

Журнал практики

Москва, 2022

1 Место и сроки проведения практики

Сроки проведения практики

-дата начала практики 29.06.2022

-дата окончания практики 12.07.2022

Наименование организации:

Московский авиационный институт (национальный исследовательский университет)

Название структурного подразделения (отдел, лаборатория):

кафедра №806 «Вычислительная математика и программирование»

2 Инструктаж по технике безопасности

_____/ Крылов С. С. / 29 июня 2022 г.
(подпись проводившего) (дата проведения)

3 Индивидуальное задание обучающегося

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

Объем практики 108 часов в течение 12 учебных дней.

Руководитель практики от МАИ:

Крылов С. С. / _____ / 29 июня 2022 г.

Руководитель от организации:

_____ / _____ / 29 июня 2022 г.

_____/ Белоусов Е. Л. / 29 июня 2022 г.
(подпись обучающегося) (дата)

4 План выполнения индивидуального задания

№	Тема	Дата
1	Организационное собрание. Выдача задания	29.06.2022
2	Основы C++	30.06.2022
3	Библиотека C++	01.07.2022
4	Динамическое программирование	02.07.2022
5	Префиксные суммы, сортировка событий, два указателя	04.07.2022
6	ДП, задача о рюкзаке	05.07.2022
7	Длинная арифметика	06.07.2022
8	Основы теории графов	07.07.2022
9	Кратчайшие пути во взвешенных графах	08.07.2022
10	Алгоритмы на строках	09.07.2022
11	Оформление журнала с электронным приложением	11.07.2022
12	Защита практики	12.07.2022

_____ / Белоусов Е. Л. / 29 июня 2022 г.
(подпись обучающегося) (дата)

5 Отзыв руководителя практики от организации

Принято участие в 9 конкурсах, прослушаны установочные лекции и разборы задач, решено 39 и доделано 9 задач конкурсов, оформлен журнал практики с электронным приложением. Задание практики выполнено. Рекомендую оценку

Руководитель от организации:

_____/_____/_____/ 12 июля 2022 г.
(подпись) (фамилия, имя, отчество) М.П. (печать)

6 Отчёт обучающегося по практике

Основы C++

Г. Покупки

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

ввод: стандартный ввод
вывод: стандартный вывод

Максим очень любит совершать покупки, и вот настал тот день, когда у него скопилось много разных купюр. Максим долго любовался каждой из них, но всё же решил потратить их на какие-то полезные вещи. Он хочет совершить несколько покупок и при этом продолжить наслаждаться видом купюр в своей копилке, поэтому он хочет оплатить каждую покупку минимальным числом купюр. Он не очень силен в математике, поэтому попросил Вас помочь ему.

У Максима есть купюры номиналами 100, 200, 500, 1000 и 5000 бурлей.

Входные данные

В первой строке вам дано единственное целое число T ($1 \leq T \leq 10$) — число покупок, которое хочет совершить Максим.

В каждой из следующих T строк находится число N ($100 \leq N \leq 1000000$) — стоимость совершаемой покупки в бурлях, при чём N нацело делится на сто.

Выходные данные

Для каждой покупки в отдельной строке выведите пять чисел — количество купюр номиналом 100, 200, 500, 1000 и 5000, которые нужны для оплаты.

Пример

входные данные	Сору
4 600 1700 2800 5900	
выходные данные	Сору
1 0 1 0 0 0 1 1 1 0 1 1 1 2 0 0 2 1 0 1	

Идея решения

Идея задачи довольно проста: чтобы в итоге получить минимальное кол-во купюр, необходимо начать с купюр наибольшего достоинства, постепенно переходя к меньшим. Для каждой купюры мы нацело делим оставшуюся сумму денег на её достоинство, получая нужное кол-во купюр. Затем присваиваем переменной n (текущей сумме бурлей) остаток от деления. Для хранения ответа используем вектор размера 5. Решение работает за $O(t)$, где t — кол-во тестов (начальных сумм денег) во входных данных (так как кол-во различных купюр фиксировано и равно 5).

Исходный код



```
1 | #include <bits/stdc++.h>
2 |
3 | using namespace std;
4 |
5 | int main() {
```

```

6      ios::sync_with_stdio(false);
7      cin.tie(0);
8      int t;
9      cin >> t;
10     for (int i = 0; i < t; ++i) {
11         int n;
12         cin >> n;
13         vector<int> banknotes{5000, 1000, 500, 200, 100};
14         vector<int> bn_nums(5);
15         for (int j = 0; j < 5; ++j) {
16             bn_nums[j] = n / banknotes[j];
17             n %= banknotes[j];
18         }
19         for (int j = 4; j >= 0; --j)
20             cout << bn_nums[j] << " ";
21         cout << '\n';
22     }
23     return 0;
24 }

```

Фрагмент турнирной таблицы контеста

Standings 			Matches: 2  Белоусов												
#	Who	=	Penalty	A	B	C	D	E	F	G	H	I	J	K	L
8	Белоусов Егор Леонидович М8О-108Б-21	9	1177	+ 00:09	+ 00:29	-3	+1 01:27	+2 01:39	+ 01:46	+ 01:59	+ 03:46		+1 02:25	+3 03:37	
	* Белоусов Егор Леонидович М8О-108Б-21	1				+									

Выводы

Задача решена, зачтена чекером с первого раза.

D. База данных

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод

вывод: стандартный вывод

Реализуйте базу данных, которая будет поддерживать следующий набор операций.

- «register *login password*» — регистрирует пользователя с логином «*login*» и паролем «*password*». Если желаемый логин уже занят, то выведите строку «login already in use», в противном случае создайте новый аккаунт и выведите строку «account created».
- «login *login password*» — даёт пользователю доступ к аккаунту «*login*» если указан верный пароль. Если требуемого пользователя не существует либо указанный пароль не верен выведите строку «wrong account info», если пользователь уже вошёл в систему с этим аккаунтом, то выведите строку «already logged in», в противном случае база должна запомнить что пользователь вошёл в систему и вывести строку «logged in».
- «logout *login*» — отбирает у пользователь доступ к аккаунту «*login*». Если требуемого пользователя не существует либо он не вошёл в систему выведите строку «incorrect operation», в противном случае база должна запомнить, что пользователь вышел из системы, и вывести строку «logged out».

Входные данные

Входные данные состоят из набора строк в указанном выше формате. Логин и пароли будут содержать только буквы латинского алфавита в верхнем и нижнем регистре, а так же цифры. Длина логина и паролей не будет превышать 20 символов. Общее количество запросов не будет превышать 10^5 .

Выходные данные

Для каждого запроса в отдельной строке выведите результат его выполнения.

Пример

входные данные	Copy
register vasya password login vasya passwordr login vasya password logout vasya register vasya qwerty register Vasya2017 qwerty logout vasya	
выходные данные	Copy
account created wrong account info logged in logged out login already in use account created incorrect operation	

Идея решения

Интересная задача, позволяющая попрактиковаться ООП. Создадим класс *Database*, единственным полем которого будет ассоциативный массив *data*. Ключами в массиве будут логины пользователей, а значениями — пары, состоящие из пароля и булевого значения "вошёл ли в систему пользователь". Реализуем три метода в классе *Database*: для регистрации нового пользователя, входа и выхода из системы. Осталось дописать в *main*'е код для считывания команд из условия задачи.



Исходный код

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5
6  class Database {
7  private:
8      map<string, pair<string, bool>> data;
9  public:
10     string register_user(string login, string password) {
11         if (data.find(login) != data.end())
12             return "login already in use";
13         data[login] = make_pair(password, false);
14         return "account created";
15     }
16     string login_user(string login, string password) {
17         if (data.find(login) == data.end())
18             return "wrong account info";
19         if (data[login].first != password)
20             return "wrong account info";
21         if (data[login].second)
22             return "already logged in";
23         data[login].second = true;
24         return "logged in";
25     }
26     string logout_user(string login) {
27         if (data.find(login) == data.end())
28             return "incorrect operation";
29         if (!data[login].second)
30             return "incorrect operation";
31         data[login].second = false;
32         return "logged out";
33     }
34 };
35
36
37 int main() {
38     ios::sync_with_stdio(false);
39     cin.tie(0);
40     Database db;
41     string cmd;
42     while (cin >> cmd) {
43         string login;
44         cin >> login;
45         if (cmd == "logout") {
46             cout << db.logout_user(login) << '\n';
47             continue;
48         }
49         string password;
50         cin >> password;
51         if (cmd == "register")
52             cout << db.register_user(login, password) << '\n';
53         else if (cmd == "login")
54             cout << db.login_user(login, password) << '\n';
55     }
56     return 0;
57 }

```


Фрагмент турнирной таблицы контеста

Standings 											Matches: 2  Белоусов				
#	Who	=	Penalty	<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>	<u>E</u>	<u>F</u>	<u>G</u>	<u>H</u>	<u>I</u>	<u>J</u>	<u>K</u>	<u>L</u>
5	Белоусов Егор Леонидович M8O-108Б-21	6	804	+ 01:13	+ 01:19	+ 01:29	+1 02:14	+ 02:38	+2 03:31						-1
	* Белоусов Егор Леонидович M8O-108Б-21	0								-6					

Выводы

Задача решена, зачтена чекером со второго раза, вначале забыл дописать перенос строки в выходных данных.

Динамическое программирование

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	Сору
82	
выходные данные	Сору
202	

Примечание

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

Идея решения

Одна из классических задач на динамическое программирование. В условии нас просят преобразовать число n в единицу, мы же пойдем "с конца", то есть, от единицы к самому числу. Изначально (в единице) стоимость операций равна нулю, т.к. нам не нужно из единицы делать единицу. Заведём вектор dp размера $n + 1$, заполним его константой INF , большей любого возможного числа n . $dp[i]$ — минимальная стоимость преобразования числа i в единицу. Обозначим $dp[1] = 0$. При этом для всех $i > 1$ должно выполняться $dp[i] = i + \min(dp[i - 1], dp[i/2], dp[i/3])$, т.е. мы приходим в число i либо из предыдущего числа $i - 1$, либо из чисел $i/2$, $i/3$ (если i нацело делится на 2 или 3). Заполняем массив dp в цикле от 2 до n включительно. Ответ на задачу — число $dp[n]$. Алгоритм работает за $O(n)$, т.е. за линейное время.

Исходный код

```
1 | #include <bits/stdc++.h>
2 |
3 | using namespace std;
4 |
5 |
6 | int main() {
7 |     ios::sync_with_stdio(false);
8 |     cin.tie(0);
9 |     int n;
10 |    cin >> n;
11 |    const int64_t INF = 1e8;
12 |    vector<int64_t> dp(n + 1, INF);
13 |    dp[1] = 0;
14 |    int i = 2;
```

```
15 while (i <= n) {
16     int64_t min_dp = INF;
17     if (i % 2 == 0)
18         if (dp[i / 2] < min_dp)
19             min_dp = dp[i / 2];
20     if (i % 3 == 0)
21         if (dp[i / 3] < min_dp)
22             min_dp = dp[i / 3];
23     if (dp[i - 1] < min_dp)
24         min_dp = dp[i - 1];
25     dp[i] = min_dp + i;
26     ++i;
27 }
28 cout << dp[n];
29 return 0;
30 }
```

Фрагмент турнирной таблицы контеста

Standings			Matches: 2												
#	Who	=	Penalty	A	B	C	D	E	F	G	H	I	J	K	L
26	Белоусов Егор Леонидович M8O-108Б-21	2	400	+3 01:57	-1	+ 03:43									
	* Белоусов Егор Леонидович M8O-108Б-21	5			+		+3	+2	+	+1					

Выводы

Задача дорешана, зачтена чекером со второго раза (поправил константу *INF* и размер массива *dp*).

Префиксные суммы, сортировка событий, два указателя

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	Сору
3 -2 2 -1 1 0 1	
выходные данные	Сору
5	

входные данные	Сору
4 -10 10 -20 0 -30 0 15 20	
выходные данные	Сору
47	

Идея решения

Представим входные данные (границы отрезков) в виде двух событий: начала и конца отрезка. Каждое событие зададим парой чисел $(i, event)$, где i — индекс события на числовой прямой, $event$ — код события (1 — начало отрезка, 2 — его конец). Причём для конца отрезка индекс будет на единицу больше, т.к. крайняя правая точка отрезка всё ещё входит в этот самый отрезок, а точка на единицу правее — уже нет. Отсортируем массив событий, и далее, идя по нему в цикле, будем с помощью переменной cnt фиксировать, сколько отрезков покрывает текущую точку (и все точки до предыдущего события). Таким образом, вместо того, чтобы идти по всей числовой прямой, границы которой по условию от -10^9 до 10^9 , или, например, по всем точкам внутри отрезков, мы рассматриваем лишь начало и конец каждого отрезка. Сложность алгоритма — $O(2 * n) == O(n)$, где n — кол-во отрезков.

Исходный код



```
1 | #include <bits/stdc++.h>
2 |
3 | using namespace std;
4 |
5 | using pii = pair<int, int>;
```

```

6 | int64_t INF = 1e18;
7 |
8 | int main() {
9 |     ios::sync_with_stdio(false);
10 |    cin.tie(0);
11 |    int n;
12 |    cin >> n;
13 |    vector<pii> events;
14 |    for (int i = 0; i < n; ++i) {
15 |        int l, r;
16 |        cin >> l >> r;
17 |        events.push_back({l, 1});
18 |        events.push_back({r + 1, 2});
19 |    }
20 |    sort(events.begin(), events.end());
21 |    int64_t lp = -INF, cnt = 0, ans = 0;
22 |    for (pii elem: events) {
23 |        int xi = elem.first, event = elem.second;
24 |        if (event == 1)
25 |            ++cnt;
26 |        else
27 |            --cnt;
28 |        if (!cnt) {
29 |            ans += xi - lp;
30 |            lp = -INF;
31 |        }
32 |        else if (lp == -INF)
33 |            lp = xi;
34 |    }
35 |    cout << ans;
36 |    return 0;
37 | }

```

Фрагмент турнирной таблицы контеста

Standings 											Matches: 1  Белоусов				
#	Who	=	Penalty	A	B	C	D	E	F	G	H	I	J	K	L
4	Белоусов Егор Леонидович M8O-1085-21	6	637	+1 00:23	+ 00:36	+1 00:52	+ 01:32	-2	-2		+ 03:10				+ 03:24

Выводы

Задача решена, зачтена чекером с первого раза.

ДП, задача о рюкзаке

Г. 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	Сору
2 2 1 1 1 1	

выходные данные	Сору
1	

входные данные	Сору
2 3 1 1 1 1 1 1	

выходные данные	Сору
2	

входные данные	Сору
2 5 1 1 1 1 1 1 1 1 1 1	

выходные данные	Сору
3	

входные данные	Сору
5 5 1 2 3 4 5 2 3 4 5 1 3 4 5 1 2 4 5 1 2 3 5 1 2 3 4	

выходные данные	Сору
16	

Идея решения

Наиболее трудным шагом в решении было понять, каким образом осуществлять обход поля, т.е. как построить динамику. Я пришёл к выводу, что неплохим вариантом будет идти по диагоналям, т.к. ход конём подразумевает, что нам в точке (i, j) должны быть известны предыдущие данные с точек $(i - 2, j + 1)$, $(i - 2, j - 1)$, $(i - 1, j - 2)$, $(i + 1, j - 2)$. Проблему разных размеров диагоналей в прямоугольниках разных размеров (неудобно обходить диагонали) я решил, выделив в отдельную функцию *get_indexes* получение нужного вектора точек обхода. Функция *get_dp*, получая на вход очередную точку (i, j) , возвращает $dp[i][j]$ — максимальное число золота, которое можно получить, попав в точку (i, j) . Из доступных предыдущих точек (индексы которых не выходят за границы поля) берётся максимум, и добавляется значение золота в текущей точке (i, j) . Ответом является максимальное значение $dp[i][j]$ для любых i, j . Алгоритм работает за $O((\max(n, m))^2)$, т.к. внешний цикл идёт до $n + m - 1$, внутренний — до текущего значения счётчика внешнего цикла k : $1 + 2 + \dots + (n + m - 1) = ((n + m - 1) * (n + m)) / 2$.

Исходный код

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 int get_dp(vector<vector<int>> &dp, vector<vector<int>> &field, int i, int j, int n, int m) {
6     vector<pair<int, int>> moves{{-2, 1}, {-2, -1}, {-1, -2}, {1, -2}};
7     int res = -1;
8     for (int k = 0; k < 4; ++k) {
9         pair<int, int> move = moves[k];
10        int res_i = move.first + i, res_j = move.second + j;
11        if (res_i >= 0 && res_i < n && res_j >= 0 && res_j < m) {
12            if (dp[res_i][res_j] == -1)
13                continue;
14            res = max(res, dp[res_i][res_j]);
15        }
16    }
17    return res != -1 ? res + field[i][j] : -1;
18 }
19
20 vector<pair<int, int>> get_indexes(int len, int n, int m) {
21     vector<pair<int, int>> res;
22     for (int i = 0; i <= len; ++i) {
23         int x = i, y = len - i;
24         if (x < n && y < m)
25             res.push_back({x, y});
26     }
27     return res;
28 }
29
30
31 int main() {
32     ios::sync_with_stdio(false);
33     cin.tie(0);
34     int n, m;
35     cin >> n >> m;
36     vector<vector<int>> field(n, vector<int>(m));
37     for (int i = 0; i < n; ++i)
38         for (int j = 0; j < m; ++j)
39             cin >> field[i][j];
40     vector<vector<int>> dp(n, vector<int>(m, -1));
```

```

41 |     dp[0][0] = field[0][0];
42 |     int max_gold = dp[0][0];
43 |     for (int k = 1; k < n + m - 1; ++k) {
44 |         for (auto ind: get_indexes(k, n, m)) {
45 |             int i = ind.first, j = ind.second;
46 |             dp[i][j] = get_dp(dp, field, i, j, n, m);
47 |             max_gold = max(max_gold, dp[i][j]);
48 |         }
49 |     }
50 |     cout << max_gold << '\n';
51 |     return 0;
52 | }

```

Фрагмент турнирной таблицы контеста

Standings				Matches: 1 Белоусов									
#	Who	=	Penalty	A	B	C	D	E	F	G	H	I	J
3	Белоусов Егор Леонидович M8O-108Б-21	4	743	+ 01:38	+ 02:22				+2 04:46	+ 02:57			

Выводы

Задача решена, зачтена чекером с третьего раза (поправил мелкие недочёты: \leq вместо $<$, объявление переменной $res = -1$ вместо $res = 0$).

Длинная арифметика

А. Быстрое преобразование Фурье

ограничение по времени на тест: 2 секунды
ограничение по памяти на тест: 256 мегабайт

ввод: стандартный ввод
вывод: стандартный вывод

Дан многочлен $A(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + a_3 \cdot x^3 + \dots + a_{n-1} \cdot x^{n-1} + a_n \cdot x^n$.

Вычислите для него быстрое преобразование Фурье.

Входные данные

Первая строка содержит целое число n ($1 \leq n \leq 2 \cdot 10^5$) — степень многочлена $A(x)$.

Вторая строка содержит n целых чисел a_i ($|a_i| \leq 9$) — коэффициенты многочлена $A(x)$.

Выходные данные

Выведите $2^{\lceil \log_2 n \rceil}$ (округление n вверх до ближайшей степени двойки) пар чисел — вещественную и мнимую часть преобразования Фурья многочлена $A(x)$.

Ваш ответ будет считаться правильным, если его абсолютная или относительная ошибка не превосходит 10^{-6} .

Примеры

входные данные	Сору
1 1	
выходные данные	Сору
1.0000000000 0.0000000000	

входные данные	Сору
1 0	
выходные данные	Сору
0.0000000000 0.0000000000	

входные данные	Сору
4 1 2 3 4	
выходные данные	Сору
10.0000000000 0.0000000000 -2.0000000000 -2.0000000000 -2.0000000000 0.0000000000 -2.0000000000 2.0000000000	

входные данные	Сору
6 1 2 3 4 5 6	
выходные данные	Сору
21.0000000000 0.0000000000 -9.6568542495 3.0000000000 3.0000000000 4.0000000000 1.6568542495 -3.0000000000 -3.0000000000 0.0000000000 1.6568542495 3.0000000000 3.0000000000 -4.0000000000 -9.6568542495 -3.0000000000	

Примечание

В четвёртом тестовом случае 6 округляется до $2^3 = 8$.

Идея решения



Быстрым преобразованием Фурье (БПФ, FFT) называется алгоритм вычисления дискретного преобразования Фурье (ДПФ), позволяющий найти его коэффициенты за $O(n * \log(n))$, что существенно быстрее по сравнению с вычислением напрямую, по формуле, за $O(n^2)$. Чтобы ускорить вычисления, применяется принцип "разделяй-и-властвуй", по которому коэффициенты исходного многочлена разбиваются на две группы: чётные и нечётные. От каждой группы считается DFT, и в результате находится DFT исходного многочлена. Используя свойства коэффициентов преобразования Фурье, мы сводим задачу к двум подзадачам вдвое меньшего размера.

Исходный код

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5
6  void fft(vector<complex<double>> &a) {
7      int n = pow(2, ceil(log2(a.size()))), n_old = a.size();
8      for (int i = 0; i < n - n_old; ++i)
9          a.push_back(complex<double>(0));
10     if (n == 1)
11         return;
12     vector<complex<double>> a0(n / 2), a1(n / 2);
13     for (int i = 0, j = 0; j < n; ++i, j += 2) {
14         a0[i] = a[j];
15         a1[i] = a[j + 1];
16     }
17     fft(a0);
18     fft(a1);
19     double phi = 2.0 * acos(-1) / n;
20     complex<double> w(1), wn(cos(phi), sin(phi));
21     for (int i = 0; i < n / 2; ++i) {
22         a[i] = a0[i] + w * a1[i];
23         a[n / 2 + i] = a0[i] - w * a1[i];
24         w *= wn;
25     }
26 }
27
28
29 int main() {
30     ios::sync_with_stdio(false);
31     cin.tie(0);
32     int n;
33     cin >> n;
34     vector<complex<double>> vc;
35     for (int i = 0; i < n; ++i) {
36         complex<double> c;
37         cin >> c;
38         vc.push_back(c);
39     }
40     fft(vc);
41     stringstream ss;
42     ss.setf(ios::fixed);
43     ss.precision(10);
44     for (auto c: vc)
45         ss << c.real() << " " << c.imag() << '\n';
46     cout << ss.str();
```

```
47 ||      return 0;
48 ||  }
```

Фрагмент турнирной таблицы контеста

Standings 		Matches: 1  <input type="text" value="Белоусов"/>						
#	Who	=	Penalty	A	B	C	D	E
6	Белоусов Егор Леонидович M8O-108Б-21	2	159	+1 00:44	+ 01:35			

Выводы

Задача решена, зачтена чекером с третьего раза (исправил ошибку с числом пи, поправил точность вывода).

Основы теории графов

В. Поиск в ширину

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

ввод: стандартный ввод
вывод: стандартный вывод

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

Входные данные

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

Выходные данные

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

Примеры

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

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

Идея решения



Поиск в ширину (BFS, breadth-first search) является одним из методов обхода графа. Он работает, перебирая все вершины, выходящие из текущей, и добавляя их в очередь (queue). Далее для каждой вершины в очереди процесс повторяется, и так далее. Таким образом, граф делится на определённые "уровни", т.е. минимальные расстояния от исходной вершины до любой другой в графе. Сложность — $O(n + m)$, где n — кол-во вершин в графе, а m — кол-во рёбер.

Исходный код

```
1 | #include <bits/stdc++.h>
2 |
3 | using namespace std;
4 |
5 | using graph = vector<vector<int>>>;
6 | const int INF = 1e9;
7 |
8 | vector<int> bfs(int start, const graph &g) {
9 |     int n = g.size();
10 |    vector<int> d(n, INF);
```

```
11 | d[start] = 0;
12 | queue<int> q;
13 | q.push(start);
14 | while (!q.empty()) {
15 |     int u = q.front();
16 |     q.pop();
17 |     for (int v: g[u]) {
18 |         if (d[v] == INF) {
19 |             d[v] = d[u] + 1;
20 |             q.push(v);
21 |         }
22 |     }
23 | }
24 | return d;
25 | }
26 |
27 | int main() {
28 |     ios::sync_with_stdio(false);
29 |     cin.tie(0);
30 |     int n, m, k;
31 |     cin >> n >> m >> k;
32 |     graph g(n);
33 |     for (int i = 0; i < m; ++i) {
34 |         int u, v;
35 |         cin >> u >> v;
36 |         --u;
37 |         --v;
38 |         g[u].push_back(v);
39 |         g[v].push_back(u);
40 |     }
41 |     for (int d: bfs(k - 1, g))
42 |         if (d == INF)
43 |             cout << "-1 ";
44 |     else
45 |         cout << d << " ";
46 |     return 0;
47 | }
```

Фрагмент турнирной таблицы контеста

Standings 							Matches: 2  Белоусов				
#	Who	=	Penalty	A	B	C	D	E	F	G	H
11	Белоусов Егор Леонидович М8О-108Б-21	3	615	+2 02:28	+ 02:40	-4	-5	+1 04:07			
	* Белоусов Егор Леонидович М8О-108Б-21	0				-1					

Выводы

Задача решена, зачтена чекером с первого раза.

Кратчайшие пути во взвешенных графах

В. Алгоритм Флойда — Уоршелла

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

ввод: стандартный ввод
вывод: стандартный вывод

Задан неориентированный взвешенный граф, вершины которого пронумерованы от 1 до n . Ваша задача найти длины кратчайших путей между всеми парами вершин.

Входные данные

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

Выходные данные

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

Примеры

входные данные	Скопировать
1 0	
выходные данные	Скопировать
0	

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

Идея решения

Алгоритм Флойда-Уоршелла — это алгоритм поиска кратчайших путей во взвешенном графе, в котором могут быть рёбра как с положительным, так и с отрицательным весом (но не должно быть отрицательных циклов). Он работает за $O(n^3)$, где n — число вершин в графе, и основывается на ограничении понятия кратчайшего пути из u в v множеством $\{1..i\}$. Обозначим $d_{uv}^{(i)}$ длину кратчайшего пути из вершины u в вершину v , содержащего, кроме начала и конца, только вершины из мн-ва $\{1..i\}$. При этом $d_{uv}^{(0)}$ равно длине ребра между u и v , если оно существует в графе, и ∞ , если нет. На каждом шаге алгоритма будем брать очередную вершину и для всех пар u и v вычислять значение $d_{uv}^{(i)} = \min(d_{uv}^{(i-1)}, d_{ui}^{(i-1)} + d_{iv}^{(i-1)})$. Эта формула говорит о том, что если вершина i содержится в кратчайшем пути из u в v , то таким путём является путь из u в i , объединённый с путём из i в v . Иначе, таким путём является кратчайший путь из u в v , содержащий лишь вершины из мн-ва $\{1..i-1\}$.

Исходный код



```
1 | #include <bits/stdc++.h>
2 |
3 | using namespace std;
4 |
```

```

5 | const int INF = 1e9;
6 |
7 | int main() {
8 |     ios::sync_with_stdio(false);
9 |     cin.tie(0);
10 |    int n;
11 |    cin >> n;
12 |    vector<vector<int>> d(n, vector<int>(n, INF));
13 |    for (int i = 0; i < n; ++i) {
14 |        for (int j = 0; j < n; ++j) {
15 |            int w;
16 |            cin >> w;
17 |            d[i][j] = w;
18 |        }
19 |    }
20 |    for (int i = 0; i < n; ++i)
21 |        d[i][i] = 0;
22 |    for (int k = 0; k < n; ++k)
23 |        for (int i = 0; i < n; ++i)
24 |            for (int j = 0; j < n; ++j)
25 |                d[i][j] = min(d[i][j], d[i][k] + d[k][j]);
26 |    for (int i = 0; i < n; ++i) {
27 |        for (int j = 0; j < n; ++j)
28 |            cout << d[i][j] << " ";
29 |        cout << endl;
30 |    }
31 |    return 0;
32 | }

```

Фрагмент турнирной таблицы контеста

Standings 				Matches: 2 								Белоусов	
#	Who	=	Penalty	A	B	C	D	E	F	G	H		
13	Белоусов Егор Леонидович М8О-108Б-21	4	854	+ 03:02	+ 03:21	+ 03:55	+ 03:56		-2				
	* Белоусов Егор Леонидович М8О-108Б-21	2						+1	+				

Выводы

Задача решена, зачтена чекером с первого раза.

Алгоритмы на строках

А. Z-функция

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

ввод: стандартный ввод
вывод: стандартный вывод

Выведите z -функцию для заданной строки.

Входные данные

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

Выходные данные

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

Примеры

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

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

Идея решения

Z -функцией строки S и позиции x называется длина максимального префикса подстроки, начинающейся с позиции x в строке S , который одновременно является и префиксом всей строки S . Значение Z -функции от первого символа в строке не определено, поэтому его обычно приравнивают либо к длине строки, либо к нулю. Существует тривиальный алгоритм поиска Z -функции, работающий за $O(n^2)$, где n — длина строки. Он представляет собой простой перебор всех индексов и префиксов с нахождением максимального для каждого индекса. Более эффективный алгоритм работает за $O(n)$ и представлен в решении этой задачи. Обозначим l и r — границы самого правого отрезка совпадения, причём r в отрезок не входит, т.е. $s[l, r) = s[0, z[i])$. Изначально $l = r = 0$. Пусть известны значения Z -функции от 0 до $i - 1$. Найдём $z[i]$. Рассмотрим 2 случая:

1. $i > r$:

В этом случае просто "наивно" пробегаемся по строке S , сравнивая символы в позициях $s[i + j]$ и $s[j]$. Первая такая позиция j , для которой не выполняется $s[i + j] = s[j]$, и является значением $z[i]$.

2. $i \leq r$:

Сравним $z[i - l] + i$ и r . Если r меньше, то "наивно" пробегаемся по строке, начиная с r , и вычисляем $z[i]$. В противном случае нам уже известно значение $z[i]$, равное $z[i - l]$.

Исходный код

```
1 #include <bits/stdc++.h>
2
```





```

3 using namespace std;
4
5 vector<int> z_func(const string &s) {
6     int n = s.size();
7     vector<int> z(n);
8     z[0] = n;
9     int l = 0, r = 0;
10    for (int i = 1; i < n; ++i) {
11        if (i <= r)
12            z[i] = min(z[i - l], r - i + 1);
13        while (i + z[i] < n && s[z[i]] == s[i + z[i]])
14            ++z[i];
15        if (i + z[i] >= r) {
16            l = i;
17            r = i + z[i] - 1;
18        }
19    }
20    return z;
21 }
22
23 int main() {
24     ios::sync_with_stdio(false);
25     cin.tie(0);
26     string s;
27     cin >> s;
28     for (int elem : z_func(s))
29         cout << elem << " ";
30     return 0;
31 }

```

Фрагмент турнирной таблицы контеста

Standings 										Matches: 2  Белоусов				
#	Who	=	Penalty	A	B	C	D	E	F	G	H	I	J	
9	Белоусов Егор Леонидович М8О-108Б-21	3	593	+ 02:00	+ 02:20	-7	-1		-1				+3 04:33	
	* Белоусов Егор Леонидович М8О-108Б-21	1				+								

Выводы

Задача решена, зачтена чекером с первого раза.