Разбор задач

39 Районная олимпиада школьников Красноярского края по информатике, 9-11 классы (Муниципальный этап ВОШ по информатике)

8 декабря 2025 г.

ID	Задача	Тема	%
2213	A. Робинзон Крузо	Задачи для начинающих	12
2209	B. Сломанные часы	Геометрия	33
2210	C. Друзья	Разное	27
2212	D. Хаотичные разбиения	Рекурсия / Перебор	42
2214	E. Счастливый билет	Моделирование	46
2215	F. Age of Empires	Динамическое программирование	54

Разбор: Беляев Сергей Николаевич

Задача A. Робинзон Крузо

(Время: 1 сек. Память: 32 Мб Баллы: 100)

Робинзон Крузо на необитаемом острове отмечает дни на стене своей хижины. Каждый день он ставит зарубку, которую будем обозначать английской буквой «I», а раз в 5 дней зачеркивает четыре предыдущие зарубки, получая символ, который мы обозначим как «V». Если данный символ дублирует предыдущий, то он их оба заменяет символом «X».

Какая запись получится на стене хижины Робинзона на N-й день?

Входные данные

Входной файл INPUT.TXT содержит целое число N ($1 \le N \le 10\,000$).

Выходные данные

В выходной файл OUTPUT.TXT выведите запись, которая получится на стене хижины Робинзона на N-й день.

Примеры

№	INPUT.TXT	OUTPUT.TXT
1	3	III
2	12	XII
3	19	XVIIII

Задача A. Робинзон Крузо

Дано:

N – время в днях Робинзона на острове

Требуется найти:

S – запись из симовлов «X», «V» и «I», которые Робинзон оставит после себя за N дней

Решение:

Вычислим количество десятков, наличие в остатке пяти дней и число оставшихся единиц:

N // 10 – количество десятков дней

N % 10 // 5 – число оставшихся пятёрок (0 или 1)

N % 5 – оставшиеся единицы

Здесь знаком «//» обозначено целочисленное деление, а знак «%» обозначает остаток от деления (как в языке Python).

```python
#Python
n = int(input())
print(n//10*'X' + n%10//5*'V' + n%5*'I')
```

```cpp
//C++
#include <bits/stdc++.h>

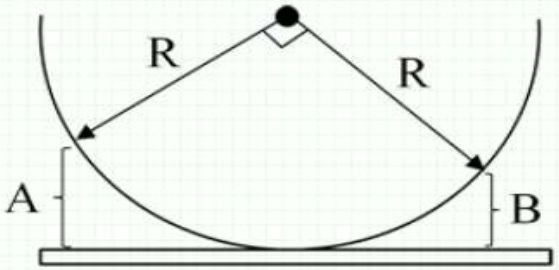
using namespace std;

int main(){
  int n;
  cin >> n;
  cout << string(n/10, 'X') +
          string(n%10/5, 'V') +
          string(n%5, 'I') << endl;
  return 0;
}
```

Задача B. Сломанные часы

(Время: 1 сек. Память: 32 Мб Баллы: 100)

Однажды механические круглые часы, висевшие на стене, упали и сломались (перестали идти). Вследствие чего секундная стрелка отлетела, а часы остались лежать на полу, прикасаясь к стене вплотную. Всё это случилось ровно в 15:00. Таким образом, угол между часовой и минутной стрелками оказался прямым, а сами стрелки оказались не выше центра циферблата часов.



Известно, что конец одной из стрелок оказался на высоте A от пола, а конец другой – на высоте B. Здесь следует полагать, что длины стрелок равны и совпадают с радиусом циферблата часов.

Требуется определить площадь циферблата часов.

Входные данные

Каждая из первых двух строк файла INPUT.TXT содержит по одному целому числу. Эти числа A и B – расстояния от концов стрелок до пола ($0 \le A, B \le R \le 100$).

Выходные данные

В выходной файл OUTPUT.TXT выведите ответ на задачу с точностью, не худшей чем 10^{-3}.

Примеры

№	INPUT.TXT	OUTPUT.TXT
1	3 3	329.589789
2	5 0	78.539816
3	3 6	706.858347

Система оценки

Решения, работающие только для A = B, будут оцениваться в 20 баллов.

Задача B. Сломанные часы

Дано:

A и B – расстояния от стрелок до пола

Требуется:

Вычислить площадь циферблата часов.

Решение:

Для решения на 20 баллов достаточно использовать данные первого теста и квадратичную зависимость вычисляемой площади от линейных размеров.

Для полного решения следует найти радиус цифербата R, после чего ответ получим по формуле $S = \pi R^2$.

Построим прямоугольник так, чтобы нижняя его сторона лежала на полу, а другие проходили через точки O, C и D.

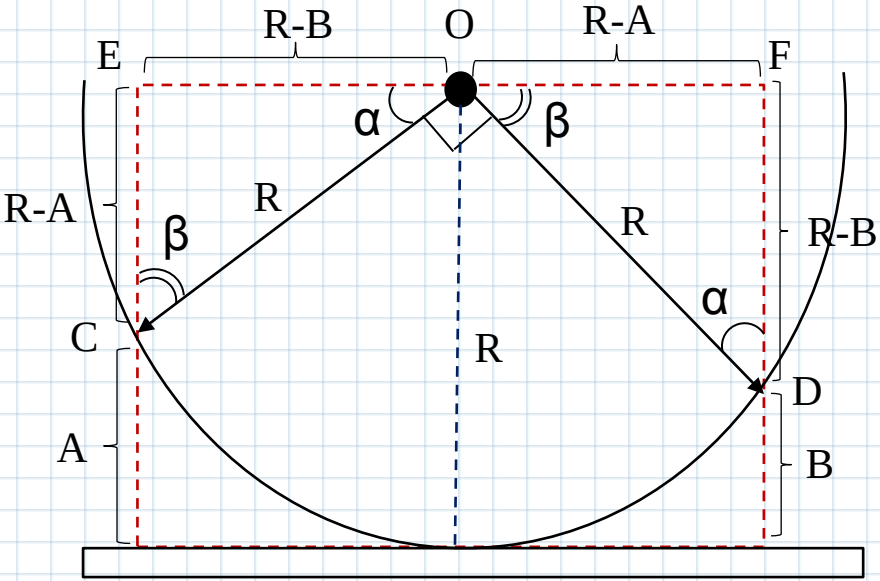
Теперь заметим, что образованные треугольники OCE и DOF равны по двум углам и гипотенузе. Действительно, гипотенуза – это стрелки длины R, а $\alpha+\beta+90^\circ = 180^\circ$ как сумма углов треугольника и как развернутый угол.

Откуда по теореме Пифагора мы можем записать равенство, привести его к квадратному уравнению и вычислить значение R.

Заметим, что при решении уравнения у нас может получиться только один корень, т.к. по условию задачи $R \geq \max(A, B)$.

Действительно, иначе мы бы получили при $0 < A \leq B < R$:

$$R = A + B - \sqrt{2AB} \leq A + B - \sqrt{2AA} = A + B - A\sqrt{2} < B$$



$$(R-A)^2 + (R-B)^2 = R^2$$

$$R^2 - 2RA + A^2 + R^2 - 2RB + B^2 = R^2$$

$$R^2 - 2(A+B)R + (A^2 + B^2) = 0$$

$$D = 4(A+B)^2 - 4(A^2 + B^2) = 8AB$$

$$R = \frac{2(A+B) + \sqrt{8AB}}{2} = A + B + \sqrt{2AB}$$

Задача B. Сломанные часы

```python
#Python
from math import *

a = int(input())
b = int(input())

r = a + b + sqrt(2*a*b)
s = pi*r*r

print(s)
```

```cpp
//C++
#include <bits/stdc++.h>

using namespace std;

double a,b,r,s,pi = 3.1415926536;

int main(){
  cin >> a >> b;
  r = a + b + sqrt(2*a*b);
  s = pi*r*r;
  cout << fixed << setprecision(6)
       << s << endl;
  return 0;
}
```

Задача C. Друзья

(Время: 2 сек. Память: 32 Мб Баллы: 100)

В одном классе учится N мальчиков, все они приходят в школу в определенном порядке. Тот, кто приходит первым имеет номер 1, кто приходит вторым – номер 2 и так далее. Последний пришедший имеет номер N.

Некоторые пары ребят описанного выше класса являются друзьями. Причем очевиден тот факт, что если Петя дружит с Васей, то и Вася дружит с Петей. Известно, что когда очередной школьник заходит в класс, то он обходит всех своих друзей и жмет им руку. При этом нам неизвестно кто именно с кем дружит, но достоверно известно: сколько каждый мальчик делает рукопожатий при входе в класс.

Классный руководитель решил выяснить: кто из ребят наиболее социально активен (имеет наибольшее количество друзей). По имеющейся информации необходимо определить какое максимальное количество друзей может быть у одного из ребят.

Входные данные

В первой строке входного файла INPUT.TXT записано единственное число N ($1 \le N \le 200\ 000$) – количество мальчиков в классе. Вторая строка содержит N целых чисел h_i ($0 \le h_i < i$) – i-е число соответствует количеству рукопожатий, которое совершил школьник с номером i сразу после своего прихода в школу, то есть до прихода следующего школьника с номером i+1.

Выходные данные

В выходной файл OUTPUT.TXT выведите ответ на задачу.

Пример

№	INPUT.TXT	OUTPUT.TXT
1	3 0 1 1	2

Пояснение к примеру

Если оба школьника с номерами 2 и 3 поздороваются с первым, то окажется, что у него два друга.

Система оценки

Решения, работающие только для $N \le 1000$, будут оцениваться в 50 баллов.

Задача C. Друзья

Дано:

N – количество школьников

H[1..N] – массив рукопожатий (H[i] – число рукопожатий, которые сделал i-й школьник

Требуется найти:

R – максимально возможное количество друзей у одного из школьников

Решение:

Вычислим максимальное возможное количество друзей у определенного школьника с номером i. Когда он вошел в класс, то поздоровался с H[i] своими друзьями, поэтому у него их уже как минимум H[i]. Также он мог еще поздороваться с частью ребят, которые вошли в класс после него. Максимальное количество таких рукопожатий равно количеству поздоровавшихся хотя бы с одним человеком, которым как раз мог бы быть i-й мальчик. Таким образом, если мы вычислим максимально мозможное количество друзей для каждого из ребят, то сможем найти ответ как максимальный из всех этих значений.

Если для каждого i-го мальчика вычислять количество поздоровавшихся с ним после его прихода в цикле, то мы получим асимптотику $O(N^2)$, что даст только 50 баллов. Поэтому следует провести оптимизацию этого подсчета, рассматривая всех школьников в обратном порядке. Так, останавливаясь на i-м школьнике мы сразу будем знать максимально возможное количество его друзей, что и даст нам линейный алгоритм решения.

Алгоритмическая реализация:

```
//чтение входных данных
read(N)
for i = 1 … N
  read(H[i])

R = 0     //будущий ответ
cnt = 0   //число поздоровавшихся

//поиск ответа в обратном порядке
for i = N … 1
  A = H[i] + cnt
  if A > R
    R = A
  if H[i] > 0
    cnt = cnt + 1

write(R)
```

Задача D. Хаотические разбиения

(Время: 2 сек. Память: 32 Мб Баллы: 100)

Рассмотрим все представления числа N в виде суммы различных целых возрастающих слагаемых:

$N = A_1 + A_2 + \ldots + A_K$, где $A_1 < A_2 < \ldots < A_K$.

Будем называть такое разбиение хаотическим, если сумма любых трех подряд идуших слагаемых не делится на 13. Иначе говоря, для всех i от 2 до K-1 должно выполняться условие: сумма $A_{i-1} + A_i + A_{i+1}$ не делится на 13.

Задано число N. Выведите все его хаотические разбиения на слагаемые.

Входные данные

Входной файл INPUT.TXT содержит целое число N ($1 \le N \le 80$).

Выходные данные

В выходной файл OUTPUT.TXT выведите все хаотические разбиения на слагаемые числа N. Разбиения можно выводить в любом порядке. Выводите слагаемые в каждом разбиении, разделяя их знаком «+» без пробелов.

Пример

№	INPUT.TXT	OUTPUT.TXT
1	9	1+2+6 1+3+5 1+8 2+3+4 2+7 3+6 4+5 9

Система оценки

Решения, работающие только для N < 13, будут оцениваться в 40 баллов.

Задача D. Хаотические разбиения

Для решения задачи организуем рекурсивный перебор всевозможных искомых последовательностей. Для чего реализуем рекурсивную функцию rec(K, N), которая будет продолжать заполнять некоторый массив A, начиная с K-го элемента так, чтобы оставшаяся сумма была равна N, а текущий элемент был строго больше предыдущего. Таким образом, в K-ю позицию мы сможет ставить числа от A[K-1]+1 до N в порядке возрастания и вызывать эту же функцию для (K+1) –го элемента, проверяя делимость последних трёх на 13.

Алгоритмическая реализация вышеописанного:

```
a[0..99]={0}
read(n)
rec(1, n)

sub rec(k, n)
  if n = 0
    for i=1..k-2
      write(a[i], '+')
    writeln(a[k-1])
  else
    for t=a[k-1]+1..n
      if k<3 or (a[k-2]+a[k-1]+t)%13>0
        a[k] = t
        rec(k+1, n-t)
```

Задача E. Счастливый билет

(Время: 1 сек. Память: 32 Мб Баллы: 100)

Счастливый билет – это такой номер автобусного билета, состоящий из чётного количества десятичных цифр, в котором сумма первой половины цифр совпадает с суммой последней. При этом известно, что все номера билетов состоят из одинакового числа цифр и могут иметь ведущие нули. Например, если в записи билета используется 4 цифры, то билет с номером 123 будет записан как 0123. Билеты на каждой ленте нумеруются от 00…01 до 99…99. Таким образом, в ленте всего 10^K-1 билетов, где K – количество используемых цифр. Билеты 2112 и 051312 счастливые, а 9993 и 74 – нет.

Однажды Вася зашел в автобус и получил билет с номером N. Поскольку Васе было далеко ехать и делать в автобусе особо нечего, то он задумался: с каким номером будет следующий счастливый билет, который будет выдан из той же ленты, что и билет у Васи.

Входные данные

Входной файл INPUT.TXT содержит целое число N – номер билета, который получил Вася. Данный номер может содержать лидирующие нули. Гарантируется, что количество цифр в записи числа N чётно и не превосходит 10^5.

Выходные данные

В выходной файл OUTPUT.TXT выведите номер следующего счастливого билета из текущей ленты в таком же формате, как и во входных данных. Если такого билета не существует, выведите номер минимального счастливого билета из новой ленты.

Примеры

№	INPUT.TXT	OUTPUT.TXT
1	0517	0523
2	0615	0624

Система оценки

Решения, работающие только для 4-значных билетов, будут оцениваться в 20 баллов.

Решения, работающие только для 4-значных и 8-значных билетов, будут оцениваться в 40 баллов.

Решения, работающие только для билетов, состоящих не более чем из 16 цифр, будут оцениваться в 60 баллов.

Задача E. Счастливый билет

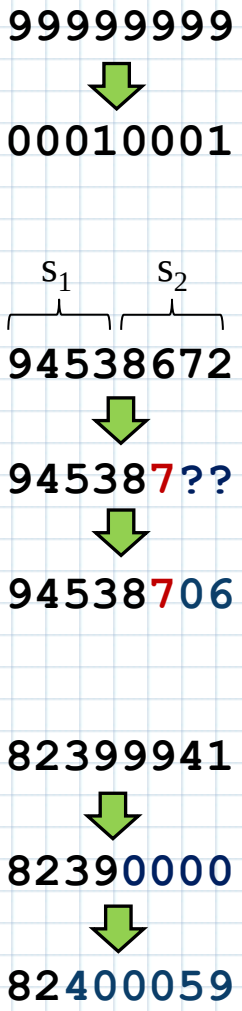
Решение:

Решение «в лоб» (перебор всех чисел, начиная с N+1 с подсчетом сумм цифр левой и правой частей для каждого числа) позволяет набрать всего 40 баллов. Для 60 баллов необходимо использовать более оптимизированный перебор. Мы не будем рассматривать эти частичные решения. Рассмотрим полное решение задачи:

1. Сначала проверим: не состоит ли наше число из одних девяток? Если так, то выведем минимальный счастливый билет из новой ленты вида 000...001 000...001.

2. Найдем самую правую цифру во второй половине числа, такую, что при увеличении ее можно изменяя цифры справа получить сумму цифр s2 такую, что $s1 \le s2$ и такую, что $s1 \ge s2$. Данную цифру нужно увеличить на минимально возможное такое значение, а цифры правее обратить в ноль. Далее, следует заменить цифры правее её так, чтобы их сумма была равна s1-s2. В правой части после этой цифры получим что-то вроде 00...0d99...9, где d – некоторая цифра.

3. Если пункт 2 невыполним, т.е. такой цифры не существует, то необходимо число в левой части увеличить на единицу, а все цифры справа заменить на нули и проделать тот же алгоритм, что и ранее, но здесь уже для всей правой половины числа. При увеличении левой части на единицу следует учесть, что там могут быть девятки. Для этого следует найти в числе самую правую цифру, отличную от 9 и увеличить её на 1, а девятки обратить в ноль.



Задача F. Age of Empires

(Время: 2 сек. Память: 32 Мб Баллы: 100)

Представьте себе стратегическую компьютерную игру «Age of Empires», в которой вы управляете империей, включающей в себя территорию определенной площади и армию. Вам необходимо поддерживать стабильность империи, которая выражается целым числом от 1 до 100. Если стабильность в определенный момент становится ниже единицы, то происходит крушение империи и игра заканчивается. В начале игры стабильность равна 100.

Каждый игровой год Вам предлагается для присоединения одна из соседних территорий, и Вы должны выбрать: присоединять или не присоединять эту территорию. В случае присоединения стабильность снижается на некоторую величину F_i, а размер империи увеличивается на размер присоединяемой территории T_i. В начале игры Вы имеете нулевой размер территории империи. Если Вы не присоединяете территорию, то стабильность возрастает на значение P. При этом если сумма становится больше 100, то она уменьшается до 100.

Определите значение максимально возможного размера империи, который может быть у Вас к концу игры при условии, что в процессе игры не было крушения империи.

Входные данные

В первой строке входного файла INPUT.TXT содержатся два целых числа: продолжительность партии в годах N ($1 \le N \le 10^4$) и скорость стабилизации в годы без присоединений P ($0 \le P \le 99$).

Далее следует N строк, каждая из которых содержит два целых числа: i-я строка описывает территорию, возможную для присоединения в i-й год: величину уменьшения стабильности при присоединении F_i ($1 \le F_i \le 99$) и размер территории T_i ($1 \le T_i \le 10^5$).

Выходные данные

В выходной файл OUTPUT.TXT выведите ответ на задачу.

Пример

№	INPUT.TXT	OUTPUT.TXT
1	3 15 40 1900 92 4740 70 3200	5100

Пояснение к примеру

Здесь нужно присоединить первую и третью территории. В первый год стабильность после присоединения снизится до 60, во второй год возрастет до 75, в третий снова снизится до 5.

Система оценки

Решения, работающие только для P = 0 и F_i = 99 и для теста из условия, будут оцениваться в 25 баллов.

Решения, работающие только для P = 99 и F_i = 99 и для теста из условия, будут оцениваться в 30 баллов.

Решения, работающие только для P = 0 и для теста из условия, будут оцениваться в 50 баллов.

Задача F. Age of Empires

N – продолжительность партии в годах
P – размер увеличения стабилизации без присоединения
F[i] – размер уменьшения стабилизации при присоединении территории в i-й год, i = 1..N
T[i] – площадь территории, которая может быть присоединена в i-й год, i = 1..N

Используем метод динамического программирования:

dp[K][R] – максимально возможная площадь государства по истечении K лет (K=0..N) со стабилизацией равной R (R=1..100)

Алгоритмическая реализация:

```
read(N, P)
dp[0..N][1..100] = {-INF}                    //неопределенные значения ставим в -бесконечность
dp[0][100] = 0                               //в начале игры стабилизация = 100, площадь = 0
//пробегаем по годам и вычисляем значения dp для i-го года, через предыдущий (i-1)-й год
for i = 1..N
    read(F, T)
    //определяем значения для случая без присоединений
    for j = P+1..99
        dp[i][j] = dp[i-1][j-p]              //стабилизация возрастает на P по отношению к прошлому году
    dp[i][100] = max(dp[i-1][100-p..100])    //стабилизация 100 может быть получена из стаб. от 100-P до 100
    //определяем значания для случая с присоединением территории
    for j = 1..100-F
        dp[i][j] = max(dp[i][j], dp[i-1][j+F]+T)  //также учитываем значения для случая без присоединения
//выводим ответ как максимальное значение среди всех возможных стабильностей после N-го года
write(max(dp[n][1..100]))
```