Минобрнауки России

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Факультет компьютерных и физико-математических наук

Кафедра прикладной математики и информатики

Отчёт о прохождении

учебной практики

02.03.02 Фундаментальная информатика   
и информационные технологии

Выполнил: студент 1 курса   
`

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель: к. п. н., доцент Котельникова А. В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Киров

2019 г.

**Содержание**

[Введение 3](#_Toc387323852)

[Список задач 4](#_Toc387323853)

[Описание решенных задач 7](#_Toc387323854)

[Список консультаций 28](#_Toc387323855)

[Заключение 29](#_Toc387323856)

[Список литературы 30](#_Toc387323857)

[Приложения 31](#_Toc387323858)

Введение

Учебная практика проходила в ФГБОУ ВО «Вятский государственный университет», на кафедре прикладной математики и информатики факультета компьютерных и физико-математических наук с 04.02.2019 г. по 28.04.2019 г., количество недель: 12. Установочная конференция состоялась 04.02.2019 г.

Целью учебной практики является закрепление и углубление теоретической подготовки, приобретение практических навыков и компетенций в сфере профессиональной деятельности, в основном путём самостоятельного решения предусмотренных программой задач.

Форма проведения учебной практики: компьютерная (практикум по решению задач на ЭВМ).

В ходе прохождения практики необходимо реализовать компьютерные программы на языке программирования С++. Каждая задача сначала должна быть принята системой (acmp.ru), а потом нужно защитить предложенный алгоритм решения задачи и его программную реализацию в часы консультаций.

Для получения доступа к архиву задач была пройдена регистрация на сайте acmp.ru, регистрационное имя: Савин Даниил Александрович.

Для реализации задач использовалась среда программирования Microsoft Visual C++.

# Список задач

Индивидуальный список задач из архива задач с сайта acmp.ru, полученный на установочной конференции:

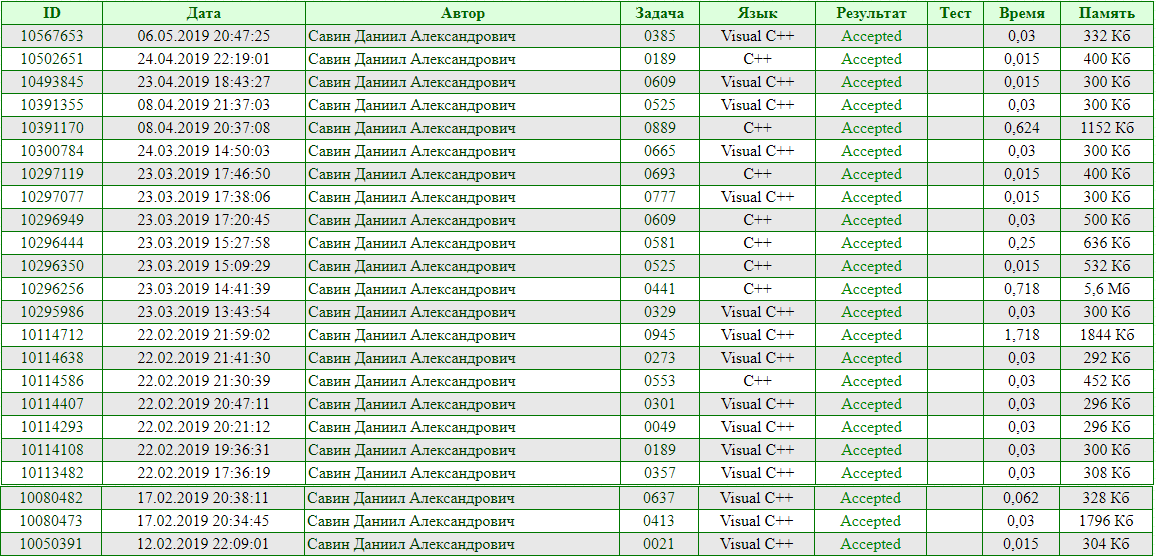
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № п/п | ID | Название | Тема | Разбор | Сложность | Решаемость |
|  | 0021 | Зарплата | Задачи для начинающих | Да | 4% | 82% |
|  | 0049 | Шаблоны | Комбинаторика | Нет | 35% | 80% |
|  | 0077 | Нолики | Комбинаторика | Нет | 63% | 37% |
|  | 0105 | Раскопки | Рекурсия, перебор | Нет | 76% | 24% |
|  | 0133 | Заправки | Теория графов | Нет | 49% | 90% |
|  | 0161 | Восстановление перестановки | Сортировка и последовательности | Нет | 44% | 90% |
|  | 0189 | Перестановка по номеру | Комбинаторика | Нет | 47% | 77% |
|  | 0217 | Еловая аллея | Динамическое программирование | Нет | 47% | 87% |
|  | 0245 | Сплоченная команда | Сортировка и последовательности | Нет | 42% | 56% |
|  | 0273 | Вычеркивание | Рекурсия, перебор | Нет | 28% | 90% |
|  | 0301 | Код | Жадный алгоритм | Нет | 36% | 88% |
|  | 0329 | Лесенка-2 | Динамическое программирование | Нет | 37% | 81% |
|  | 0357 | Делимость на 11 | Целочисленная арифметика | Нет | 22% | 87% |
|  | 0385 | Развлечения с измерителем | Структуры данных | Нет | 28% | 87% |
|  | 0413 | [Военная база](https://acmp.ru/index.asp?main=task&id_task=413) | Двумерные массивы | Нет | 32% | 90% |
|  | 0441 | [Карточки - 2](https://acmp.ru/index.asp?main=task&id_task=441) | Структуры данных | Нет | 73% | 80% |
|  | 0469 | [Химическая тревога](https://acmp.ru/index.asp?main=task&id_task=469) | Теория графов | Нет | 50% | 92% |
|  | 0497 | [Индикатор](https://acmp.ru/index.asp?main=task&id_task=497) | Жадный алгоритм | Нет | 52% | 91% |
|  | 0525 | [Сумма степеней двойки](https://acmp.ru/index.asp?main=task&id_task=525) | Динамическое программирование | Нет | 43% | 89% |
|  | 0553 | [Объединение блоков](https://acmp.ru/index.asp?main=task&id_task=553) | Динамическое программирование | Нет | 46% | 89% |
|  | 0581 | [Атака летающих тарелок](https://acmp.ru/index.asp?main=task&id_task=581) | Геометрия | Нет | 36% | 85% |
|  | 0609 | [Разбиения множества](https://acmp.ru/index.asp?main=task&id_task=609) | Математическое моделирование | Нет | 69% | 80% |
|  | 0637 | [NEERC](https://acmp.ru/index.asp?main=task&id_task=637) | Задачи для начинающих | Нет | 17% | 95% |
|  | 0665 | [Палиндромное время](https://acmp.ru/index.asp?main=task&id_task=665) | Математическое моделирование | Нет | 24% | 81% |
|  | 0693 | [Перестановка](https://acmp.ru/index.asp?main=task&id_task=693) | Разбор строк | Нет | 24% | 79% |
|  | 0721 | [Стрелки - 2](https://acmp.ru/index.asp?main=task&id_task=721) | Теория графов | Нет | 68% | 86% |
|  | 0749 | [Неправильный RSA](https://acmp.ru/index.asp?main=task&id_task=749) | Целочисленная арифметика | Нет | 64% | 83% |
|  | 0777 | [Будильник](https://acmp.ru/index.asp?main=task&id_task=777) | Задачи для начинающих | Нет | 12% | 89% |
|  | 0805 | [Атака инопланетян](https://acmp.ru/index.asp?main=task&id_task=805) | Геометрия | Нет | 36% | 68% |
|  | 0833 | [Преобразование ДНК](https://acmp.ru/index.asp?main=task&id_task=833) | Разбор строк | Нет | 36% | 81% |
|  | 0861 | [Осколки](https://acmp.ru/index.asp?main=task&id_task=861) | Целочисленная арифметика | Нет | 88% | 30% |
|  | 0889 | [Паутина](https://acmp.ru/index.asp?main=task&id_task=889) | Сортировка и последовательности | Нет | 38% | 86% |
|  | 0917 | [Забор - 3](https://acmp.ru/index.asp?main=task&id_task=917) | Геометрия | Нет | 72% | 85% |
|  | 0945 | [Баллы](https://acmp.ru/index.asp?main=task&id_task=945) | Структуры данных | Нет | 45% | 79% |
|  | 0973 | [Главные дороги](https://acmp.ru/index.asp?main=task&id_task=973) | Теория графов | Нет | 82% | 81% |

# Описание решенных задач

В ходе прохождения практики мною были решены следующие задачи: 21, 49, 189, 273, 301, 329, 357, 385, 413, 441, 525, 553, 581, 609, 637, 665, 693, 777, 889, 945 всего 20 задач. Каждая задача была реализована на языке программирования С++.

Детально будут описаны наиболее интересные с точки зрения решения и тестирования решения задачи 0273 и 0637.

На следующем скриншоте показаны принятые системой задачи.



1. ***0021 Зарплата***  
   *(Время: 1 сек. Память: 16 Мб Сложность: 4%)*

В отделе работают 3 сотрудника, которые получают заработную плату в рублях. Требуется определить: на сколько зарплата самого высокооплачиваемого из них отличается от самого низкооплачиваемого.

**Входные данные**

В единственной строке входного файла INPUT.TXT записаны размеры зарплат всех сотрудников через пробел. Каждая заработная плата – это натуральное число, не превышающее 105.

**Выходные данные**

В выходной файл OUTPUT.TXT необходимо вывести одно целое число — разницу между максимальной и минимальной зарплатой.  
Выходные данные   
В выходной файл OUTPUT.TXT необходимо вывести одно целое число — разницу между максимальной и минимальной зарплатой.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 100 500 1000 | 900 |
| 36 11 20 | 25 |

***Описание решения:*** Для решения задачи необходимо считать три числа и вывести на экран разницу максимального и минимального чисел. Используются целочисленный массив.

Листинг программы приведен в Приложении 1.

1. ***0049 Шаблоны***  
   *(Время: 1 сек. Память: 16 Мб Сложность: 35%)*

Шаблоном размера n назовем строку длины n, каждый из символов которой входит в множество {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, g, ?}. Шаблоны преобразуются в строки из цифр по следующим правилам:

• символы от 0 до 9 могут быть преобразованы только сами в себя;

• символ a может преобразован в любой из символов 0,1, 2, 3;

• символ b может преобразован в любой из символов 1,2,3,4;

• символ c может преобразован в любой из символов 2,3,4,5;

• символ d может преобразован в любой из символов 3,4,5,6;

• символ e может преобразован в любой из символов 4,5,6,7;

• символ f может преобразован в любой из символов 5,6,7,8;

• символ g может преобразован в любой из символов 6,7,8,9;

• символ ? может преобразован в любой из символов от 0 до 9;

Даны два шаблона: p1 и p2. Рассмотрим множество S1 строк, которые могут быть получены из p1 по описанным правилам, и множество S2 строк, которые могут быть получены из p2. Необходимо найти количество строк, входящих в оба этих множества.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит шаблон p1, вторая — шаблон p2. Шаблоны имеют одинаковый положительный размер, не больше 9.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите ответ на задачу.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| ???  abc | 64 |
| ??? 000 | 1 |
| abc 999 | 0 |

***Описание решения:*** Для каждой позиции находим, сколько цифр может быть в этой позиции. Эти значения для разных позиций перемножаем.

Листинг программы приведен в Приложении 2.

1. ***0189 Перестановка по номеру***  
   *(Время: 1 сек. Память: 16 Мб Сложность: 47%)*

Перестановкой из N элементов называется упорядоченный набор из N различных чисел от 1 до N. Количество всех перестановок порядка N равно PN = N!

Требуется найти перестановку по ее номеру в лексикографическом порядке (по алфавиту). Например, для N=3 лексикографический порядок перестановок выглядит следующим образом:

(1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1).

Таким образом, перестановка (2,3,1) имеет номер 4 в этой последовательности.

**Входные данные**

В первой строке входного файла INPUT.TXT записано число N (1 ≤ N ≤ 12) - количество элементов в перестановке, во второй - число K (1 ≤ K ≤ N!) - номер перестановки.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите через пробел N чисел - искомую перестановку.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 1 1 | 1 |
| 3 2 | 1 3 2 |

***Описание решения:*** Для решения этой задачи используем факториал.

Листинг программы приведен в Приложении 3.

1. ***0273 Вычеркивание***

*(Время: 1 сек. Память: 16 Мб Сложность: 28%)*

Задано натуральное число N. Требуется написать программу, вычисляющую количество различных трехзначных чисел получающихся из N вычеркиванием цифр из его десятичной записи.

**Входные данные**

Входной текстовый файл INPUT.TXT содержит одно натуральное число N (1 ≤ N ≤ 10100).

**Выходные данные**

Выходной текстовый файл OUTPUT.TXT должен содержать одно целое число - найденное количество трехзначных чисел.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 12 | 0 |
| 111111111110011111111 | 4 |

***Описание решения:*** Считываем данные в строку «buffer» размером 101+1 (+1) – под код нуля (чем заканчивается строка). Создаем строку string s(buffer), чтобы было удобнее работать, создаем вектор «vector<bool> was(1000, false)» размером 1000, значения «false». Дальше бежим: первая циферка от нуля до длины строки – «for (int i = 0; i < (int)s.size(); i++)», вторая циферка – «for (int j = i + 1; j < (int)s.size(); j++)» (j=i+1 – т.к 2 циферка должна быть правее 1), третья циферка – «for (int k = j + 1; k < (int)s.size(); k++)» (k=j+1 – т.к 3 циферка должна быть правее 2). Дальше находим число: (s[i] - '0') – вычитаем код нуля и умножаем на 100, чтобы получить сотни; затем прибавляем десятки: «s[j] - '0') \* 10», и единицы «(s[k] - '0')». Три цифры комбинируем в число и отмечаем в логическом массиве размера 1000, что такое число есть. Считаем сколько чисел от 100 до 999 в логическом массиве отмечено.

Листинг программы приведен в Приложении 4.

1. ***0301 Код***

*(Время: 1 сек. Память: 16 Мб Сложность: 36%)*

В наши дни в космосе находятся сотни спутников, и все они обмениваются данными. При этом система распознавания сигналов работает по схеме «Свой-Чужой». Один из спутников отправляет запрос другому спутнику в формате двух целых чисел, а второй спутник отвечает первому так же двумя целыми числами. Первые два числа первого спутника представляют собой сумму цифр и количество цифр тех двух чисел, которыми должен ответить второй спутник. При этом в качестве ответа должны получиться числа, представляющие наибольшее и наименьшее возможные значения, которые могут быть сформированы по описанному выше методу.

Вам предстоит написать программу, формирующую ответ для второго спутника по известным числам, полученным от первого спутника.

**Входные данные**

Во входном файле INPUT.TXT записаны 2 натуральных числа S и K, представляющих сумму и количество цифр соответственно (K ≤ 100). При этом гарантируется, что возможно составить хотя бы одно K-значное число, сумма цифр которого равна S.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите два числа – ответ второго спутника. При этом следует помнить, что все числа не имеют лидирующих нулей.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 1 3 | 100 100 |
| 2 3 | 200 101 |
| 3 4 | 3000 1002 |

***Описание решения:*** Макс число:

- первую цифру делаем как можно больше.

- вторую цифру делаем как можно больше и т.д.

При этом не позволяем сумме цифр вылезти за заданную сумму.

Мин число:

Не может начинаться на 0.

Чтобы получить в начале цифры поменьше, нужно в конце сделать цифры побольше.

Листинг программы приведен в Приложении 5.

1. ***0329 Лесенка-2***

*(Время: 1 сек. Память: 16 Мб Сложность: 37%)*

Вова стоит перед лесенкой из N ступеней. На каждой из ступеней написаны произвольные целые числа. Первым шагом Вова может перейти на первую ступень или, перепрыгнув через первую, сразу оказаться на второй. Также он поступает и дальше, пока не достигнет N-ой ступени. Посчитаем сумму всех чисел, написанных на ступенях через которые прошел Вова.

Требуется написать программу, которая определит оптимальный маршрут Вовы, при котором, шагая, он получит наибольшую сумму.

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке натуральное число N – количество ступеней лестницы (2 ≤ N ≤ 1000). Во второй строке через пробел заданы числа, написанные на ступенях лестницы, начиная с первой. Числа, написанные на ступенях, не превосходят по модулю 1000.

**Выходные данные**

Выходной файл OUTPUT.TXT должен содержать в первой строке наибольшее значение суммы. Во второй строке должны быть записаны через пробел номера ступеней по возрастанию, по которым должен шагать Вова. Если существует несколько различных правильных маршрутов, то можно вывести любой из них.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 3 1 2 1 | 4 1 2 3 |
| 3 1 -1 1 | 2 1 3 |

***Описание решения:*** maxSum[i] – максимальная сумма, которую можно набрать, дойдя (снизу) до i-той ступеньки.

maxSum[0]=0

maxSum[1]=a[1]

maxSum[2]=a[2]+max(maxSum[1], maxSum[0]).

Тогда мы можем вычислисть сумму по формуле:

maxSum[i]=a[i]+max(maxSum[i-1], maxSum[i-2]), ‘a’ индексируется с 1.

Листинг программы приведен в Приложении 6.

1. ***0357 Делимость на 11***

*(Время: 1 сек. Память: 16 Мб Сложность: 22%)*

Для делимости числа на 11 необходимо, чтобы разность между суммой цифр, стоящих на четных местах, и суммой цифр, стоящих на нечетных местах, делилась на 11.

Требуется написать программу, которая проверит делимость заданного числа на 11.

**Входные данные**

Входной файл INPUT.TXT содержит одно натуральное число N, делимость которого надо проверить (1 ≤ N ≤ 1010000).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите “YES”, если число делится на 11, или “NO” иначе.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 121 | YES |
| 1211 | NO |

***Описание решения:*** Проверяем делится ли число на 11 и выводим ответ: «Да» или «Нет».

Листинг программы приведен в Приложении 7.

1. ***0385 Развлечения с измерителем***

*(Время: 1 сек. Память: 16 Мб Сложность: 32%)*

Дима обнаружил у папы на столе специальный чертежный прибор, похожий на циркуль-измеритель. Измеритель отличается от обычного циркуля тем, что в обеих его ножках находятся иголки (у обычного циркуля в одной ножке находится иголка, а в другой – грифель).

Кроме измерителя Дима нашел на столе клетчатый лист бумаги, в углах некоторых клеток которого были нарисованы точки. Так как измеритель служит для измерения расстояний, то Дима решил измерить все попарные расстояния между всеми точками на листе бумаги.

Ваша задача - написать программу, которая по координатам точек определит, сколько различных расстояний встречается среди расстояний, которые измерил Дима.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит число n – количество точек (2 ≤ n ≤ 50). Следующие n строк содержат по два целых числа – координаты точек. Координаты не превышают 104 по абсолютной величине.

**Выходные данные**

На первой строке выходного файла OUTPUT.TXT выведите k – количество различных расстояний, которые измерил Дима. Следующие k строк должны содержать по одному вещественному числу – сами расстояния. Расстояния должны быть выведены в возрастающем порядке. Каждое число должно быть выведено с точностью не менее чем 10-9.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 4 0 0 1 1 1 0 0 1 | 2 1.0 1.414213562373 |

***Описание решения:*** Считываем данные, а затем попарно проверяем все точки и записываем различные расстояния в массив, сортируем массив и выводим его значения.

Листинг программы приведен в Приложении 8.

1. ***0413 Военная база***

*(Время: 1 сек. Память: 16 Мб Сложность: 32%)*

Со спутника-шпиона получено изображение в некотором волновом диапазоне сверхсекретной военной базы предполагаемого противника. База расположена на Антарктиде, все постройки на ней высечены из кубов льда и имеют на фотографии квадратную форму и не имеют общих фрагментов стен ненулевой длины (по всей видимости, это сделано в целях маскировки от локаторов, работающих в инфракрасном спектре). Благодаря мастерству операторов оказалось, что стены разных построек параллельны границам фотографии.

Для того, чтобы составить сверхсрочный отчет для командования, необходимо узнать, сколько зданий находятся на базе. Напишите программу, которая это сделает.

**Входные данные**

В первой строке входного файла INPUT.TXT записаны числа N и M (1 ≤ M, N ≤ 500) – размеры фотографии в пикселях по вертикали и по горизонтали. Следующие N строк содержат по M символов каждая: символ '.' соответствует пустому месту, '#' – элементу постройки.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите единственное число – количество построек на базе.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 8 6 ...... ...##. ...##. ...... .###.. .###.. .###.. ...... | 2 |

***Описание решения:*** Считываем 2 строки и сравниваем есть ли у нас точка над и перед первой «#». Считаем сколько таких уголков и выводим это количество.

Листинг программы приведен в Приложении 9.

1. ***0441 Карточки-2***

*(Время: 1 сек. Память: 128 Мб Сложность: 73%)*

Вася и Петя играют в следующую игру. Вася кладет на стол два ряда карточек. Первый ряд состоит из N карточек, на каждой из которых написано некоторое число ai. Второй ряд состоит из N карточек, на каждой из которых написано некоторое число bi.

Пете требуется переупорядочить карточки первого ряда так, чтобы на столе получилось два одинаковых ряда карточек. За одну секунду Петя может поменять местами i-ую и (i+1)-ую (1 ≤ i ≤ N-1) карточки первого ряда.

Помогите Пете переупорядочить карточки, затратив на это минимальное время.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит целое число N (1 ≤ N ≤ 105). Вторая строка содержит N целых чисел ai. Третья строка содержит N целых чисел bi. Все числа по абсолютной величине не превосходят 106.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите количество секунд, за которые Петя сможет переупорядочить карточки требуемым образом. Если переупорядочить карточки требуемым образом невозможно, выведите единственное число -1.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 5 3 2 3 4 5 5 4 3 2 3 | 7 |
| 8 1 2 1 2 1 2 1 2 2 1 2 1 2 1 2 2 | -1 |

***Описание решения:*** Первый ряд превращаем в пары (число, исходная позиция).

Второй ряд превращаем в пары (число, исходная позиция).

Сортируем каждый ряд в первую очередь по числу, во вторую очередь по позиции.

(Невозможно, если после сортировки в первом массиве пар число в какой-то позиции отличается от числа во втором)

Если возможно, - получить массив троек (число, позиция во 2 массиве, позиция в 1 массиве)

Отсортируем массив троек по позициям во 2 массиве.

. . . . \*

3 2 3 4 5  
 5 4 3 2 3

Если поменялось число – ставим звездочку в логическом массиве.

Позиция в реальном времени = позиция исходной + количество звездочек левее исходной позиции.

Листинг программы приведен в Приложении 10.

1. ***0525 Сумма степеней двойки***

*(Время: 1 сек. Память: 16 Мб Сложность: 43%)*

Любое натуральное число можно представить в виде суммы натуральных слагаемых, каждое из которых является степенью числа 2. Суммы, различающиеся лишь порядком слагаемых, считаются одинаковыми. Например, для числа 7 таких представлений 6 (4+2+1, 4+1+1+1, 2+2+2+1, 2+2+1+1+1, 2+1+1+1+1+1, 1+1+1+1+1+1+1).

Требуется написать программу, которая найдет количество способов такого представления заданного числа N.

**Входные данные**

Входной файл INPUT.TXT содержит число N (1 ≤ N ≤ 1000).

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно число – найденное количество способов представления числа N.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 4 | 4 |
| 7 | 6 |

***Описание решения:*** По формуле a[i]=a[i/2]+a[i-2] можно посчитать количество представлений числа i. Для наглядности ниже разложения нескольких чисел.

a[1]=1 1

a[2]=2 1+1 2

a[3]=2 1+1+1 1+2

a[4]=4 1+1+1+1 1+1+2 2+2 4

Листинг программы приведен в Приложении 11.

1. ***0553 Объединения блоков***

*(Время: 1 сек. Память: 16 Мб Сложность: 46%)*

Изделие изготавливают из n блоков, каждый из которых имеет два технологических параметра – mi и ki. Известно, что ki=mi+1, i=1, 2, …, n-1. При этом условии два последовательных блока i и i+1 можно объединять в один новый, который будет иметь технологические параметры - mi и ki+1, и на это потребуется mi\*ki+1 технологических операций. Новый блок можно опять объединять с одним из соседних и так далее. Меняя порядок сборки блоков можно добиться уменьшения количества технологических операций.

Поясним это на примере трех блоков: 34 и 29, 29 и 4, 4 и 15. Если собрать вначале 2 и 3 блок, а затем присоединить собранное к первому, то потребуется 29\*15+34\*15=435+510=945 операций. Если собрать вначале блок из 1 и 2 исходных блоков, а затем присоединить 3 блок, то потребуется 34\*4+34\*15=136+510=646 операций.

Требуется написать программу, которая найдет минимальное число технологических операций для изготовления изделия.

**Входные данные**

Входной файл INPUT.TXT содержит в первой строке число n – количество блоков (1 ≤ n ≤ 100). Последующие n строк содержат пары чисел (разделенных пробелом) – технологические параметры блоков. Технологические параметры – целые неотрицательные числа, не превышающие 100.

**Выходные данные**

Выходной текстовый файл OUTPUT.TXT должен содержать одно число – минимальное число технологических операций.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 3 34 29 29 4 4 15 | 646 |

***Описание решения:*** «Блок» в процессе сборки мог получиться только ищ «подстроки» исходных блоков, то есть из последовательности идущий подряд блоков.

cost[left][right] – минимальная стоимость получения единого блока из исходных блоков от left до right.

cost[x][x]=0

cost[left][right]=min(cost[left][right1]+cost[left1][right]+m[left]\*k[right], right+1=left1, left1<=right1, left1 <=right)

Листинг программы приведен в Приложении 12.

1. ***0581 Атака летающих тарелок***

*(Время: 1 сек. Память: 16 Мб Сложность: 36%)*

Вы работаете в фирме, занимающейся разработкой компьютерных игр. Сейчас вы занимаетесь разработкой новой компьютерной игры "Атака летающих тарелок". По сюжету игры на планету Зумла приземляются летающие тарелки, и их надо уничтожать. Игрок управляет лазерной пушкой. Для того, чтобы произвести выстрел он указывает две точки на поверхности Зумлы (которая в игре считается плоской), через которые должен проходить лазерный луч (который является прямой).

Вы должны написать программу, определяющую, какие летающие тарелки были уничтожены выстрелом.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит целое число n (1 ≤ n ≤ 30000) - число приземлившихся летающих тарелок. Вторая строка содержит числа xp1, yp1, xp2, yp2 - координаты точек, через которые проходит лазерный луч. Далее идут n строк, каждая из которых содержит описание одной летающей тарелки в формате xi yi ri, где xi, yi - координаты центра, ri - радиус тарелки. Все числа целые и не превосходят по модулю 10000. Радиусы летающих тарелок - целые и положительные. Летающие тарелки могут касаться и пересекаться.

**Выходные данные**

В первую строку выходного файла OUTPUT.TXT выведите количество уничтоженных летающих тарелок. Во вторую строку выведите номера уничтоженных летающих тарелок в возрастающем порядке. Тарелка считается уничтоженной, если она имеет, хотя бы одну общую точку с лазерным лучом.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 2 0 0 1 1 2 2 100 1000 1000 1 | 2 1 2 |

***Описание решения:*** Нам нужно найти с какими кругами прямая, заданная двумя точками имеет хотя бы одну общую точку.

Расстояние от прямой до центра круга <= радиус круга.

|Ax+By+C|/sqrt(A2 +B2) <=r

Листинг программы приведен в Приложении 13.

1. ***0609 Разбиения множества***

*(Время: 1 сек. Память: 16 Мб Сложность: 69%)*

Рассмотрим множество, состоящее из первых n натуральных чисел: Nn = {1, 2, …, n}. Разбиение - это представление этого множества в виде объединения одного или нескольких непустых попарно непересекающихся множеств. Примерами разбиений для n = 5 являются:

{1, 2, 3, 4, 5} = {1, 2, 3} U {4, 5}

{1, 2, 3, 4, 5} = {1, 3, 5} U {2, 4}

{1, 2, 3, 4, 5} = {1, 2, 3, 4, 5}

{1, 2, 3, 4, 5} = {1} U {2} U {3} U {4} U {5}

Всего существует 52 разбиения множества N5. Заметим, что разбиения, отличающиеся только порядком объединяемых множеств, не различаются.

Разбиения множества Nn можно упорядочить лексикографически. Для того, чтобы определить этот порядок, вначале определим лексикографический порядок на подмножествах Nn. Будем говорить, что подмножество A={a1, a2, ..., ak} (a1 < a2 < ... < ak) множества Nn лексикографически меньше подмножества B={b1, b2, ..., bm} (b1 < b2 < ... < bm) множества Nn и писать A < B, если верно одно из следующих утверждений:

найдётся i, такое что a1 = b1, a2 = b2, ..., ai - 1 = bi - 1 и ai < bi;

k < m и при этом a1 = b1, a2 = b2, ..., ak = bk.

Таким образом, для любых двух подмножеств A и B верно ровно одно из трёх утверждений: или A < B, или A совпадает с B, или B < A, а также есть транзитивность: если A < B и B < C, то A < C. Теперь определим каноническое представление разбиения как представление, в котором объединяемые множества упорядочены лексикографически.

Разбиения упорядочиваются лексикографически следующим образом. Разбиение Nn = A1 U A2 U … U Ak лексикографически меньше разбиения Nn = B1 U B2 U … U Bm, если существует такое i, что A1 = B1, A2 = B2, . . . , Ai-1 = Bi-1 и Ai < Bi.

По разбиению множества Nn найдите следующее в лексикографическом порядке разбиение.

**Входные данные**

Входной файл INPUT.TXT содержит несколько описаний тестов. Каждое описание является каноническим представлением разбиения. Первая строка описания содержит n и k - количество элементов в разбиваемом множестве и количество частей в разбиении (1 ≤ n ≤ 200). Последующие k строк содержат элементы разбиения. Элементы каждого множества упорядочены по возрастанию. Описания тестов отделены друг от друга пустыми строками. Последняя строка входного файла содержит два нуля. Этот тест не должен обрабатываться. Сумма n по всем описаниям не превосходит 2000.

**Выходные данные**

В выходной файл OUTPUT.TXT для каждого теста выведите следующее в лексикографическом порядке разбиение. Если разбиение во входном файле является последним в лексикографическом порядке, выведите первое в лексикографическом порядке. Используйте тот же формат, что и во входном файле. Отделяйте разбиения друг от друга пустыми строками. Элементы каждого множества следует выводить в порядке возрастания.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 5 2 1 2 3 4 5  5 2 1 3 5 2 4  5 1 1 2 3 4 5  5 5 1 2 3 4 5  0 0 | 5 2 1 2 3 4 5  5 4 1 4 2 3 5  5 2 1 2 3 5 4  5 4 1 2 3 4 5 |

***Описание решения:*** Нужно изменять как можно меньше последних элементов.

Меньше всего – это последнее множество разбить на части.

Так же: как можно дальше идти по одинаковым элементам и изменять как модно меньше последних.

Если в последнем множестве разложений больше 2 – то предпоследний элемент выделяется в отдельное множество. Положение этого множества в конце.

Если последнее множество из 2 элементов, то разделение его на 2 множества даст меньший в лексикографическом порядке результат.

Если в последнем множестве 1 или 2 элемента, то придется изменять 2 последних множества.

Нужно заменить минимальное количество элементов в конце первого из изменяемых множеств. Заменить число большим и после этого закрыть это множество, а все оставшиеся числа отправить в следующее множество.

Листинг программы приведен в Приложении 14.

1. ***0637 NEERC***

*(Время: 1 сек. Память: 16 Мб Сложность: 17%)*

В полуфинале студенческого чемпионата мира по программированию NEERC (http://neerc.ifmo.ru) участвуют команды из n институтов. Участники для проведения соревнований распределяются по k залам, каждый из которых имеет размеры, достаточные для размещения всех команд от всех институтов. При этом по правилам соревнований в одном зале может находиться не более одной команды от института.

Многие институты уже подали заявки на участие в полуфинале. Оргкомитет полуфинала хочет допустить до участия максимально возможное количество команд. При этом, разумеется, должна существовать возможность рассадить их по залам без нарушения правил.

Напишите программу, определяющую максимальное количество команд, которые можно допустить до участия в полуфинале.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит число n - число институтов, подавших заявки. Вторая строка входного файла содержит n чисел a1, …, an (ai - это количество команд, заявленных от института номер i). Последняя строка входного файла содержит число k - количество залов, в которых проходят соревнования.

Все числа во входном файле целые, положительные и не превосходят 10000.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите одно целое число - ответ на задачу.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 3 1 2 4 3 | 6 |
| 3 1 2 4 4 | 7 |

***Описание решения:*** Создаем целочисленный массив размера 10000, и считываем в него количество команд от i-того института, дальше считываем в переменную количество залов. Дальше в цикле проверяем, если количество команд от i-того института больше количества залов, то увеличиваем сумму на максимально допустимое количество залов, в ином случае, если команд меньше, чем залов, которыми мы можем воспользоваться, то увеличиваем на меньшее число (количество команд). Выводим сумму.

Пример:

3 – число институтов  
 1 2 4 – количество команд от каждого института  
 3 – количество залов

Первую команду от первого института ставим в 1-ый зал, первую команду от второго института ставим в 1-ый зал, первую команду от 3-его института ставим в 1 зал, итого 3 команды от разных институтов находится в 1-ом зале, вторую команду от 2-ого института ставим во 2-ой зал, вторую команду от 3-его института ставим во 2 зал, итого 5 команд в 2-ух залах, остается 2 команды от 3-его института. Т.к в один зал можно поместить одну команду от одного института, то в зал влезет только одна команда от 3-его института. Итого – 6 команд.

Листинг программы приведен в Приложении 15.

1. ***0665 Палиндромное время***

*(Время: 1 сек. Память: 16 Мб Сложность: 24%)*

Пекарь считает, что для получения рождественского пирога идеальной симметричной формы его нужно вынимать из духовки в тот момент, когда часы показывают «палиндромное» время, которое читается одинаково слева-направо и справа-налево.

Напишите программу, которая определяет по времени установки пирога в духовку время, когда будет подходящее время для его извлечения.

**Входные данные**

В единственной строке входного файла INPUT.TXT записано время установки пирога в духовку в формате HH:MM (00 ≤ HH ≤ 23, 00 ≤ MM ≤ 59).

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести ближайшее «палиндромное» время в формате HH:MM.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 00:00 | 01:10 |
| 12:34 | 13:31 |
| 23:59 | 00:00 |

***Описание решения:*** Используя бесконечный цикл прибавляем наше время до ближайшего зеркального. И если время зеркальное выходим из цикла.

Листинг программы приведен в Приложении 16.

1. ***0693 Перестановка***

*(Время: 1 сек. Память: 16 Мб Сложность: 24%)*

Если Вы читали Гарри Поттера, то знаете, что повелитель зла, Лорд Волдеморт создал свое имя путем перестановки букв в своем настоящем имени. Так из имени «Tom Marvolo Riddle» он получил «I am Lord Voldemort».

Напишите программу, которая проверяет, можно ли получить из одного имени другое путем перестановки его букв. При этом регистром букв нужно пренебречь.

**Входные данные**

В первой строке входного файла INPUT.TXT записаны два слова, разделенные единственным в строке пробелом. Слова содержат только символы английского алфавита. Длина слов больше 0 и не превышает 50 символов.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите «Yes», если возможно получить из одного имени другое, и «No» в противном случае.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| TomMarvoloRiddle IamLordVoldemort | Yes |
| stop pots | Yes |
| abbc bac | No |

***Описание решения:*** Сравниваем 2 строки и, если символы находящиеся в 1 строке есть и во 2, то выводим ответ «Да». Иначе «Нет».

Листинг программы приведен в Приложении 17.

1. ***0777 Будильник***

*(Время: 1 сек. Память: 16 Мб Сложность: 12%)*

Известный исследователь Чарльз Ф. Мантц, устав от долгого путешествия через джунгли, лег спать в 10 часов вечера, но предварительно он завел будильник на 12 часов следующего дня. Но проспать 14 часов ему не удалось – будильник зазвонил через 2 часа. Исследователь забыл, что на будильнике, имеющем 12-тичасовой циферблат, можно задать время до звонка только менее 12 часов.

Напишите программу, которая определяет, сколько часов успеет проспать исследователь, прежде чем будильник его разбудит.

**Входные данные**

В единственной строке входного файла INPUT.TXT записаны два целых числа S и T (1 ≤ S, T ≤ 12), разделенные одним пробелом - час, когда исследователь лег спать, и час, на который он установил будильник.

**Выходные данные**

В единственную строку выходного файла OUTPUT.TXT нужно вывести одно целое число – через сколько часов зазвонит будильник.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 10 12 | 2 |

***Описание решения:*** Находим разницу между наибольшим и наименьшим. И, если результат положительный, то выводим это число. Иначе прибавляем 12 и выводим ответ.

Листинг программы приведен в Приложении 18.

1. ***0889 Паутина***

*(Время: 1 сек. Память: 16 Мб Сложность: 38%)*

Имеется паутина на плоскости, состоящая из нитей, параллельных осям координат. В основе конструкции паутины лежит бесконечное множество вертикальных нитей, пронумерованных слева направо, начиная с единицы. Смежные вертикальные нити могут быть соединены горизонтальными нитями на некоторой высоте. Для каждой вертикальной нити на определенной высоте может быть не более одного подобного соединения.

На одной из вертикальных нитей, в самом верху (выше всех горизонтальных соединений) находится паук, который начинает свое движение вниз. Натыкаясь на очередную нить, паук меняет свое направление. Движение паука продолжается до тех пор, пока все горизонтальные нити не окажутся выше паука.

По заданной конструкции паутины и начальному расположению паука требуется определить номер нити, на которой окажется паук в конце своего движения.

**Входные данные**

Первая строка входного файла INPUT.TXT содержит два числа K и M, где K – номер нити, соответствующей начальному положению паука. Далее идут M строк, определяющих горизонтальные соединения, по одному в каждой строке. Каждое такое соединение определяется двумя числами P и H, обозначающие соединение смежных вертикальных нитей с номерами P и P+1 на высоте H. Во входных данных все числа натуральные, имеющие следующие ограничения: K, P, H ≤ 109, M ≤ 105.

**Выходные данные**

В выходной файл OUTPUT.TXT выведите целое число – ответ на задачу.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 3 6 1 2 3 4 2 5 3 1 1 4 2 3 | 2 |

***Описание решения:*** Сначала сортируем перемычки по уменьшению высоты. Если очередная перемычка относится к текущей нити, то перескакиваем на ее соседа, который определяется этой перемычкой.

Листинг программы приведен в Приложении 19.

1. ***0945 Баллы***

*(Время: 1 сек. Память: 16 Мб Сложность: 45%)*

Мир наш развивается, строятся города, люди улетают в космос, изменяется система аттестации студентов в СФУ. Но вот проблема - систему аттестации студентов изменили, а программное обеспечение, которое поставлено в деканатах для контроля успеваемости, оставили прежним. Поэтому Вам срочно требуется внедрить во всех деканатах новую программу поиска студентов с заданным баллом!

**Входные данные**

В первой строке входного файла INPUT.TXT содержатся натуральные числа N и K (N, K ≤ 200 000) – соответственно количество студентов, подлежащих аттестации, и число запросов декана об успеваемости студентов. Во второй строке находятся N целых чисел ai, упорядоченных по возрастанию. Эти числа - аттестационные баллы студентов. В третьей строке располагаются K целых чисел bi, определяющие искомый балл. (0 ≤ ai, bi ≤ 232)

**Выходные данные**

В выходной файл OUTPUT.TXT выведите для каждого из K запросов через пробел слово «YES», если студент с таким баллом есть, и «NO» в противном случае.

|  |  |
| --- | --- |
| INPUT.TXT | OUTPUT.TXT |
| 3 4 1 6 9 7 9 10 1 | NO YES NO YES |
| 2 2 1 2 1 3 | YES NO |

***Описание решения:*** С помощью бинарного поиска ищем есть ли во 2 строке числа из 1.

Листинг программы приведен в Приложении 20.

**Список консультаций**

За период прохождения практики были посещены следующие консультации.

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Дата | Содержание | Сданные задачи |
|  | 04.02.2019 | Установочная конференция |  |
|  | 26.02.2019 | Защита решений задач | 0022, 0049, 0189, 0273, 0301, 0329, 0357, 0385 |
|  | 26.03.2019 | Защита решений задач | 0413, 0441, 0525, 0553, 0581, |
|  | 18.04.2019 | Защита решений задач | 0609, 0637, 0665 |
|  | 25.04.2019 | Защита решений задач | 0693, 0777, 0889, 0945 |
|  | 27.04.2019 | Заключительная конференция |  |

# Заключение

Учебная практика способствовала закреплению и углублению теоретической подготовки, приобретению практических навыков и компетенций в сфере профессиональной деятельности путём самостоятельного решения задач.

В ходе практики было решено 20 задач, реализованных на языке программирования С++. Листинги всех составленных программ приведены в приложении.

Наиболее интересной оказалась 637.

Наиболее сложной оказалась 49.

# Список литературы

1. Ахо А., Хопкрофт Дж., Ульман Дж. Структуры данных и алгоритмы. – М.: Вильямс, 2010.
2. Вирт Н. Алгоритмы и структуры данных. М.: ДМК, 2009.
3. Гагарина, Л. Г. Алгоритмы и структуры данных [Электронный ресурс]: учебное пособие / Л. Г. Гагарина, В. Д. Колдаев. – М.: Финансы и статистика, 2009. – 304 с. (режим доступа: <http://biblioclub.ru/index.php?page=book&id=225965>).
4. Кнут Д. Искусство программирования. Том 1. Основные алгоритмы. – 3-е изд. – М.: Вильямс, 2011.
5. Кнут Д. Искусство программирования. Том 3. Сортировка и поиск. – 2-е изд. – М.: Вильямс, 2000.
6. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы. Построение и анализ. – М.: Вильямс, 2011.
7. Окулов С. М. Основы программирования [Текст] / С. М. Окулов. – 6-е изд., перераб. – М.: БИНОМ. Лаборатория Знаний, 2012. – 336 с.
8. Окулов С. М. Программирование в алгоритмах [Текст] / С. М. Окулов. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2012. – 383 с.
9. Окулов С. М. Алгоритмы обработки строк [Электронный ресурс] / С. М. Окулов. – М.: БИНОМ. Лаборатория знаний, 2012. – 256 с. (режим доступа: <http://biblioclub.ru/index.php?page=book&id=222847>).
10. Окулов С. М., и др. Задачи по программированию [Текст] / С. М. Окулов, Т. В. Ашихмина, Н. А. Бушмелева, М. А. Корчемкин, Е. В. Разова, Р. В. Шарыгин; под ред. С. М. Окулова. – М.: БИНОМ. Лаборатория знаний, 2014, 823 с.
11. Окулов, С. М. Дискретная математика. Теория и практика решения задач по информатике [Текст] : учеб. пособие / С. М. Окулов. – М.: БИНОМ. Лаборатория знаний, 2013. – 422 с.

Приложения

**Приложение 1. Задача 0021. Зарплата**

#include <iostream>

using namespace std;

int main()

{

int n = 3, max = 0, min = 999999, result = 0;

int \*A = new int[n];

for (int i = 0; i < n; i++)

{

cin >> \*(A + i);

}

for (int i = 0; i < n; i++)

{

if (max < \*(A + i))

{

max = \*(A + i);

}

if (min > \*(A + i))

{

min = \*(A + i);

}

}

result = max - min;

cout << max - min << endl;

return 0;

}

**Приложение 2. Задача 0049. Шаблоны**

#include <cstdio>

#include <string>

#include <vector>

#include <iostream>

using namespace std;

int main()

{

vector<string> possible(128);

possible['0'] = "0";

possible['1'] = "1";

possible['2'] = "2";

possible['3'] = "3";

possible['4'] = "4";

possible['5'] = "5";

possible['6'] = "6";

possible['7'] = "7";

possible['8'] = "8";

possible['9'] = "9";

possible['a'] = "0123";

possible['b'] = "1234";

possible['c'] = "2345";

possible['d'] = "3456";

possible['e'] = "4567";

possible['f'] = "5678";

possible['g'] = "6789";

possible['?'] = "0123456789";

char buffer[9 + 1];

cin >> buffer;

string s1(buffer);

cin >> buffer;

string s2(buffer);

int p = 1;

for (int i = 0; i < (int)s1.size(); i++)

{

int c1 = s1[i];

int c2 = s2[i];

int digit = 0;

for (char d = '0'; d <= '9'; d++)

{

if ((int)possible[c1].find(d) != -1 && (int)possible[c2].find(d) != -1)

{

digit++;

}

}

p \*= digit;

}

cout << p;

return 0;

}

**Приложение 3. Задача 0189. Перестановки по номеру**

#include <iostream>

#include <vector>

#include <set>

using namespace std;

int fact[13] = { 1, 1, 2, 6, 24, 120, 720, 5040, 40320, 362880, 3628800, 39916800, 479001600 };

int notUsed(vector<bool> used, int blockNum) {

int j = 0, pos = 0;

for (j = 1; j < used.size(); j++) {

if (!used[j]) pos++;

if (blockNum == pos)

break;

}

return j;

}

vector<int> permutation\_by\_num(int n, int num) {

vector<int> res(n);

vector<bool> used(n + 1, false);

for (int i = 0; i < n; i++) {

int blockNum = (num - 1) / fact[n - i - 1] + 1;

int j = notUsed(used, blockNum);

res[i] = j;

used[j] = true;

num = (num - 1) % fact[n - i - 1] + 1;

}

return res;

}

int main() {

int n, num;

cin >> n >> num;

vector<int> perm = permutation\_by\_num(n, num);

for (int i = 0; i<perm.size(); i++)

cout << perm[i] << ' ';

return 0;

}

**Приложение 4. Задача 0273. Вычеркивание**

#include <iostream>

#include <vector>

#include <string>

#include <cstdio>

int main()

{

char buffer[101 + 1];

std::cin >> buffer;

std::string s(buffer);

std::vector<bool> was(1000, false);

for (int i = 0; i < (int)s.size(); i++)

{

for (int j = i + 1; j < (int)s.size(); j++)

{

for (int k = j + 1; k < (int)s.size(); k++)

{

was[(s[i] - '0') \* 100 + (s[j] - '0') \* 10 + (s[k] - '0')] = true;

}

}

}

int count = 0;

for (int i = 100; i <= 999; i++)

{

if (was[i])

{

count++;

}

}

std::cout << count;

return 0;

}

**Приложение 5. Задача 0301. Код**

#include <iostream>

#include <cstdio>

#include <vector>

#include <algorithm>

using namespace std;

int main()

{

int sum, n;

cin >> sum >> n;

vector<int> max(n);

{

int s = sum;

for (int i = 0; i < n; i++)

{

max[i] = min(9, s);

s -= max[i];

}

}

vector<int> min(n);

{

int s = sum-1;

for (int i = n-1; i >= 0; i--)

{

min[i] = std::min(9, s);

s -= min[i];

}

min[0]++;

}

for (int d : max)

{

cout << d;

}

cout << endl;

for (int d : min)

{

cout << d;

}

}

**Приложение 6. Задача 0329. Лесенка-2**

#include <iostream>

#include <vector>

int main()

{

int n;

std::cin >> n;

std::vector<int> a(1 + n);

a[0] = 0;

for (int i = 1; i <= n; i++)

{

std::cin >> a[i];

}

std::vector<int> maxsumfron(1 + n);

std::vector<int> next(1 + n);

maxsumfron[n] = a[n];

next[n] = 0;

maxsumfron[n-1] = a[n-1]+a[n];

next[n-1] = n;

for (int i = n - 2; i >= 0; i--)

{

if (maxsumfron[i + 1] > maxsumfron[i + 2])

{

maxsumfron[i] = a[i] + maxsumfron[i + 1];

next[i] = i + 1;

}

else

{

maxsumfron[i] = a[i] + maxsumfron[i + 2];

next[i] = i + 2;

}

}

std::cout << maxsumfron[0]<<std::endl;

for (int i = next[0]; i != n; i = next[i])

{

std::cout << i <<" ";

}

std::cout << n;

}

**Приложение 7. Задача 0357. Делимость на 11**

#include <iostream>

#include <string>

using namespace std;

int even=0,odd=0,i;

int main()

{freopen("input.txt","r",stdin);

freopen("output.txt","w",stdout);

string a;

cin>>a;

for(i=0;i<a.length();i++){

{if((i + 1) % 2 == 0)even+=a[i]-'0';

if((i + 1)%2==1)odd+=a[i]-'0';}

}if((odd-even)%11==0)cout<<"YES";

else cout<<"NO";

return 0;

}

**Приложение 8. Задача 0385. Развлечения с измерителем**

#include <iostream>

#include <fstream>

#include <cmath>

#include <ctime>

#include <stdlib.h>

using namespace std;

int main(){

unsigned int start, end;

start=clock();

ifstream fin("input.txt");

ofstream fout("output.txt");

int x[200],y[200];

int n,i,j,flag,h;

double d,r[5000];

int k;

fin>>n; if((n<2)&&(n>50)) return 0;

for(i=0;i<n;i++){

fin>>x[i]>>y[i];

if(x[i]>10000) return 0;

if(y[i]>10000) return 0;

cout<<x[i]<<" "<<y[i]<<endl;

}

fin.close();

k=0;

for(i=0; i<(n-1); i++){

for(j=i+1; j<n; j++){

d=sqrt(pow(x[j]-x[i],2)+pow(y[j]-y[i],2));

flag=0;

for(h=0; h<k; h++){

if(d==r[h]) flag=1;

}

if(flag==0){

r[k]=d;

k++;

}

}

}

for(i=0; i<k; i++)

for(j=0; j<(k-1); j++)

if(r[j]>r[j+1]){

d=r[j];

r[j]=r[j+1];

r[j+1]=d;

}

fout<<k<<endl;

fout.setf(std::ios::fixed);

fout.precision(9);

for(i=0;i<k;i++) fout<<r[i]<<endl;

fout.close();

return 0;

}

**Приложение 9. Задача 0413. Военная база**

#include <stdio.h>

char \*\* matrix;

unsigned int a, b;

void check(unsigned int x, unsigned int y)

{

if (x<a-1)

if (matrix[x+1][y]=='#')

{

matrix[x+1][y] = '.';

check(x+1, y);

}

if (x>0)

if (matrix[x-1][y]=='#')

{

matrix[x-1][y] = '.';

check(x-1, y);

}

if (y<b-1)

if (matrix[x][y+1]=='#')

{

matrix[x][y+1] = '.';

check(x, y+1);

}

if (y>0)

if (matrix[x][y-1]=='#')

{

matrix[x][y-1] = '.';

check(x, y-1);

}

}

int main()

{

FILE \* f = fopen("input.txt", "r");

fscanf(f, "%d %d", &a, &b);

matrix = new char\* [a];

for (int i = 0; i < a; i++)

{

matrix [i] = new char [b];

fscanf(f,"%s", matrix[i]);

}

fclose(f);

unsigned int sum = 0;

for (int i = 0; i < a; i++)

for (int j = 0; j<b; j++)

if (matrix[i][j] == '#')

{

sum++;

check(i,j);

}

f = fopen("output.txt", "wt");

fprintf(f, "%d", sum);

fclose(f);

return 0;

}

**Приложение 10. Задача 0441. Карточки-2**

#include <iostream>

#include <vector>

#include <algorithm>

#include <stdio.h>

void check(bool e)

{

if (!e)

{

throw 1;

}

}

struct Pair

{

int value;

int index;

Pair()

{

value = 5;

index = 10;

}

Pair(const int value, const int index)

: value(value),

index(index)

{}

};

struct Triple

{

int value;

int index1;

int index2;

Triple(const int value, const int index1, const int index2)

: value(value)

, index1(index1)

, index2(index2)

{}

};

struct Set

{

const int PART = 400;

std::vector<bool> isMarked;

std::vector<int> count;

Set(const int n)

: isMarked(n, false)

, count((n + PART - 1) / PART, 0)

{}

void mark(const int index)

{

check(!isMarked[index]);

isMarked[index] = true;

count[index / PART]++;

}

int countUP(int index) const

{

int range = (index + PART + 1) / PART \* PART;

int res = 0;

while (index < (int)isMarked.size() && index < range)

{

if (isMarked[index])

{

res++;

}

index++;

}

for (int i = range / PART; i < (int)count.size(); i++)

{

res += count[i];

}

return res;

}

};

int main()

{

int n;

std::cin >> n;

std::vector<Pair> a;

for (int i = 0; i < n; i++)

{

int value;

std::cin >> value;

a.push\_back(Pair(value, i));

}

std::stable\_sort(a.begin(), a.end(), [](const Pair &left, const Pair &right)

{

return left.value < right.value;

});

std::vector<Pair> b;

for (int i = 0; i < n; i++)

{

int value;

std::cin >> value;

b.push\_back(Pair(value, i));

}

std::stable\_sort(b.begin(), b.end(), [](const Pair &left, const Pair &right)

{

return left.value < right.value;

});

std::vector<Triple> c;

for (int i = 0; i < n; i++)

{

if (a[i].value != b[i].value)

{

std::cout << "-1";

return 0;

}

c.push\_back(Triple(a[i].value, a[i].index, b[i].index));

}

std::sort(c.begin(), c.end(), [](const Triple &left, const Triple &right)

{

return left.index2 < right.index2;

});

long long ans = 0;

Set set(n);

for (const Triple &t : c)

{

ans += t.index1 + set.countUP(t.index1 + 1) - t.index2;

set.mark(t.index1);

}

std::cout << ans;

}

**Приложение 11. Задача 0525. Сумма степеней двойки**

#include <iostream>

#include <vector>

#include <cstdio>

int main()

{

int n;

std::cin >> n;

std::vector<int> count(1+n);

count[0]=1;

count[1]=1;

for (int i = 2; i <= n; i++)

{

count[i]=count[i-2]+count[i/2];

}

std::cout << count.back();

}

**Приложение 12. Задача 0553. Объединение блоков**

#include <iostream>

#include <cstdio>

#include <vector>

int main()

{

int n;

std::cin >> n;

std::vector<int> leftPar(1 + n);

std::vector<int> rightPar(1 + n);

for (int i = 1; i <= n; i++)

{

std::cin >> leftPar[i] >> rightPar[i];

}

std::vector<std::vector<int>>cost(1 + n, std::vector<int>(1 + n));

for (int len = 1; len <= n; len++)

{

for (int left = 1; left + len - 1 <= n; left++)

{

int right = left + len - 1;

if (len == 1)

{

cost[left][right] = 0;

}

else

{

int min = 1000\*1000\*1000;

for (int right1 = left; right1 < right; right1++)

{

int left2 = right1 + 1;

min = std::min(min, cost[left][right1] + cost[left2][right]);

}

cost[left][right] = min + leftPar[left] \* rightPar[right];

}

}

}

std::cout << cost[1][n];

return 0;

}

**Приложение 13. Задача 0581. Атака летающих тарелок**

#include <iostream>

#include <cstdio>

#include <vector>

int main()

{

int n;

std::cin >> n;

int x0, y0, x1, y1;

std::cin >> x0 >> y0 >> x1 >> y1;

int a = y1 - y0;

int b = x0 - x1;

int c = y0 \* x1 - x0 \* y1;

std::vector<int> ans;

for (int i = 1; i <= n; i++)

{

int x, y, r;

std::cin >> x >> y >> r;

if (1LL\*(a\*x + b \* y + c)\*(a\*x + b \* y + c) <= 1LL\*(r \* r)\*(a\*a + b \* b))

{

ans.push\_back(i);

}

}

std::cout << int(ans.size())<< std::endl;

for (int i = 0; i < (int)ans.size(); i++)

{

if (i > 0)

{

std::cout << " ";

}

std::cout << ans[i];

}

}

**Приложение 14. Задача 0609. Разбиения множества**

#include <iostream>

#include <cstdio>

#include <set>

#include <vector>

#include <cstdlib>

#include <algorithm>

#include <cassert>

bool nextSplit(std::vector<std::vector<int>> &a)

{

std::vector<int> free;

for (int m = (int)a.size() - 1; m >= 0; m--)

{

int last = a[m].back();

std::sort(free.begin(), free.end());

int found = -1;

for (int i = 0; i < (int)free.size(); i++)

{

if (free[i] > last)

{

found = free[i];

break;

}

}

if (found != -1)

{

a[m].push\_back(found);

for (int x : free)

{

if (x != found)

{

a.push\_back(std::vector<int>(1, x));

}

}

return true;

}

if (a[m].size() == 1)

{

free.push\_back(last);

a.pop\_back();

}

else

{

free.push\_back(last);

a[m].pop\_back();

for (int i = (int)a[m].size() - 1; i >= 1; i--)

{

int cur = a[m][i];

std::sort(free.begin(), free.end());

found = -1;

for (int i = 0; i < (int)free.size(); i++)

{

if (free[i] > cur)

{

found = free[i];

break;

}

}

if (found != -1)

{

a[m][i] = found;

for (int x : free)

{

if (x == found)

{

x = cur;

}

a.push\_back(std::vector<int>(1, x));

}

return true;

}

free.push\_back(cur);

a[m].pop\_back();

}

assert(a[m].size() == 1);

free.push\_back(a[m][0]);

a.pop\_back();

}

}

assert(a.size() == 0);

std::sort(free.begin(), free.end());

for (int x : free)

{

a.push\_back(std::vector<int>(1, x));

}

return false;

}

int main()

{

bool first = true;

while (true)

{

int n, nLines;

scanf("%d %d ",&n, &nLines);

if (nLines == 0)

{

break;

}

if (first)

{

first = false;

}

else

{

printf("\n");

}

std::vector<std::vector<int>> a(nLines);

for (int i = 0; i < nLines; i++)

{

int cur = 0;

while (true)

{

int c = getchar();

if ('0' <= c && c <= '9')

{

cur = cur \* 10 + (c - '0');

}

else

{

assert(c == ' ' || c == '\n');

if (cur > 0)

{

a[i].push\_back(cur);

cur = 0;

}

if (c == '\n')

{

break;

}

}

}

}

nextSplit(a);

printf("%d %d\n", n, (int)a.size());

for (auto line : a)

{

for (int i = 0; i < (int)line.size(); i++)

{

if (i > 0)

{

printf(" ");

}

printf("%d", line[i]);

}

printf("\n");

}

}

}

**Приложение 15. Задача 0637. NEERC**

#include <iostream>

using namespace std;

int main ()

{

int c=0,n,a[10000],k;

cin>>n;

for(int i=0;i<n;i++)

cin>>a[i];

cin>>k;

for(int i=0;i<n;i++)

{

if(a[i]>=k)

c+=k;

else

c+=a[i];

}

cout<<c;

return 0;

}

**Приложение 16. Задача 0665. Палиндромное время**

#include <iostream>

#include <stdio.h>

#include <string.h>

using namespace std;

int main()

{

char h[6] = "\0"; // формат ввода 00:00

int a, b; //десятки и еденицы часов

int c, d; //десятки и еденицы минут

cin >> h;

a = h[0] - '0';

b = h[1] - '0';

c = h[3] - '0';

d = h[4] - '0';

if (strcmp(h, \_strrev(h)) == 0)d += 1;

while (1)

{

if (a == d && b == c)

{

cout << a << b << ":" << c << d;

return 0;

}

d++;

if (d >= 10)

{

d = 0; c++;

}

if (c == 6)

{

c = 0; b++;

}

if (b == 10)

{

b = 0; a++;

}

if (a == 2 && b == 4)

{

a = 0; b = 0;

}

}

}

**Приложение 17. Задача 0693. Перестановка**

#include <iostream>

bool is\_cmp(const char\* s1, const char\* s2)

{

const int N = 26;

int c, cs[N] = { 0 };

while (\*s1) {

c = \*s1++;

if (c >= 'A' && c <= 'Z')

++cs[c - 'A'];

else if (c >= 'a' && c <= 'z')

++cs[c - 'a'];

}

while (\*s2) {

c = \*s2++;

if (c >= 'A' && c <= 'Z')

--cs[c - 'A'];

else if (c >= 'a' && c <= 'z')

--cs[c - 'a'];

}

for (int i = 0; i < N; ++i) {

if (cs[i] != 0)

return false;

}

return true;

}

int main(void)

{

char s1[] = "TomMarvoloRiddle";

char s2[] = "IamLordVoldemort";

std::cin >> s1>>s2;

if (is\_cmp(s1, s2))

std::cout << "Yes";

else

std::cout << "No";

std::cout << std::endl;

return 0;

}

**Приложение 18. Задача 0777. Будильник**

#include <iostream>

int main()

{

int S, T, result;

std::cin >> S >> T;

result = T - S;

if (result < 0)

{

result += 12;

}

std::cout << result;

}

**Приложение 19. Задача 0889. Паутина**

#include <iostream>

#include <stdio.h>

#include <vector>

#include <algorithm>

struct Hor {

int x, y;

};

int main()

{

int start, n;

std::cin >> start >> n;

std::vector<Hor> h(n);

for (int i = 0; i < n; i++)

{

std::cin >> h[i].x >> h[i].y;

}

std::sort(h.begin(), h.end(), [](const Hor &left, const Hor &right)

{

return left.y > right.y;

});

int cur = start;

for (Hor hor : h)

{

if (hor.x == cur)

{

cur++;

}

else

{

if (hor.x + 1 == cur)

{

cur--;

}

}

}

std::cout << cur;

}

**Приложение 20. Задача 0945. Баллы**

#include <iostream>

#include <stdio.h>

#include <vector>

#include <algorithm>

#include <cassert>

int main()

{

int n, q;

std::cin >> n >> q;

std::vector<long long> a(n);

for (int i = 0; i < n; i++)

{

std::cin >> a[i];

}

for (int i = 0; i < q; i++)

{

long long x;

std::cin >> x;

if (i > 0)

{

std::cout << std::endl;

}

if (std::binary\_search(a.begin(), a.end(), x))

{

std::cout << "YES";

}

else

{

std::cout << "NO";

}

}

return 0;

}