Муниципальный этап всероссийской олимпиады школьников
по информатике в 2021/2022 учебном году для 7-8 классов

Разбор заданий

Задача 1. Андрей и аквариум

Будем перебирать первое число (назовем его A) так, чтобы $A \cdot A \cdot A \leq 24$, а также 24 делилось на A. Тогда нам подходят числа 1, 2 (3 не подходит, так как $3 \cdot 3 \cdot 3 = 27 > 24$). Второе число (назовем его B) будем перебирать так, чтобы $A \leq B \cdot B \leq 24/A$. Тогда получаются следующие случаи:

1. 1 1 24
2. 1 2 12
3. 1 3 8
4. 1 4 6
5. 2 2 6
6. 2 3 4

Таким образом, всего существует шесть подходящих аквариумов.

Если в какой-то строке введено не три числа, либо введено не число, либо произведение в какой-либо строке не равно 24, то участник получает 0 баллов.

Иначе участник получается $\left[\frac{100\ cnt}{6}\right]$, где cnt — количество различных троек в ответе участника. Тройки, различающиеся порядком, считаются одинаковыми.

Задача 2. Удаление данных

Для решения первой подзадачи можно было написать простейший перебор с возвратом. Такое решение набирало не менее 20 баллов.

Для решения второй подзадачи требовалось заметить следующий факт: выгодно удалять элемент, соседний с одним из минимальных элементов.

Таким образом, решение второй подзадачи выглядит следующий образом:

• Если в массиве остался один элемент, завершаем алгоритм.

• Если в массиве хотя бы два элемента, находим минимальный. Пусть он равен amin (если их несколько, выбирем любой), и удаляем одного из его соседей. При этом к ответу добавляется amin.

Чтобы решить задачу на 100%, нужно было заметить следующий факт: При удалении элемента, соседнего с минимальным, минимум массива не меняется. Это значит, что на каждой итерации цикла из предыдущего листинга к ответу будет прибавляться одно и то же число, равное минимальному элементу изначального массива a. Поскольку всего будет ровно n − 1 интерация, то ответ равен amin · (n − 1).

Таким образом, решение на 100 баллов имеет следующий вид:

```
n = int (input ( ) )
mn = 10 ** 9
for i in range ( n ) :
        cu r = int (input ( ) )
        mn = min(mn, cu r )
print (mn * ( n − 1 ) )
```

Задача 3. Путешествие по джунглям

Для начала поймем, что если Коко может добраться до лианы с номером i, то она может добраться и до всех предыдущих лиан. Будем рассматривать лианы слева направо и поддерживать номер самой правой лианы, до которой сможет добраться горилла.

Обозначим за X максимальный номер лианы, до которой сможет добраться Коко. Изначально скажем, что X = 1, так как в начале Коко находится на лиане с номером 1. Пусть мы рассмотрели первые i − 1 лиан, и на данный момент известно, что Коко сможет добраться до лианы с номером X. Теперь рассмотрим лиану с номером i и поймем, насколько далеко горилла сможет прыгнуть, используя ее.

Если X < i, то Коко никак не сможет попасть на лиану с номером i, а значит она не сможет воспользоваться данной лианой. В этом случае ответ на задачу равен X.

Если же X ≥i, то можно попробовать воспользоваться текущей лианой. Известно, что с нее Коко может прыгнуть не далее, чем на ai метров вправо. Так как расстояние между соседними лианами равно D, Коко сможет допрыгнуть с текущей лианы на лиану с номером i + $\left[\frac{a_i}{D}\right]$. Если данное число больше, чем X, то обновим значение X, ведь теперь горилла смогла добраться до лианы с бОльшим номером.

Ниже приведен пример решения задачи на языке Python.

```
n = int (input ( ) )
d = int (input ( ) )
X = 1
i = 1
while i <= n and i <= X:
        jump = int (input ( ) )
        X = max(X, i + jump // d )
        i += 1
X = min(X, n )
print (X)
```

Задача 4. Земляничная поляна

Пример из 11 команд:

E>A
E>C
E>D
C>B
A>C
D>F
F>A
A>D
B>D
D>A
A>B

Пример решения задачи представлен в таблице:

шаг	20	3	5	6	7	30
1.	17	3	0	0	0	0
2.	11	3	0	6	0	0
3.	4	3	0	6	7	0
4.	выбыл	3	5	1	7	0
5.	выбыл	0	5	4	7	0
6.	выбыл	0	5	выбыл	0	7
7.	выбыл	3	5	выбыл	0	4
8.	выбыл	0	5	выбыл	3	выбыл
9.	выбыл	0	1	выбыл	7	выбыл
10.	выбыл	3	1	выбыл	4	выбыл
11.	выбыл	0	4	выбыл	выбыл	выбыл

Задача 5. Долгое вычитание, Карл!

Частные решения.

Первая подзадача.

Если число n — однозначное, то ответ равен самому числу. Если число n — двузначное четное, то потребуется [(n − 8)/2]с операций (через [·] обозначается целая часть от деления), чтобы число стало равно 8. Итого ans = (n - 8) // 2 + 8. Если число n — двузначное нечетное, то потребуется [(n − 9)/2] операций, чтобы число стало равно 9. Итого ans = (n - 9) // 2 + 9.

```
n = int (input ( ) )
i f  n % 2 :
      ans = ( n − 9 ) // 2 + 9
e l s e :
```

```
    ans = ( n − 8 ) // 2 + 8
i f  n < 1 0:
    ans = n
 print ( ans )
```

Вторая подзадача.

Моделирование процесса вычитания «в чистом виде»: пока число не обратится в 0, вычитать из него его длину, подсчитывая количество операций.

```
n = int (input ( ) )
ans = 0
while n :
       ans += 1
      n −= len ( str ( n ) )
print ( ans )
```

Полное решение.

Разберем конкретный пример. Пусть n = 123456.

Найдем первое пятизначное число, встретившееся нам в процессе уменьшения n. Очевидно, оно имеет такой же остаток от деления на 6, что и n. Возьмем число 99999 и будем уменьшать его на 1, пока остатки не равны. Получим новое n = 99996. При этом мы использовали [(123456 − 99996)/6] = 3910 операций.

Далее аналогично. Найдем первое четырехзначное число, встретившееся нам в процессе уменьшения n. Очевидно, оно имеет такой же остаток от деления на 5, что и n. Возьмем число 9999 и будем уменьшать его на 1, пока остатки не равны. Получим новое n = 9996. При этом мы использовали [(99996 − 9996)/5] = 18000 операций (всего 21910).

Найдем первое трехзначное число, встретившееся нам в процессе уменьшения n. Очевидно, оно имеет такой же остаток от деления на 4, что и n. Возьмем число 999 и будем уменьшать его на 1, пока остатки не равны. Получим новое n = 996. При этом мы использовали [(9996 − 996)/4] = 2250 операций (всего 24160).

Найдем первое двузначное число, встретившееся нам в процессе уменьшения n. Очевидно, оно имеет такой же остаток от деления на 3, что и n. Возьмем число 99 и будем уменьшать его на 1, пока остатки не равны. Получим новое n = 99. При этом мы использовали [(996 − 99)/3] = 299 операций (всего 24459).

Наконец, найдем первое однозначное число, встретившееся нам в процессе уменьшения n. Очевидно, оно имеет такой же остаток от деления на 2, что и n. Возьмем число 9 и будем уменьшать его на 1, пока остатки не равны. Получим новое n = 9. При этом мы использовали [(99 − 9)/2] = 45 операций (всего 24504).

Не забудем, что n нужно уменьшить до 0, поэтому добавим к ответу еще 9. Итого 24513.

Запрограммируем этот процесс: будем последовательно находить наибольшие числа длины меньше, чем исходное на 1, которое встретится нам в процессе вычитания (и будет последним, полученным при вычитании длины исходного числа), пока исходное число не уменьшится до однозначного.

```
n = int (input ( ) )
ans = 0 while n > 9 :
       new_n = int ( ’ 9 ’ ∗ (len ( str ( n ) ) − 1 ) )
       while new_n % len ( str ( n ) ) != n % len ( str ( n ) ) :
            new_n −= 1
       ans += ( n − new_n) // len ( str ( n ) )
       n = new_n
ans += n
print ( ans )
```