |         string num1 = "", num2="";
18 |         bool flag = false;
19 |         for (auto i : s){
```




```

20         if (i != '.' && !flag){
21             num1 += i;
22         }
23         else if (!flag) {
24             flag = true;
25             continue;
26         }
27         if (flag){
28             num2 += i;
29         }
30     }
31     long long n1 = stoll(num1);
32     long long n2 = stoll(num2);
33     chislo1 += n1;
34     chislo2 += 1e15;
35     chislo2 += n2;
36     long long k = chislo2 / 1e15;
37     if (k > 1){
38         chislo2 -= 1e15 * (k - 1);
39         chislo1 += (k - 1);
40     }
41     string temp = to_string(chislo2).substr(1, to_string(chislo2).size() - 1);
42     for (int i = temp.size(); i < 15; i++){
43         temp += "0";
44     }
45     answer = to_string(chislo1 - (counter - 1)) + "." + temp;
46 }
47 cout << answer << "\n";
48 }

```

Фрагмент турнирной таблицы конкурса

Вы можете использовать двойной щелчок (или Ctrl + щелчок) по ячейкам таблицы для просмотра истории

Положение 												Совпадений: 1   Постнов				
№	Кто	=	Штраф	A	B	C	D	E	F	G	H	I	J	K	L	
	* Постнов Александр Вячеславович М80-101Б-21	5									+	+	+3	+	+3	

Вывод

Задача дорешана. Эта задача показывает, что нужно внимательно смотреть на входные данные.

Библиотека C++ [3]

G. Ошибка в уставе

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

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

Входные данные

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

Выходные данные

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

Примеры

входные данные	Скопировать
2 ()	
выходные данные	Скопировать
Ja	

Идея решения

Классическая задача на использование дека. Пушим открыв. скобки. Если на вход закрыв. скобка, попадаем элемент из дека, проверяем на одинаковость типа открыв. скобки. $O(n)$

Исходный код

```
1 | #include <iostream>
2 | #include <algorithm>
3 | #include <vector>
4 | #include <string>
5 | #include <iomanip>
6 |
7 |
8 | using namespace std;
9 |
10 | long long flag(long long a, long long b, long long t){
11 |     return t / a + t / b;
12 | }
13 |
14 |
15 |
16 | int main(){
17 |     ios::sync_with_stdio(false);
18 |     cin.tie(0);
19 |     int n;
20 |     cin >> n;
21 |     vector <char> deck;
```

```
22 |     bool flag = true;
23 |     for (int i = 0; i < n; i++){
24 |         char temp;
25 |         cin >> temp;
26 |         if (temp == '(' || temp == '{' || temp == '['){
27 |             deck.push_back(temp);
28 |         }
29 |         else{
30 |             if (deck.size() != 0){
31 |                 auto f = deck.back();
32 |                 deck.pop_back();
33 |                 if (f == '[' && temp == ']'){
34 |                     continue;
35 |                 }
36 |                 else if (f == '{' && temp == '}'){
37 |                     continue;
38 |                 }
39 |                 else if (f == '(' && temp == ')'){
40 |                     continue;
41 |                 }
42 |             }
43 |             flag = false;
44 |             break;
45 |         }
46 |     }
47 |     if (flag && deck.size() == 0){
48 |         cout << "Ja";
49 |     }
50 |     else {
51 |         cout << "Nein";
52 |     }
53 | }
```

Фрагмент турнирной таблицы контеста

Положение															
Совпадений: 2 постнов															
№	Кто	=	Штраф	A	B	C	D	E	F	G	H	I	J	K	L
16	Постнов Александр Вячеславович М80-101Б-21	6	456	+ 00:02	+ 00:13	+2 00:36		+2 01:14		+ 01:59		+2 01:32			
	* Постнов Александр Вячеславович М80-101Б-21	6					+12		+7		+		+	+3	+

Вывод

Задача решена.

Е. Постфиксная запись

ограничение по времени на тест: 0.25 секунд

ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

Однажды в очереди в столовую Вася услышал разговор старшекурсников, в котором один из них упомянул постфиксную запись выражений. Нашему герою стало очень интересно, что это за такая запись. Поэтому он решил поискать про неё в интернете и вообще разобраться в вопросе.

Оказалось, постфиксная запись (или обратной польской записи) выражения - это такая запись, в которой операция записывается после двух операндов. Например, сумма двух чисел A и B записывается как $A B +$. Запись $B C + D *$ обозначает привычное нам $(B + C) * D$, а запись $A B C + D * +$ означает $A + (B + C) * D$.

Затем Вася узнал, что достоинство постфиксной записи в том, что она не требует скобок и дополнительных соглашений о приоритете операторов для своего чтения, и решил попробовать вычислить несколько выражений в таком виде. Для проверки своих ответов, он просит написать вас программу, которая будет вычислять написанные им выражения.

Входные данные

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

Выходные данные

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

Пример

входные данные	Скопировать
7 1 2 + * 3 9 - *	
выходные данные	Скопировать
-126	

Идея решения

Очередная задача на использование дека. Пушим числа в дек, если встречаем знак операции, то попадаем 2 числа из дека и производим действие над числами, ответ пушим обратно в дек. Итоговая сложность: $O(n)$

Исходный код

```
1 | #include <iostream>
2 | #include <algorithm>
3 | #include <vector>
4 | #include <string>
5 | #include <iomanip>
6 | #include <deque>
7 |
8 |
9 |
10 | using namespace std;
11 |
12 |
13 | int main(){
14 |     ios::sync_with_stdio(false);
15 |     cin.tie(0);
16 |     cout.tie(0);
17 |     string t;
18 |     deque <string> dack;
```

```

19 while (cin >> t){
20     if (t == " "){
21         continue;
22     }
23     else if (t == "*"){
24         auto t1 = dack.back();
25         dack.pop_back();
26         auto t2 = dack.back();
27         dack.pop_back();
28         dack.push_back(to_string(stoll(t1) * stoll(t2)));
29     }
30     else if (t == "-"){
31         auto t1 = dack.back();
32         dack.pop_back();
33         auto t2 = dack.back();
34         dack.pop_back();
35         dack.push_back(to_string(stoll(t2) - stoll(t1)));
36     }
37     else if (t == "+"){
38         auto t1 = dack.back();
39         dack.pop_back();
40         auto t2 = dack.back();
41         dack.pop_back();
42         dack.push_back(to_string(stoll(t1) + stoll(t2)));
43     }
44     else{
45         dack.push_back(t);
46     }
47 }
48 cout << dack[0];
49 return 0;
50 }

```

Фрагмент турнирной таблицы контеста

Положение

по ячейкам таблицы для просмотра истории

Совпадений: 2

Постнов

№	Кто	=	Штраф	A	B	C	D	E	F	G	H	I	J	K	L
16	Постнов Александр Вячеславович M80-1015-21	5	587	+ 00:23	+ 00:32	+1 02:20		+1 02:24		+3 02:28					
	* Постнов Александр Вячеславович M80-1015-21	7					+7		+5		+3	+	+	+5	+7

Вывод

Задача решена.

Теория чисел [5]

Е. Делимость

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

Проверьте делится ли заданное число на 97.

Входные данные

В первой строке входного файла вам дано неотрицательное число N количество знаков в котором не превышает 10^6 .

Выходные данные

Выведите строку "YES" без кавычек, если заданное число делится на 97, в противном случае выведите строку "NO" без кавычек. Регистр не важен.

Примеры

входные данные	Скопировать
97	
выходные данные	Скопировать
YES	

Идея решения

Стоит обратить внимание, что кол-во знаков в числе 10^6 . Это означает, что число гигантское и нельзя взять просто остаток от деления. Значит, работаем с числом как со строкой, необходимо реализовать алгоритм деления в столбик. Итоговая сложность: $O(n)$, где n - кол-во знаков в числе.

Исходный код

```
1 | #include <bits/stdc++.h>
2 |
3 | using namespace std;
4 | #define int long long
5 |
6 | int32_t main(){
7 |     ios::sync_with_stdio(false);
8 |     cin.tie(0);
9 |     cout.tie(0);
10 |     string s;
11 |     cin >> s;
12 |     string temp = "";
13 |     for (auto j : s){
14 |         temp += j;
15 |         if (stoll(temp) < 97){
16 |             continue;
17 |         }
18 |         else {
19 |             int number = stoll(temp);
20 |             number = number % 97;
21 |             temp = to_string(number);
22 |         }
23 |     }
24 |     if (stoll(temp) == 0){
25 |         cout << "YES" << "\n";
```

```
26 |     }
27 |     else {
28 |         cout << "NO" << "\n";
29 |     }
30 |     return 0;
31 | }
```

Фрагмент турнирной таблицы контеста

Вы можете использовать двойной щелчок (или Ctrl + щелчок) по ячейкам таблицы для просмотра истории

Положение		Совпадений: 2 <input type="text" value="постнов"/>										
№	Кто	=	Штраф	A	B	C	D	E	F	G	H	I
17	Постнов Александр Вячеславович М8О-101Б-21	3	209	+2 00:08	+ 01:02			-17	+1 01:19			
	* Постнов Александр Вячеславович М8О-101Б-21	6				+1	+1	+4		+14	+2	+2

Вывод

Задача дорешена.

Г. 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 ($2 \leq N \leq 2 \cdot 10^7$)

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

В этой задаче стоит использовать идею динамического программирования. Создадим массив $a[n]$, в ячейках находится минимальная стоимость преобразования числа в 0, используя заданные операции. Просчитаем для всех чисел до n . Тогда разделим числа на 4 случая, когда число делится только на 2, только на 3, и на 2 и 3 одновременно и ни на что не делится. Рассмотрим случай например, когда число делится только на 2: $a[i] = \min(a[i/2], a[i-1]) + i$. Аналогично для других случаев. Ответ содержится в $a[n]$. Итоговая сложность: $O(n)$

Исходный код


```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4 #define int long long
5
6 int32_t main(){
7     ios::sync_with_stdio(false);
8     cin.tie(0);
9     cout.tie(0);
10    int n;
11    cin >> n;
12    vector<int> a(n + 1, 0);
13    a[1] = 0;
14    a[2] = 2;
15    for (int i = 3; i <= n; i++){
16        if (i % 3 == 0 && i % 2 != 0){
17            a[i] += min(a[i / 3], a[i - 1]) + i;
```

```
18  
19     }  
20     else if (i % 2 == 0 && i % 3 != 0){  
21         a[i] += min(a[i / 2], a[i - 1]) + i;  
22     }  
23     else if (i % 2 == 0 && i % 3 == 0) {  
24         auto temp = min(a[i / 3], a[i / 2]);  
25         a[i] += min(temp, a[i - 1]) + i;  
26     }  
27     else {  
28         a[i] += a[i - 1] + i;  
29     }  
30 }  
31 cout << a[n];  
32 return 0;  
33 }
```

Фрагмент турнирной таблицы контеста

Результаты

Вы можете использовать двойной щелчок (или Ctrl + щелчок) по ячейкам таблицы для просмотра истории

Совпадений: 2  Постнов										
C	D	E	F	G	H	I	J	K	L	
+1 01:03		-6	+ 01:29							
	+3	+1		+1	+6	+	+	+27	+1	

Вывод

Задача дорешена.

Арифметика в кольце, комбинаторика, функция Эйлера [7]

Н. Черепашка в безопасности

ограничение по времени на тест: 1 секунда

ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

Черепашка живёт на прямоугольном поле и хочет добраться из точки $(0, 0)$ в точку (n, m) . Передвигается она таким образом, что из точки с координатами (i, j) может попасть только в точки $(i + 1, j)$ и $(i, j + 1)$. Помогите черепашке определить, сколькими путями она сможет добраться до пункта назначения. Так как ответ может быть очень большой, выведите его по модулю $10^9 + 7$

Входные данные

В первой строке вам даны два числа n и m ($0 \leq n, m \leq 10^7$) — пункт назначения.

Выходные данные

Выведите единственное число — ответ на задачу

Пример

входные данные	Скопировать
4 4	
выходные данные	Скопировать
70	

Идея решения

Эту задачу можно решать с помощью динамического программирования, где в $a[i][j]$ хранится кол-во путей, чтобы добраться в клетку с координатами i, j , но это слишком долго, так как итоговая сложность будет $O(n * m)$, а n, m могут достигать 10^7 . Поэтому используем комбинаторику. Итоговый ответ - это $(n + m)! / (n! * m!)$

Исходный код

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4 #define int long long
5
6 int factmod (int n, int p) {
7     int res = 1;
8     while (n > 1) {
9         res = (res * ((n/p) % 2 ? p-1 : 1)) % p;
10        for (int i=2; i<=n%p; ++i)
11            res = (res * i) % p;
12        n /= p;
13    }
14    return res % p;
15 }
16 int bin_pow(int a, int n, int mod) {
17     int res = 1;
18     while (n > 0) {
19         if (n % 2 == 1)
20             res = (res * a) % mod;
21         a = (a * a) % mod;
22         n /= 2;
```

```
23     }
24     return res;
25 }
26
27
28 int32_t main(){
29     ios::sync_with_stdio(false);
30     cin.tie(0);
31     cout.tie(0);
32     int mod = 1000000000 + 7;
33     int n, m;
34     cin >> n >> m;
35     int s = factmod(n + m, mod);
36
37     int a1 = bin_pow(factmod(n, mod), mod - 2, mod);
38     int a2 = bin_pow(factmod(m, mod), mod - 2, mod);
39     int answer = ((s * a1) % mod * a2) % mod;
40     cout << answer << "\n";
41     return 0;
42 }
```

Фрагмент турнирной таблицы контеста

Вы можете использовать двойной щелчок (или Ctrl + щелчок) по ячейкам таблицы для просмотра истории

Положение		Совпадений: 2											
№	Кто	=	Штраф	A	B	C	D	E	F	G	H	I	J
4	Постнов Александр Вячеславович М8О-101Б-21	3	125	+ 00:02	+ 00:11		+ 01:52	-6			-4		
	* Постнов Александр Вячеславович М8О-101Б-21	7				+1		+	+26	+3	+	+1	+

Вывод

Задача дорешена.

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

Ж. Ясновидацкий

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

Василий — начинающий игрок на бирже. А ещё он ясновидацкий. Недавно он заглянул в будущее и узнал изменения цен на некоторый товар в ближайшее время. Теперь он хочет узнать, каковы его оптимальные действия, если он хочет заработать как можно больше. Он не хочет привлекать к себе внимание, поэтому решил, что сделает только одну покупку и только одну продажу. Определите в какие моменты времени ему наиболее выгодно совершить покупку и продажу.

Входные данные

В первой строке вам дано число N ($1 \leq N \leq 2 \cdot 10^5$) — количество изменений цены, о которых узнал Василий. В следующей строке даны N чисел, a_i ($|a_i| \leq 10^9$) — отрицательные значения говорят о том что цена падает, положительные — что растёт.

Выходные данные

Выведите два числа — момент времени, в который Василию нужно купить товар и момент времени для продажи, если существует несколько пар времён, которые дадут одинаковый доход, выведите ту пару, которая имеет наиболее ранний момент покупки и продажи. Если пары моментов времени, дающих прибыль, нет, то выведите «-1 -1» (без кавычек).

Примеры

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

Идея решения

Буду использовать метод двух указателей. Первый указатель следит за минимумом, а второй проходит по массиву. Если разница между текущим числом и миним. максимальная, запоминаем позиции минимума и текущего числа. Итоговая сложность: $O(n)$

Исходный код

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4 #define int long long
5
6 int32_t main(){
7     ios::sync_with_stdio(false);
8     cin.tie(0);
9     cout.tie(0);
10     int n;
11     cin >> n;
12     vector<int> a(n);
13     for (int i = 0; i < n; i++){
14         cin >> a[i];
15     }
16     int ans = a[0],
```



```

17     ans_l = 0,
18     ans_r = 0,
19     sum = 0,
20     min_sum = 0,
21     min_pos = -1;
22     for (int r = 0; r < n; ++r) {
23         sum += a[r];
24
25         int cur = sum - min_sum;
26         if (cur > ans) {
27             ans = cur;
28             ans_l = min_pos + 1;
29             ans_r = r;
30         }
31
32         if (sum < min_sum) {
33             min_sum = sum;
34             min_pos = r;
35         }
36     }
37     if (ans > 0){
38         cout << ans_l + 1 << " " << ans_r + 2 << "\n";
39     }
40     else {
41         cout << -1 << " " << -1 << "\n";
42     }
43     return 0;
44 }

```

Фрагмент турнирной таблицы контеста

Вы можете использовать двойной щелчок (или Ctrl + щелчок) по ячейкам таблицы для просмотра истории

Положение 		Совпадений: 2  <input type="text" value="постнов"/>													
№	Кто	=	Штраф	A	B	C	D	E	F	G	H	I	J	K	L
6	Постнов Александр Вячеславович M8O-101B-21	5	117	+ 00:02	+ 00:10	+ 00:15	+1 00:30		+ 00:40			-1			
	* Постнов Александр Вячеславович M8O-101B-21	8		+				+1		+	+2	+3	+1	+	+

Вывод

Задача дорешена.

Двумерное ДП, задача о рюкзаке [9]

Ф. Путь рыцаря

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

Василий начинающий рыцарь, и сегодня он решил отправиться совершать подвиги чтобы стать более богатым и известным. Он живёт в королевстве расположенном на прямоугольной сетке, в которой левая верхняя ячейка имеет координату $(1, 1)$, а правая нижняя (N, M) . Для начала он решил разобраться с богатством. Он разослал своих слуг чтобы они узнали, кто в королевстве готов заплатить за его помощь, и теперь планирует свой путь. Будучи суеверным рыцарем и любителем шахмат, он решил, что будет передвигаться по королевству только ходом шахматного коня и только в направлениях удаляющих его от начальной точки. То есть из клетки (i, j) он будет перемещаться только в клетки $(i + 2, j - 1)$, $(i + 2, j + 1)$, $(i + 1, j + 2)$ и $(i - 1, j + 2)$. Помогите ему определить максимальное количество золота которое он сможет заработать, если будет получать оплату за свою помощь в каждой посещённой клетке.

Входные данные

В первой строке вам даны два числа N и M ($1 \leq N, M \leq 1000$) — размеры мира. В следующих N строках заданы по M чисел a_{ij} ($0 \leq a_{ij} \leq 1000$), количество золота, которое рыцарь сможет заработать проходя по данной области.

Выходные данные

Выведите единственное число, максимальное количество золота, которое рыцарь сможет заработать в своём путешествии.

Примеры

входные данные	Скопировать
2 2 1 1 1 1	
выходные данные	Скопировать
1	

Идея решения

В этой задаче нужно использовать метод ДП, в ячейке $a[i][j]$ будет храниться максимальное кол-во золота, которое он может заработать. Так как путь рыцаря довольно сложно реализуемый на циклах, буду использовать рекурсию, чтобы она была эффективной, буду проверять, что клетка не была посещена. Итоговая сложность: $O(n * m)$

Исходный код

```
1 | #include <bits/stdc++.h>
2 |
3 |
4 |
5 | using namespace std;
6 | #define int long long
7 |
8 | int solve(int n, int m, vector <vector <int>>& a, vector <vector <int>>& dp, int nn, int mm){
9 |     if (dp[n][m] != -100){
10 |         return dp[n][m];
11 |     }
12 |     if (n - 2 >= 0 && m + 1 < mm){
13 |         dp[n - 2][m + 1] = solve(n - 2, m + 1, a, dp, nn, mm);
14 |         if (dp[n - 2][m + 1] > 0){
15 |             dp[n][m] = max(dp[n][m], dp[n - 2][m + 1] + a[n][m]);
16 |         }
17 |     }
18 |     if (n - 2 >= 0 && m - 1 >= 0){
```

```

19     dp[n - 2][m - 1] = solve(n - 2, m - 1, a, dp, nn, mm);
20     if (dp[n - 2][m - 1] > 0){
21         dp[n][m] = max(dp[n][m], dp[n - 2][m - 1] + a[n][m]);
22     }
23 }
24 if (n - 1 >= 0 && m - 2 >= 0){
25     dp[n - 1][m - 2] = solve(n - 1, m - 2, a, dp, nn, mm);
26     if (dp[n - 1][m - 2] > 0){
27         dp[n][m] = max(dp[n][m], dp[n - 1][m - 2] + a[n][m]);
28     }
29 }
30 if (n + 1 < nn && m - 2 >= 0){
31     dp[n + 1][m - 2] = solve(n + 1, m - 2, a, dp, nn, mm);
32     if (dp[n + 1][m - 2] > 0){
33         dp[n][m] = max(dp[n][m], dp[n + 1][m - 2] + a[n][m]);
34     }
35 }
36 if (dp[n][m] == -100){
37     dp[n][m] = 0;
38 }
39 return dp[n][m];
40 }
41
42
43 int32_t main(){
44     ios::sync_with_stdio(false);
45     cin.tie(0);
46     cout.tie(0);
47     int n, m;
48     cin >> n >> m;
49     vector <vector <int>> a(n, vector <int>(m, -100));
50     vector <vector <int>> dp(n, vector <int>(m, -100));
51     for (int i = 0; i < n; i++){
52         for (int j = 0; j < m; j++){
53             cin >> a[i][j];
54         }
55     }
56     int maxis = 0;
57     dp[0][0] = a[0][0];
58     for (int i = 0; i < n; i++){
59         for (int j = 0; j < m; j++){
60             dp[i][j] = solve(i, j, a, dp, n, m);
61             if (dp[i][j] > maxis){
62                 maxis = dp[i][j];
63             }
64         }
65     }
66     cout << maxis << "\n";
67     return 0;
68 }




```


Фрагмент турнирной таблицы контеста

Двумерное ДП, задача о рюкзаке [9]

Результаты

Вы можете использовать двойной щелчок (или Ctrl + щелчок) по ячейкам таблицы для просмотра истории

Положение 													
Совпадений: 3   по													
№	Кто	=	Штраф	A	B	C	D	E	F	G	H	I	J
18	Постнов Александр Вячеславович М8О-101Б-21	2	94	+1 00:24	+ 00:50						-8		
	* Постнов Александр Вячеславович М8О-101Б-21	5				+	+	+	+2	+	-8		
	Количество решивших			40	35	25	23	13	13	11	6	5	5
	Количество попытавшихся			40	36	27	39	13	17	18	15	6	5

Вывод

Задача дорешена.

Геометрия, тернарный поиск [10]

I. Квадраты

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

Студенту первого курса 8-го факультета Василию Зайцеву подарили на день рождения набор точек. Так как Василий очень любил простые геометрические фигуры, он решил составить квадрат максимальной площади, такой, чтобы его углы были точками из набора. Однако Василий Зайцев был не фанатом лекций и семинаров по информатике, поэтому справиться с данной задачей он не смог. Помогите ему найти такой квадрат, а точнее его площадь.

Входные данные

В первой строке задано целое число N ($1 \leq N \leq 2000$) — количество точек. Следующие N строк содержат координаты точек — два целых числа x и y . Обе координаты не превосходят по модулю 10^9 .

Выходные данные

Выведите максимальную площадь квадрата, построенного на заданных точках. Если такого не существует, выведите «0».

Примеры

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

Идея решения

Зафиксирую 2 координаты и построю по ним квадрат. 2 другие координаты могут быть расположены 2 способами. Вычисляю площадь. Если один из способов присутствует в множестве координат, значит построить квадрат именно так возможно. Обновляю по необходимости ответ. Итоговая сложность: $O(n^2 * \log_2 n)$

Исходный код

```
1 | #include <bits/stdc++.h>
2 |
3 |
4 |
5 | using namespace std;
6 | #define int long long
7 |
8 |
9 |
10 | int32_t main(){
11 |     ios::sync_with_stdio(false);
12 |     cin.tie(0);
13 |     cout.tie(0);
14 |     int n;
15 |     cin >> n;
16 |     vector <pair <int, int>> a;
17 |     set <pair <int, int>> s;
18 |     int answer = 0;
```

```
19   for (int i = 0; i < n; i++){
20       int x, y;
21       cin >> x >> y;
22       a.push_back({x, y});
23       s.insert({x, y});
24   }
25   for (int i = 0; i < n; i++){
26       for (int j = i + 1; j < n; j++){
27           int d = (a[i].first - a[j].first) * (a[i].first - a[j].first) + (a[i].second - a[j].second)
28               * (a[i].second - a[j].second);
29           int x1 = a[i].first - a[j].first;
30           int y1 = a[i].second - a[j].second;
31           int dx = y1;
32           int dy = -x1;
33
34           int x_a1 = a[i].first + dx;
35           int y_a1 = a[i].second + dy;
36           int x_a2 = a[j].first + dx;
37           int y_a2 = a[j].second + dy;
38
39           int x_b1 = a[i].first - dx;
40           int y_b1 = a[i].second - dy;
41           int x_b2 = a[j].first - dx;
42           int y_b2 = a[j].second - dy;
43           if ((s.count({x_a1, y_a1}) && s.count({x_a2, y_a2})) || (s.count({x_b1, y_b1}) && s.count({
44               x_b2, y_b2}))) {
45               if (d > answer) {
46                   answer = d;
47               }
48           }
49       }
50   }
51   cout << answer << "\n";
52   return 0;
53 }
```

Фрагмент турнирной таблицы контеста

результаты

Вы можете использовать двойной щелчок (или Ctrl + щелчок) по ячейкам таблицы для просмотра истории

Положение

Совпадений: 2

Пост

№	Кто	=	Штраф	A	B	C	D	E	F	G	H	I	J
9	Постнов Александр Вячеславович М80-101Б-21	5	387	+ 00:17	+ 00:47	+ 00:54	+2 01:31	+1 01:58					
	* Постнов Александр Вячеславович М80-101Б-21	6						+	+1	+	+3	+	+4

Вывод

Задача дорешена.

Осенняя олимпиада первого курса

В. Симулятор чёрных дыр

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

Максим посмотрел много видео про космос и чёрные дыры. Он не смог позволить себе любительский телескоп, да и разглядеть в него чёрную дыру не получится, поэтому он решил запустить симулятор космических объектов на компьютере.

Максим называет чёрной дырой небесное тело настолько большой массы и плотности, что всё, что пересекает «горизонт событий», не может покинуть его, даже объект, движущийся со скоростью света. Чёрная дыра характеризуется своим радиусом Шварцшильда или просто размером. Сверхмассивная чёрная дыра — это чёрная дыра с массой от 10^5 до 10^{11} масс Солнца. Сверхмассивные чёрные дыры обнаружены в центре многих галактик, включая Млечный Путь.

Недавний снимок сверхмассивной чёрной дыры в центре галактики *M 87*:

Максим решил поиграться с чёрными дырами, сталкивая их. Как известно, при столкновении одна чёрная дыра поглощает другую, и образуется новая чёрная дыра, размер которой равен сумме размеров столкнувшихся чёрных дыр. Чёрная дыра может поглотить другую чёрную дыру, если её размер **строго больше** размера поглощаемой. Изначально в центре галактики находится сверхмассивная чёрная дыра массой Ω масс Солнца. По галактике разбросаны другие чёрные дыры, массы которых ω_i масс Солнца. Максим может перемещать центральную чёрную дыру в любую точку пространства и сталкивать с любой другой чёрной дырой строго меньшего размера. Максим хочет узнать, может ли сверхмассивная чёрная дыра в центре галактики поглотить все другие чёрные дыры.

Входные данные

В первой строке записаны два числа n и Ω ($1 \leq n \leq 2 \cdot 10^5$, $1 \leq \Omega \leq 10^9$) — количество чёрных дыр в галактике и начальная масса сверхмассивной чёрной дыры в центре галактики.

В следующей строке находится n чисел ($1 \leq \omega_i \leq 10^9$) — массы чёрных дыр, разбросанных по галактике.

Выходные данные

Если сверхмассивная чёрная дыра не может поглотить все чёрные дыры галактики, выведите «NO». Иначе выведите «YES» и порядок, в котором Максиму следует сталкивать сверхмассивную чёрную дыру с другими чёрными дырами.

Примеры

входные данные	Скопировать
1 3 3	
выходные данные	Скопировать
NO	

Идея решения

Чтобы знать в каком порядке черная дыра поглощала другие дыры, необходимо создать структуру, в которой содержится ее масса и номер. Сортируем дыры по массе. И идем от меньшей к большей. Проверяем, может ли наша дыра съесть текущую дыру, если нет, то ответ на задачу нет, если да, то поглощаем и продвигаемся дальше. Итоговая сложность: $O(n * \log_2 n)$ (сложность сортировки)




Исходный код

```
1 | #include <bits/stdc++.h>
2 |
3 |
4 |
5 | using namespace std;
6 | #define int long long
7 |
8 |
9 |
10 |
11 | int32_t main(){
```

```
12 ios::sync_with_stdio(false);
13 cin.tie(0);
14 cout.tie(0);
15 int n, m;
16 cin >> n >> m;
17 vector <pair <int, int>> a(n);
18 for (int i = 0; i < n; i++){
19     cin >> a[i].first;
20     a[i].second = i + 1;
21 }
22 sort(a.begin(), a.end());
23 int flag = 0;
24 vector <int> ans;
25 for (int i = 0; i < n; i++){
26     if (a[i].first >= m){
27         flag = 1;
28         break;
29     }
30     else {
31         m += a[i].first;
32         ans.push_back(a[i].second);
33     }
34 }
35 if (flag == 0){
36     cout << "YES \n";
37     for (auto i : ans){
38         cout << i << " ";
39     }
40 }
41 else {
42     cout << "NO";
43 }
44 return 0;
45 }
```

Фрагмент турнирной таблицы контеста

Вы можете использовать двойной щелчок (или Ctrl + щелчок) по ячейкам таблицы для просмотра истории

Положение 														Совпадений: 2   постнов									
№	Кто	=	Штраф	A	B	C	D	E	F	G	H	I	J	K									
16	Постнов Александр Вячеславович MSO-101B-21	3	337	+ 00:17	+1 00:41		+3 03:19					-5											
	* Постнов Александр Вячеславович MSO-101B-21	2				+2		+															

Вывод

Задача решена.

Тренировочный констест на разные темы

В. Перестановка

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

Для заданного массива a из n целых чисел и целого числа m узнайте, можно ли изменить порядок элементов массива a так, чтобы $\sum_{i=1}^n \sum_{j=i}^n \frac{a_j}{j}$ было равно m ? Удалять элементы, а также добавлять новые запрещается. Обратите внимание, при делении не происходит округления, например, $\frac{5}{2} = 2.5$.

Входные данные

В первой строке задается целое число t — количество тестовых случаев ($1 \leq t \leq 100$). Далее задаются сами тестовые случаи, в двух строках каждый.

В первой строке тестового случая задаются два целых числа n и m ($1 \leq n \leq 100, 0 \leq m \leq 10^6$). Во второй строке задаются целые числа a_1, a_2, \dots, a_n — элементы массива ($0 \leq a_i \leq 10^6$).

Выходные данные

Для каждого тестового случая выведите «YES», если существует такая перестановка элементов массива, что заданная формула равна заданному значению, а иначе выведите «NO».

Пример

входные данные	Скопировать
2 3 8 2 5 1 4 4 0 1 2 3	
выходные данные	Скопировать
YES NO	

Идея решения

Методом пристального взгляда можно определить, что формула в условии обозначает просто сумму элементов массива. Задача становится тривиальной. Итоговая сложность: $O(t * n)$

Исходный код

```
1 #include <bits/stdc++.h>
2
3
4
5 using namespace std;
6 #define int long long
7
8
9 int32_t main(){
10     ios::sync_with_stdio(false);
11     cin.tie(0);
12     cout.tie(0);
13     int t;
14     cin >> t;
15     for (int i = 0; i < t; i++){
16         int n, m;
17         int s = 0;
```

```
18     cin >> n >> m;
19     for (int j = 0; j < n; j++){
20         int temp;
21         cin >> temp;
22         s += temp;
23     }
24     if (s == m){
25         cout << "YES \n";
26     }
27     else {
28         cout << "NO \n";
29     }
30 }
31 return 0;
32 }
```

Фрагмент турнирной таблицы контеста

Тренировочный контест на разные темы

Результаты

Вы можете использовать двойной щелчок (или Ctrl + щелчок) по ячейкам таблицы для просмотра истории

Положение															Совпадений: 1					постнов				
№	Кто	=	Штраф	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O						
4	Постнов Александр Вячеславович M8O-101B-21	5	311	+ 00:12	+ 00:22	+1 02:03	+ 00:44	-1				+ 01:30												

Вывод

Задача решена.

Основы теории графов [11]

Г. Поиск в ширину на клетчатом поле

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

Для каждой проходимой клетки поля определите длину кратчайшего пути от заданной начальной клетки. Если клетка непроходима или пути до неё не существует, длина пути считается равной -1 .

Входные данные

В первой строке вам заданы два числа N и M ($1 \leq N, M \leq 100$) — количество строк и столбцов в решётке. В следующих N строках вам дано описание решётки, символ '.' означает что клетка проходима, '#' — непроходима. В последней строке вам заданы два числа x и y ($1 \leq x \leq N, 1 \leq y \leq M$) — номер строки и столбца клетки до которой требуется рассчитать кратчайшие пути. Гарантируется что эта клетка проходима.

Выходные данные

Выведите $N \times M$ чисел — для каждой клетки длину кратчайшего пути до неё.

Примеры

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

Идея решения

Необходимо реализовать bfs для двумерного массива.

Исходный код

```
1 | #include <bits/stdc++.h>
2 |
3 |
4 |
5 | using namespace std;
6 | using graph = vector<vector<int>>>;
7 |
8 | #define int long long
9 |
10 | vector <string> ans;
11 |
12 | void bfs(int u,graph &g, vector<int> &dist, vector <int>& path) {
13 |     queue<int> q;
14 |     q.push(u);
15 |     dist[u] = 0;
16 |     while(!q.empty()) {
17 |         int s = q.front();
18 |         q.pop();
19 |         for(int v: g[s]) {
20 |             if(dist[v] == -1) {
```



```

21         q.push(v);
22         dist[v] = dist[s] + 1;
23         path[v] = s;
24     }
25 }
26 }
27 }
28
29
30 void bfs_dv(pair <int, int>temp, vector <vector <char>> &enable, vector <vector <int>>& dist, int n,
    int m) {
31     queue<pair<int, int>> q;
32     q.push(temp);
33     dist[temp.first][temp.second] = 0;
34     while(!q.empty()) {
35         auto s = q.front();
36         q.pop();
37         //cout << s.first << " " << s.second << "\n";
38         if (s.first + 1 < n && enable[s.first + 1][s.second] != '#' && dist[s.first + 1][s.second] ==
            -1){
39             q.push({s.first + 1, s.second});
40             dist[s.first + 1][s.second] = dist[s.first][s.second] + 1;
41         }
42         if (s.first - 1 >= 0 && enable[s.first - 1][s.second] != '#' && dist[s.first - 1][s.second] ==
            -1){
43             q.push({s.first - 1, s.second});
44             dist[s.first - 1][s.second] = dist[s.first][s.second] + 1;
45         }
46         if (s.second + 1 < m && enable[s.first][s.second + 1] != '#' && dist[s.first][s.second + 1] ==
            -1){
47             q.push({s.first, s.second + 1});
48             dist[s.first][s.second + 1] = dist[s.first][s.second] + 1;
49         }
50         if (s.second - 1 >= 0 && enable[s.first][s.second - 1] != '#' && dist[s.first][s.second - 1] ==
            -1){
51             q.push({s.first, s.second - 1});
52             dist[s.first][s.second - 1] = dist[s.first][s.second] + 1;
53         }
54     }
55 }
56
57 void print_graph(graph g, int n){
58     for (int i = 0; i < n; i++){
59         cout << i + 1 << ": ";
60         for (int j : g[i]){
61             cout << j + 1 << " ";
62         }
63         cout << "\n";
64     }
65 }
66
67 void print_chars(vector <vector <char>> a, int n, int m){
68     for (int i = 0; i < n; i++){
69         for (int j = 0; j < m; j++){
70             cout << a[i][j] << " ";
71         }
72         cout << "\n";
73     }
74 }
75

```

```

76 int32_t main(){
77     // ios::sync_with_stdio(false);
78     // cin.tie(0);
79     // cout.tie(0);
80     int n, m;
81     cin >> n >> m;
82     vector <vector <char>> enable(n, vector <char>(m));
83     vector <vector <int>> dist(n, vector <int>(m, -1));
84     for (int i = 0; i < n; i++){
85         for (int j = 0; j < m; j++){
86             cin >> enable[i][j];
87         }
88     }
89     int a, b;
90     cin >> a >> b;
91     a--;
92     b--;
93     bfs_dv({a, b}, enable, dist, n, m);
94     for (int i = 0; i < n; i++){
95         for (int j = 0; j < m; j++){
96             cout << dist[i][j] << " ";
97         }
98         cout << "\n";
99     }
100     //print_graph(g, n);
101     return 0;
102 }

```

Фрагмент турнирной таблицы контеста

Положение		Совпадений: 2 Постнов									
№	Кто	=	Штраф	A	B	C	D	E	F	G	H
17	Постнов Александр Вячеславович М80-101Б-21	3	267	+1 00:25	+ 00:28	+5 01:34		-1			
	* Постнов Александр Вячеславович М80-101Б-21	5					+7	+4	+3	+	+1

Вывод

Задача дорешена.

Кратчайшие пути во взвешенных графах [12]

Е. 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 кратчайший путь из вершины 1 в вершину n .

Входные данные

В первой строке содержатся целые числа n и m ($2 \leq n \leq 10^5$, $0 \leq m \leq 10^5$), где n — количество вершин, а m — количество ребер в графе. Далее в m строках содержатся сами ребра, по одному в строке. Каждое ребро задается тремя числами a_i, b_i, w_i ($1 \leq a_i, b_i \leq n$, $1 \leq w_i \leq 10^6$), где a_i, b_i — это концы ребра, а w_i — его длина.

Граф может содержать кратные ребра и петли.

Выходные данные

Выведите число -1 если пути нет, или сам кратчайший путь, если он существует.

Пример

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

Идея решения

Так как $n, m \leq 10^5$ необходимо использовать алгоритм Дейкстры для разреженных графов. Чтобы вывести найденный путь, буду использовать массив предков. Итоговая сложность: $O(m * \log_2 n)$

Исходный код

```
1 #include <bits/stdc++.h>
2
3
4
5 using namespace std;
6
7 #define int long long
8
9 using graph = vector < vector < pair<int,int> > >;
10
11 const int INF = 1e18;
12
13 int32_t main(){
14     ios::sync_with_stdio(false);
15     cin.tie(0);
16     cout.tie(0);
17     int n, m;
18     cin >> n >> m;
```



```

19 graph g(n);
20 for (int i = 0; i < m; i++){
21     int x, y, z;
22     cin >> x >> y >> z;
23     x--;
24     y--;
25     g[x].push_back({y, z});
26     g[y].push_back({x, z});
27 }
28 int s = 0;
29 vector<int> d (n, INF), p (n);
30 d[s] = 0;
31 set < pair<int,int> > q;
32 q.insert (make_pair (d[s], s));
33 while (!q.empty()) {
34     int v = q.begin()->second;
35     q.erase (q.begin());
36
37     for (size_t j=0; j<g[v].size(); ++j) {
38         int to = g[v][j].first,
39         len = g[v][j].second;
40         if (d[v] + len < d[to]) {
41             q.erase (make_pair (d[to], to));
42             d[to] = d[v] + len;
43             p[to] = v;
44             q.insert (make_pair (d[to], to));
45         }
46     }
47 }
48
49 int t = n - 1;
50 if (d[t] == INF){
51     cout << -1 << "\n";
52 }
53 else {
54     vector<int> path;
55     for (int v=t; v!=s; v=p[v])
56         path.push_back (v);
57     path.push_back (s);
58     reverse (path.begin(), path.end());
59     for (int b = 0; b < path.size(); b++){
60         cout << path[b] + 1 << " ";
61     }
62     cout << "\n";
63 }
64 return 0;
65 }

```

Фрагмент турнирной таблицы контеста

Вы можете использовать двойной щелчок (или Ctrl + щелчок) по ячейкам таблицы для просмотра истории

Положение 		Совпадений: 2  Постнов									
№	Кто	=	Штраф	A	B	C	D	E	F	G	H
2	Постнов Александр Вячеславович М8О-101Б-21	6	291	+ 00:40	+ 00:25	+ 00:17	+2 00:21	+1 00:30			+ 01:38
	* Постнов Александр Вячеславович М8О-101Б-21	2							+2	+6	

Вывод

Задача дорешена.

СНМ, минимальное остовное дерево [13]

Е. Разрушение графа

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

Промоделируйте процесс разрушения графа, при котором последовательно разрушаются рёбра соединяющие пары вершин. Гарантируется что в конце процедуры все рёбра в графе уничтожены.

Входные данные

В первой строке вам дано два числа N и M ($1 \leq N, M \leq 2 \cdot 10^5$) — количество вершин и рёбер в графе. Далее следуют M строк с описанием рёбер в порядке, в котором они разрушались, в виде пар чисел u_i и v_i ($1 \leq u_i, v_i \leq N$) — пара вершин соединяемых ребром.

Выходные данные

После каждого запроса выведите количество компонент связности в оставшемся графе.

Примеры

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

Идея решения

Строим граф в обратном порядке, выписывая компонент связности графа, используя СНМ.

Исходный код

```
1 #include <bits/stdc++.h>
2
3
4
5 using namespace std;
6
7 #define int long long
8
9 using graph = vector < vector < pair<int,int> > >;
10 struct edge {
11     int u, v, w;
12 };
13
14 bool operator < (const edge & lhs, const edge & rhs) {
15     return lhs.w < rhs.w;
16 }
17
18 struct dsu {
19     vector <int> leader;
20     vector <int> size;
21     dsu (int n) {
22         leader.resize(n);
```

```




23     for (int i = 0; i < n; i++){
24         leader[i] = i;
25     }
26     size.resize(n, 1);
27 }
28 int find(int u) {
29     if (leader[u] == u) {
30         return u;
31     }
32     int v = find(leader[u]);
33     leader[u] = v;
34     return v;
35 }
36 void join(int u, int v) {
37     int lead_u = find(u);
38     int lead_v = find(v);
39     if (lead_u == lead_v) {
40         return ;
41     }
42     if (size[lead_u] > size[lead_v]) {
43         leader[lead_v] = lead_u;
44         size[lead_u] += size[lead_v];
45     } else {
46         leader[lead_u] = lead_v;
47         size[lead_v] += size[lead_u];
48     }
49 }
50 };
51
52
53 const int INF = 1000000000;
54
55 int32_t main(){
56     ios::sync_with_stdio(false);
57     cin.tie(0);
58     cout.tie(0);
59     int n, m;
60     cin >> n >> m;
61     dsu d(n);
62     vector <pair <int, int>> q(m);
63     int ans = n;
64     for (int i = 0; i < m; i++){
65         int u, v;
66         cin >> u >> v;
67         u--;
68         v--;
69         q[i] = {u, v};
70     }
71     vector <int> answers(m);
72     answers[0] = n;
73     for (int i = m - 1; i >= 1; i--){
74         int u = q[i].first;
75         int v = q[i].second;
76         if (d.find(u) != d.find(v)){
77             d.join(u, v);
78             answers[i] = --ans;
79         }
80         else {
81             answers[i] = ans;
82         }

```

```
83 | }
84 | for (int i = 1; i < m; i++){
85 |     cout << answers[i] << "\n";
86 | }
87 | cout << answers[0] << "\n";
88 |
89 | return 0;
90 | }
```

Фрагмент турнирной таблицы контеста

по ячейкам таолицы для просмотра истории

Положение 		Совпадений: 2   <input type="text" value="постнов"/>								
№	Кто	=	Штраф	A	B	C	D	E	F	G
37	Постнов Александр Вячеславович М80-101Б-21	0	0			-12				
	* Постнов Александр Вячеславович М80-101Б-21	7		+1	+2	+	+	+	+1	+

Вывод

Задача дорешена.

Деревья, наименьший общий предок [14]

В. Красоты Берляндии (простая версия)

ограничение по времени на тест: 1 секунда

ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

В простой версии задачи в Берляндии не строят новые города каждый день, а в сложной версии требуется учитывать каждый новый построенный город.

Берляндия славится своими уездными городами и красотами по пути к ним. В стране находится n городов и их соединяют $n - 1$ дорога так, что из любого города можно попасть в любой другой.

Путешественник хочет как можно больше насладиться красотами Берляндии, но при этом он категорически против посетить какой-либо город дважды. Ваша задача — определить максимальную длину пути между двумя городами в Берляндии, чтобы путешественник смог дольше всего наслаждаться красотами страны.

Входные данные

На первой строке находится число t ($1 \leq t \leq 10^4$) — число запросов в тесте.

Каждый тест описывается числом n ($1 \leq n \leq 10^5$) — числом городов в Берляндии.

Следующие $n - 1$ строк описывают дороги между городами в виде пары чисел u и v ($1 \leq u, v \leq n, u \neq v$).

Гарантируется, что сумма n по всем запросам не превышает 10^5 .

Выходные данные

На каждый тестовой случай выведите единственное число — ответ на задачу.

Пример

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

Идея решения

Для того, чтобы определить максимальное расстояние между городами, использовать 2 раза алгоритм bfs. 1 раз применяем с любой вершины графа, а 2 раз с максимальным расстоянием от неё. В массиве dist после использования bfs будет находится ответ. Итоговая сложность: $O(n)$.

Исходный код

```
1 | #include <bits/stdc++.h>
2 |
3 |
4 | #define int long long
5 |
6 |
7 | using namespace std;
```

```

8
9 using pii = pair <int, int>;
10
11 using graph = vector <vector <int> >;
12
13
14
15 void bfs(int u, graph &g, vector<int> &dist, vector <int> &path) {
16     queue<int> q;
17     q.push(u);
18     dist[u] = 0;
19     while(!q.empty()) {
20         int s = q.front();
21         q.pop();
22         for(int v: g[s]) {
23             if(dist[v] == -1) {
24                 q.push(v);
25                 dist[v] = dist[s] + 1;
26                 path[v] = s;
27             }
28         }
29     }
30 }
31
32
33 int32_t main() {
34     ios::sync_with_stdio(false);
35     cin.tie(0);
36     cout.tie(0);
37     int q;
38     cin >> q;
39     for (int h = 0; h < q; h++) {
40         int n;
41         cin >> n;
42         graph g(n);
43         for (int i = 0; i < n - 1; i++){
44             int u, v;
45             cin >> u >> v;
46             u--;
47             v--;
48             g[u].push_back(v);
49             g[v].push_back(u);
50         }
51         int u_b = 0;
52         vector <int> path1(n, -1);
53         vector <int> path2(n, -1);
54         vector <int> dist1(n, -1);
55         vector <int> dist2(n, -1);
56         bfs(u_b, g, dist1, path1);
57         int u_max = 0;
58         int maxis = 0;
59         for (int i = 0; i < n; i++) {
60             if (dist1[i] > maxis) {
61                 maxis = dist1[i];
62                 u_max = i;
63             }
64         }
65         bfs(u_max, g, dist2, path2);
66         int ans = 0;
67         for (int i = 0; i < n; i++) {

```

```

68         if (dist2[i] > ans) {
69             ans = dist2[i];
70         }
71     }
72     cout << ans << "\n";
73 }
74 }

```

Фрагмент турнирной таблицы контеста

Вы можете использовать двойной щелчок (или Ctrl + щелчок) по ячейкам таблицы для просмотра истории

Положение 		Совпадений: 3  <input type="text" value="по"/>							
№	Кто	=	Штраф	A	B	C	D	E	F
5	Постнов Александр Вячеславович M8O-101Б-21	2	73	+ 01:08	+ 00:05				
	* Постнов Александр Вячеславович M8O-101Б-21	2				+2	-4	+10	
	Количество решивших Количество попытавшихся			31 35	37 42	17 36	5 6	6 7	2 4

Вывод

Задача решена.

Паросочетания в двудольном графе, потоки в транспортной сети [15]

D. Прочная кладка

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

Строителям нужно покрыть прямоугольную область размером $m \times n$ (m и n чётные) двумя слоями прямоугольных кирпичей размерами 1×2 . Первый слой кирпичей уже завершён. Второй слой (чтобы сделать кладку действительно прочной) должен быть выполнен так, что никакой кирпич не лежит целиком на кирпиче первого слоя.

По расположению кирпичей в первом слое определите возможное расположение кирпичей во втором слое или сообщите, что второй слой сделать невозможно.

Приведённые рисунки показывают расположение кирпичей в первом и втором слое. Размер области 2×4 . Каждый кирпич помечен номером на обоих его половинках.

Входные данные

В первой строке даны два чётных числа n и m ($2 \leq m, n \leq 100$).

Следующие n строк содержат по m чисел, которые описывают расположение кирпичей в первом слое. Каждый кирпич помечен двумя равными числами, записанными в квадратах области, покрываемых этим кирпичом. Все кирпичи помечены целыми числами в диапазоне от 1 до общего числа кирпичей.

Выходные данные

Если решения нет, выведите -1 . Если решение существует, выведите n строк по m чисел, которые должны описывать расположение кирпичей во втором слое описанным выше способом.

Примеры

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

Идея решения

Каждый кирпич будет вершиной графа, каждый кирпич будет связан с соседними кирпичами. Тогда необходимо просто максимальное паросочетание в этом графе и вывести в надобном порядке. Итоговая сложность: $O(n * m)$

Исходный код

```
1 | #include <bits/stdc++.h>
2 |
3 |
4 | using namespace std;
5 | #define int long long
6 |
7 | using pii = pair <int, int>;
8 |
9 | using graph = vector <vector <int> >;
10 |
11 | void bfs(int u, graph &g, vector<int> &dist) {
12 |     queue<int> q;
13 |     q.push(u);
```

```

14     dist[u] = 0;
15     while(!q.empty()) {
16         int s = q.front();
17         q.pop();
18         for(int v: g[s]) {
19             if(dist[v] == -1) {
20                 q.push(v);
21                 dist[v] = dist[s] + 1;
22             }
23         }
24     }
25 }
26
27 bool dfs_kuhn(graph &g, int u, vector <int> &matching, vector <bool> &visited) {
28     if (visited[u]) {
29         return false;
30     }
31     visited[u] = true;
32     for (int i = 0; i < g[u].size(); ++i) {
33         int v = g[u][i];
34         if (matching[v] == -1 || dfs_kuhn(g, matching[v], matching, visited)){
35             matching[v] = u;
36             return true;
37         }
38     }
39     return false;
40 }
41
42 int32_t main() {
43     ios::sync_with_stdio(false);
44     cin.tie(0);
45     cout.tie(0);
46     int n, m;
47     cin >> n >> m;
48     vector <vector <int>> a(n, vector <int>(m));
49     for (int i = 0; i < n; i++){
50         for (int j = 0; j < m; j++){
51             cin >> a[i][j];
52         }
53     }
54     graph g(n * m);
55     int counter = 0;
56     for (int i = 0; i < n; i++){
57         for (int j = 0; j < m; j++){
58
59             if (i - 1 >= 0){
60                 if (a[i - 1][j] != a[i][j]) {
61                     g[counter].push_back(counter - m);
62                 }
63             }
64             if (i + 1 < n) {
65                 if (a[i + 1][j] != a[i][j]) {
66                     g[counter].push_back(counter + m);
67                 }
68             }
69             if (j - 1 >= 0){
70                 if (a[i][j - 1] != a[i][j]) {
71                     g[counter].push_back(counter - 1);
72                 }
73             }

```

```

74         if (j + 1 < m) {
75             if (a[i][j + 1] != a[i][j]) {
76                 g[counter].push_back(counter + 1);
77             }
78         }
79         counter++;
80     }
81 }
82 vector<int> dist(n * m, -1);
83 for (int i = 0; i < n * m; i++){
84     if (dist[i] == -1) {
85         bfs(i, g, dist);
86     }
87 }
88 graph g1(n * m);
89 for (int i = 0; i < n * m; i++) {
90     for (int j : g[i]) {
91         if (dist[i] % 2 == 0){
92             g1[i].push_back(j);
93         }
94     }
95 }
96 vector<int> matching(n * m, -1);
97 for (int i = 0; i < n * m; ++i){
98     vector<bool> visited(n * m);
99     dfs_kuhn(g1, i, matching, visited);
100 }
101 counter = 0;
102 vector<vector<int>> ans(n, vector<int>(m));
103 for (int i = 0; i < n * m; ++i){
104     if (matching[i] != -1){
105         int brick1 = matching[i];
106         int brick2 = i;
107         int x1 = brick1 / m;
108         int y1 = brick1 - x1 * m;
109         int x2 = brick2 / m;
110         int y2 = brick2 - x2 * m;
111         ans[x1][y1] = counter + 1;
112         ans[x2][y2] = counter + 1;
113         counter++;
114     }
115 }
116 for (int i = 0; i < n; i++){
117     for (int j = 0; j < m; j++){
118         cout << ans[i][j] << " ";
119     }
120     cout << "\n";
121 }
122 return 0;
123 }

```

Фрагмент турнирной таблицы контеста

Положение

Совпадений: 3

№	Кто	=	Штраф	A	B	C	D	E	F
22	Постнов Александр Вячеславович M8O-101Б-21	2	270	+8 01:03		+2 00:07			
	* Постнов Александр Вячеславович M8O-101Б-21	4			+1		+1	+	+
	Количество решивших Количество попытавшихся			31 35	10 20	26 28	7 9	5 6	15 15

Вывод

Задача дорешена.

Строки, Z-функция, хеши, префиксное дерево [16]

D. Период строки

ограничение по времени на тест: 1 секунда

ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

Назовём периодом строки S такой её кратчайший префикс P , который, будучи повторён достаточное количество раз, даёт строку S . Найдите длину периода заданной строки.

Входные данные

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

Выходные данные

Выведите единственное число — длину периода строки.

Примеры

входные данные	Скопировать
aaaaa	
выходные данные	Скопировать
1	

Идея решения

В этой задаче необходимо использовать z-функцию строки. Тогда, чтобы определить период, необходимо пройти по строке, и если номер символа + значение z-функции == значению z-функции для 1 символа и период + период <= размеру строки, то ответ найден. Итоговая сложность: $O(n)$

Исходный код

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



```
26
27 int32_t main() {
28     ios::sync_with_stdio(false);
29     cin.tie(0);
30     cout.tie(0);
31     string s;
32     cin >> s;
33     vector<int> z = z_func(s);
34     int proverka = z[0];
35     int powerless = z[0];
36     for (int i = 1; i < s.size(); i++) {
37         if (i + z[i] == proverka && i + i <= s.size()){
38             powerless = i;
39             break;
40         }
41     }
42     cout << powerless << "\n";
43     cout << "\n";
44     return 0;
45 }
```

Фрагмент турнирной таблицы контеста

по ячейкам таблицы для просмотра истории

Положение		Совпадений: 2 <input type="text" value="пос"/>										
№	Кто	=	Штраф	A	B	C	D	E	F	G	H	I
6	Постнов Александр Вячеславович M8O-101B-21	5	352	+ 00:00	+1 00:04	+3 01:10	+1 01:44		-1			+ 01:14
	* Постнов Александр Вячеславович M8O-101B-21	2						+	+			

Вывод

Задача решена.

ДП по подмножествам, ДП по профилю [17]

D. Максимальный хог

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

Из заданного набора выберите k чисел с максимальным значением хог.

Входные данные

В первой строке заданы два числа n и k ($1 \leq k \leq n \leq 24$) — размер исходного набора и количество чисел, которые следует оставить. В следующей строке дан сам набор a_i ($1 \leq a_i \leq 10^8$).

Выходные данные

Выведите значение максимального хог для поднабора из k чисел.

Примеры

входные данные	Скопировать
5 3 1 2 3 4 5	

Идея решения

В этой задаче нельзя использовать какой-либо алгоритм кроме полного перебора. Перебор будет производиться с помощью двоичной маски, где 0 - не берем в набор, а 1 - берем. Кол-во единиц должно быть равно k . Итоговая сложность: $O(n * 2^n)$

Исходный код

```
1 #include <bits/stdc++.h>
2
3 #define int long long
4
5 using namespace std;
6 using graph = vector <vector <int> >;
7 const unsigned long long INF = 1e18;
8 const int MOD = 1e9 + 7;
9
10 int32_t main() {
11     ios::sync_with_stdio(false);
12     cin.tie(0);
13     cout.tie(0);
14     int n, k;
15     cin >> n >> k;
16     vector <int> a(n);
17     for (int i = 0; i < n; i++){
18         cin >> a[i];
19     }
20     //cout << (1 << n) << "\n";
21     int ans = -1;
22     for (int i = 0; i < (1 << n); i++){
23         int temp_ans = -1;
24         int counter = 0;
25         for (int j = 0; j < n; j++){
26             if ((i >> j) & 1) {
27                 counter++;
```

```

28         if (temp_ans == -1){
29             temp_ans = a[j];
30         }
31         else {
32             temp_ans ^= a[j];
33         }
34     }
35 }
36 if (counter == k) {
37     ans = max(ans, temp_ans);
38 }
39 }
40 cout << ans << "\n";
41 }




```

Фрагмент турнирной таблицы контеста

ДП по подмножествам, ДП по профилю [17]

Результаты

Вы можете использовать двойной щелчок (или Ctrl + щелчок) по ячейкам таблицы для просмотра истории

Положение 		Совпадений: 2   <input type="text" value="постно"/>									
№	Кто	=	Штраф	A1	A2	B	C	D	E	F	G
6	Постнов Александр Вячеславович M8O-101Б-21	3	51	+ 00:17	-2					+ 00:08	+1 00:06
	* Постнов Александр Вячеславович M8O-101Б-21	4			+	+	+	+			

Вывод

Задача дорешена.

Теория игр, функция Шпрага-Гранди [18]

Г. Фишка на прямоугольнике

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

Два игрока играют в игру на пустом клетчатом листе размером $N \times M$. Изначально в правой верхней клетке находится фишка. Каждый игрок на своём ходу может передвинуть фишку на клетку влево, на клетку вниз или по диагонали на клетку вниз-влево, проигрывает игрок, который не может сделать ход. Определите кто победит при оптимальной игре обоих игроков.

Входные данные

В первой строке вам даны два числа N и M ($1 \leq N, M \leq 10^4$) размеры листа.

Выходные данные

Если побеждает игрок делающий первый ход выведите «1» в противном случае выведите «2».

Примеры

входные данные	Скопировать
1 10	
выходные данные	Скопировать
1	

Идея решения

После построения массива по функции Гранди можно заметить закономерность, если n, m - нечетные, то побеждает 2 игрок, иначе - первый.

Исходный код

```
1 | #include <bits/stdc++.h>
2 |
3 | #define int long long
4 |
5 | using namespace std;
6 | using graph = vector <vector <int> >;
7 | const unsigned long long INF = 1e18;
8 |
9 | int32_t main() {
10 |     ios::sync_with_stdio(false);
11 |     cin.tie(0);
12 |     cout.tie(0);
13 |     int n, m;
14 |     cin >> n >> m;
15 |     if (n % 2 == 1 && m % 2 == 1) {
16 |         cout << 2 << "\n";
17 |     }
18 |     else {
19 |         cout << 1 << "\n";
20 |     }
21 |     return 0;
22 | }
```

Фрагмент турнирной таблицы контеста

по ячейкам таблицы для просмотра истории

Положение													Совпадений: 2		пост	
№	Кто	=	Штраф	A	B	C	D	E	F	G	H	I	J			
13	Постнов Александр Вячеславович M8O-101Б-21	1	2	<div><div>+</div><div>00:02</div></div>	-2	-2		-9								
	* Постнов Александр Вячеславович M8O-101Б-21	8			+	+	+2	+1	+	+	-5	+4	+			

Вывод

Задача дорешена.

Дерево отрезков [19]

B. GCD

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 64 мегабайта
ввод: стандартный ввод
вывод: стандартный вывод

Реализуйте структуру данных которая позволит вам отвечать на запросы о наибольшем общем делителе всех значений на отрезке.

Входные данные

В первой строке дано единственное число N ($1 \leq N \leq 10^5$) — число элементов в массиве. В следующей строке содержатся N натуральных чисел, не превосходящих 10^{18} — элементы массива. Далее идет число K ($0 \leq K \leq 10^5$) — количество запросов к структуре данных. Каждая из следующих K строк содержит два целых числа l и r ($1 \leq l \leq r \leq N$) — левую и правую границы отрезка в массиве для данного запроса.

Выходные данные

Для каждого запроса выведите наибольший общий делитель значений на заданном отрезке.

Пример

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

Идея решения

Для эффективного решения буду использовать дерево отрезков. Поскольку операция gcd ассоциативна ДО можно использовать. Построение ДО - $O(n * \log_2 n)$, нахождение gcd на отрезке - $O(\log_2 n)$

Исходный код

```
1 | #include <bits/stdc++.h>
2 |
3 | #define int long long
4 |
5 | using namespace std;
6 | using graph = vector <vector <int> >;
7 | const unsigned long long INF = 1e18;
8 |
9 | using pii = pair <int, int>;
10 |
11 | int gcd (int a, int b) {
12 |     return b ? gcd (b, a % b) : a;
13 | }
14 |
15 |
16 | struct seg_tree {
17 |     vector <int> data;
18 |     int n;
19 | }
```

```

20 seg_tree (int _n) {
21     n = _n;
22     data.resize(4 * n);
23 }
24
25 seg_tree(const vector <int> & a) {
26     n = a.size();
27     data.resize(4 * n);
28     build(1, 1, n, a);
29 }
30
31 int get(int ql, int qr) {
32     return get(1, ql, qr, 1, n);
33 }
34
35 int get(int id, int ql, int qr, int l, int r) {
36     if (ql <= l && r <= qr) {
37         return data[id];
38     }
39     int m = (l + r) / 2;
40     if (qr <= m) {
41         return get(id * 2, ql, qr, l, m);
42     }
43     if (ql > m) {
44         return get(id * 2 + 1, ql, qr, m + 1, r);
45     }
46     return gcd(get(id * 2, ql, qr, l, m), get(id * 2 + 1, ql, qr, m + 1, r));
47 }
48
49 void build(int id, int l, int r, const vector <int> & a) {
50     if (l == r) {
51         data[id] = a[l - 1];
52         return;
53     }
54     int m = (l + r) / 2;
55     build(id * 2, l, m, a);
56     build(id * 2 + 1, m + 1, r, a);
57     data[id] = gcd(data[id * 2], data[id * 2 + 1]);
58 }
59
60
61 void set(int p, int x) {
62     set(1, p, x, 1, n);
63 }
64
65 void set(int id, int p, int x, int l, int r) {
66     if (l == r) {
67         data[id] = x;
68         return;
69     }
70     int m = (l + r) / 2;
71     if (p <= m) {
72         set(id * 2, p, x, l, m);
73     }
74     else {
75         set(id * 2 + 1, p, x, m + 1, r);
76     }
77     data[id] = gcd(data[id * 2], data[id * 2 + 1]);
78 }
79

```




```

80 };
81
82
83 int32_t main() {
84     ios::sync_with_stdio(false);
85     cin.tie(0);
86     cout.tie(0);
87     int n;
88     cin >> n;
89     vector<int> a(n);
90     for (int i = 0; i < n; i++){
91         cin >> a[i];
92     }
93     int q;
94     cin >> q;
95     seg_tree st(a);
96     while (q--) {
97         int l, r;
98         cin >> l >> r;
99         cout << st.get(l, r) << "\n";
100     }
101     return 0;
102 }

```

Фрагмент турнирной таблицы контеста

по ячейкам таблицы для просмотра истории

Положение 		Совпадений: 3   по								
№	Кто	=	Штраф	A	B	C	D	E	F	G
10	Постнов Александр Вячеславович М80-1015-21	3	93	+ 00:06	+ 00:10	+2 00:37				
	* Постнов Александр Вячеславович М80-1015-21	4					+1	+14	+	+1

Вывод

Задача решена.

Декартово дерево [21]

В. Вставка в последовательность

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

Реализуйте структуру данных способную быстро осуществлять вставку элемента на любую позицию.

Входные данные

В первой строке вам дано единственное число N ($0 \leq N \leq 3 \cdot 10^5$) — число запросов вставки. В следующих N строках вам даны запросы в виде пар чисел v_i, p_i ($|v_i| \leq 10^9, 0 \leq p_i \leq i$) — значение и позиция для вставки очередного элемента.

Выходные данные

Выведите полученную последовательность.

Пример

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

Идея решения

В этой задаче я буду использовать такую структуру данных, как Декартово дерево. Вставка элементов по ключу(в данном случае по позиции) имеет сложность $O(\log_2 n)$. Вставка должна использоваться n раз. Итоговая сложность: $O(n * \log_2 n)$

Исходный код

```
1 | #include <bits/stdc++.h>
2 | using namespace std;
3 |
4 | #define int long long
5 |
6 | const unsigned long long INF = 1e9;
7 |
8 | using pii = pair<int, int>;
9 | using graph = vector<vector<int>>>;
10 | using Wgraph = vector<vector<pair<int, int>>>>;
11 |
12 | std::mt19937 rng(std::chrono::steady_clock::now().time_since_epoch().count());
13 |
14 |
15 | struct treap {
16 |     struct node {
17 |         node* left;
18 |         node* right;
19 |         int x;
20 |         int64_t y;
21 |         int data;
22 |         int data_max;
```

```

23     bool reverse;
24
25     node(int num) {
26         right = nullptr;
27         left = nullptr;
28         x = 1;
29         y = rng();
30         data = num;
31         data_max = num;
32         reverse = false;
33     }
34 };
35
36 node* root;
37 treap() {
38     root = nullptr;
39 }
40
41 int get_key(node* t) {
42     if (t != nullptr) {
43         return t->x;
44     }
45     return 0;
46 }
47
48 node* merge(node* t1, node* t2) {
49     if (t1 == nullptr) {
50         return t2;
51     }
52     if (t2 == nullptr) {
53         return t1;
54     }
55     if (t1->y < t2->y) {
56         push(t1);
57         t1->right = merge(t1->right, t2);
58         update(t1);
59         return t1;
60     } else {
61         push(t2);
62         t2->left = merge(t1, t2->left);
63         update(t2);
64         return t2;
65     }
66 }
67
68 int get_max(node* t) {
69     if (t != nullptr) {
70         return t->data_max;
71     }
72     return 0;
73 }
74
75 void update(node* t) {
76     if (t != nullptr) {
77         t->x = 1 + get_key(t->left) + get_key(t->right);
78         t->data_max = max(t->data, max(get_max(t->left), get_max(t->right)));
79     }
80 }
81
82 void push_reverse(node* t) {

```

```

83     if (t != nullptr) {
84         t->reverse ^= 1;
85     }
86 }
87
88 void push(node* t) {
89     if (t != nullptr && t->reverse) {
90         push_reverse(t->left);
91         push_reverse(t->right);
92         swap(t->left, t->right);
93         t->reverse = false;
94     }
95 }
96
97 void print() {
98     print(root, 0);
99 }
100
101 void print(node* t, int h) {
102     if (t != nullptr) {
103         push(t);
104         print(t->left, h + 1);
105         cout << t->data << " ";
106         print(t->right, h + 1);
107     }
108 }
109
110 void split(node* t, int x0, node* & t1, node* & t2) {
111     if (t == nullptr) {
112         t1 = nullptr;
113         t2 = nullptr;
114         return;
115     }
116     push(t);
117     if (x0 > 1 + get_key(t->left)) {
118         split(t->right, x0 - 1 - get_key(t->left), t->right, t2);
119         t1 = t;
120     } else {
121         split(t->left, x0, t1, t->left);
122         t2 = t;
123     }
124     update(t);
125 }
126
127 void insert(int _x, int _p) {
128     node* l = nullptr;
129     node* r = nullptr;
130     split(root, _p, l, r);
131     node* m = new node(_x);
132     root = merge(merge(l, m), r);
133 }
134
135 void reverse_seg(int ql, int qr) {
136     node* l = nullptr;
137     node* m = nullptr;
138     node* r = nullptr;
139     split(root, qr + 1, m, r);
140     split(m, ql, l, m);
141     push_reverse(m);
142     root = merge(l, merge(m, r));

```




```

143     }
144
145     void replace(int ql, int qr, int index) {
146         node* l = nullptr;
147         node* m0 = nullptr;
148         node* m1 = nullptr;
149         node* r = nullptr;
150         split(root, qr + 1, m1, r);
151         split(m1, ql, m0, m1);
152         split(m0, index, l, m0);
153         root = merge(merge(l, m1), merge(m0, r));
154     }
155
156 };
157
158 int32_t main() {
159     ios::sync_with_stdio(false);
160     cin.tie(0);
161     cout.tie(0);
162     treap t;
163     int n;
164     cin >> n;
165     for (int i = 1; i < n + 1; i++){
166         int x, p;
167         cin >> x >> p;
168         p++;
169         t.insert(x, p);
170     }
171     t.print();
172     cout << "\n";
173     return 0;
174 }

```

Фрагмент турнирной таблицы контеста

по ячейкам таблицы для просмотра истории

Положение 		Совпадений: 1   <input type="text" value="пос"/>						
№	Кто	=	Штраф	A	B	C	D	E
7	Постнов Александр Вячеславович М80-1015-21	3	191	+ 00:48	+ 01:22	+ 01:01		

Вывод

Задача решена.

Весенняя олимпиада первого курса

У. да, это обязательно

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

После неудачной стрелки с небезызвестным преподавателем Вы были отчислены и призваны в армию.

Товарищу полковнику понравилось, как Вы красите траву на поле, поэтому он поручил Вам новую боевую задачу — чистку снега! Плац представляет собой прямоугольник n на m метров. Изначально весь плац в снегу. Убрать весь снег сразу довольно трудно, поэтому Вам дают приказы последовательно — чистить по одному квадратному метру снега.

Представим плац как клетчатое поле n на m . Тогда приказ полковника — очистить клетку с координатами x и y . Теперь товарищ полковник решил Вам напомнить, для чего нужен плац — конечно же, для строевой подготовки!

Заниматься строевой подготовкой можно только на очищенных местах. Причём если очищены две соседние клетки, то торжественный марш проходит сразу по обоим клеткам. После каждой очистки снега товарищ полковник интересуется, в скольких местах возможно заниматься строевой подготовкой.

Входные данные

Товарищ полковник показывает Вам t плацев, которые Вам предстоит чистить ($1 \leq t \leq 10^4$).

Сначала товарищ полковник доводит до Вас три целых числа n, m, q ($1 \leq n, m \leq 2 \cdot 10^3, 1 \leq q \leq 2 \cdot 10^5$) — размеры очередного плаца и количество приказов.

Следующие q раз товарищ полковник отдаёт приказ в виде двух целых чисел x, y ($1 \leq x \leq n, 1 \leq y \leq m$) — координаты клетки плаца, которую нужно очистить.

Товарищ полковник не заставляет Вас чистить уже чистую клетку.

Гарантируется, что за весь срок службы Вам придётся чистить снег не более $2 \cdot 10^5$ раз ($\sum q \leq 2 \cdot 10^5$).

Выходные данные

Каждый раз, когда Вы очистили очередной квадратный метр плаца, доложите полковнику количество мест, где можно заниматься строевой подготовкой.

Пример

входные данные	Скопировать
2 5 5 8 3 3 5 3 2 4 2 2 2 1 4 3 1 3 2 3 2 2 4 1 1 2 2 1 2 2 1	
выходные данные	Скопировать
1 2 3 4 4 3 4 1 1 2 2	

Идея решения

Буду использовать СНМ, представлю клетки как вершины графа. Как только очищаю клетку(к итоговому ответу ++), смотрю влево, вправо, вверх, вниз, если клетка тоже очищена и лидер с текущей клеткой разный, то объединяю эти множества и отнимаю от ответа 1. Итоговая сложность:

$$O(t * (n * m + q * \alpha(n * m)))$$

Исходный код

```

1 | #include <bits/stdc++.h>
2 |
3 |
4 |
5 | using namespace std;
6 |
7 | #define int long long
8 |
9 | using graph = vector < vector < pair<int,int> > >;
10 | struct edge {
11 |     int u, v, w;
12 | };
13 |
14 | bool operator < (const edge & lhs, const edge & rhs) {
15 |     return lhs.w < rhs.w;
16 | }
17 |
18 | struct dsu {
19 |     vector <int> leader;
20 |     vector <int> size;
21 |     dsu (int n) {
22 |         leader.resize(n);
23 |         for (int i = 0; i < n; i++){
24 |             leader[i] = i;
25 |         }
26 |         size.resize(n, 1);
27 |     }
28 |     int find(int u) {
29 |         if (leader[u] == u) {
30 |             return u;
31 |         }
32 |         int v = find(leader[u]);
33 |         leader[u] = v;
34 |         return v;
35 |     }
36 |     void join(int u, int v) {
37 |         int lead_u = find(u);
38 |         int lead_v = find(v);
39 |         if (lead_u == lead_v) {
40 |             return ;
41 |         }
42 |         if (size[lead_u] > size[lead_v]) {
43 |             leader[lead_v] = lead_u;
44 |             size[lead_u] += size[lead_v];
45 |         } else {
46 |             leader[lead_u] = lead_v;
47 |             size[lead_v] += size[lead_u];
48 |         }
49 |     }
50 | };
51 |
52 |
53 | const int INF = 1000000000;
54 |
55 | int32_t main(){
56 |     ios::sync_with_stdio(false);

```

```

57 cin.tie(0);
58 cout.tie(0);
59 int t;
60 cin >> t;
61 for (int t0 = 0; t0 < t; t0++) {
62     int n, m;
63     cin >> n >> m;
64     vector <vector <int>> enable(n, vector <int>(m, 0));
65     dsu d(n * m);
66     int ans = 0;
67     int q;
68     cin >> q;
69     for (int i = 0; i < q; i++){
70         int x, y;
71         cin >> x >> y;
72         x--;
73         y--;
74         int key = y * n + x;
75         ans++;
76         enable[x][y] = 1;
77         if (x > 0) {
78             int key1 = x - 1 + y * n;
79             if (enable[x - 1][y] == 1) {
80                 if (d.find(key1) != d.find(key)) {
81                     d.join(key, key1);
82                     ans--;
83                 }
84             }
85         }
86         if (x < n - 1) {
87             int key2 = x + 1 + y * n;
88             if (enable[x + 1][y] == 1) {
89                 if (d.find(key2) != d.find(key)) {
90                     d.join(key, key2);
91                     ans--;
92                 }
93             }
94         }
95         if (y > 0) {
96             int key3 = x + (y - 1) * n;
97             if (enable[x][y - 1] == 1) {
98                 if (d.find(key3) != d.find(key)) {
99                     d.join(key, key3);
100                     ans--;
101                 }
102             }
103         }
104         if (y < m - 1) {
105             int key4 = x + (y + 1) * n;
106             if (enable[x][y + 1] == 1) {
107                 if (d.find(key4) != d.find(key)) {
108                     d.join(key, key4);
109                     ans--;
110                 }
111             }
112         }
113         cout << ans << "\n";
114     }
115 }
116 return 0;

```

Фрагмент турнирной таблицы контеста

Вы можете использовать двойной щелчок (или Ctrl + щелчок) по ячейкам таблицы для просмотра истории

Положение			Совпадений: 2 пос										
№	Кто	=	Штраф	A	B	C	D	E	F	G	H	I	J
11	Постнов Александр Вячеславович M80-1015-21	1	357									-4	+3 04:57
	* Постнов Александр Вячеславович M80-1015-21	3			+						+	+11	

Вывод

В ходе практики изучил классические алгоритмы по обработке строк, графов. Изучил идею динамического программирования, улучшил знания по теории чисел, изучил новые для себя типы данных: дерево отрезков, декартово дерево, префиксное дерево. Подтянул навыки аналитической геометрии. А также улучшил навыки работы с LaTeX, написав отчет на нем. Думаю, что полученные знания мне пригодятся в будущем!