Школьный этап ВсОШ 2020-2021

Новосибирск

Разбор задач 9-11 класса

Автором всех задач и разбора является Львов Антон Павлович.
Автор благодарит за тестирование и ценные замечания Голованова Александра Игоревича и Атучина Михаила Михайловича.

Задача A. Лунтик-бизнесмен

Если Лунтин отдаст деньги Вупсеню, то в итоге он получит $(N - X) \cdot A$ рублей, а если Пупсеню, то $(N - Y) \cdot B$. Нужно просто вывести максимум из этих двух чисел. Пример решения на языке Python:

```
N = int(input())
X = int(input())
A = int(input())
Y = int(input())
B = int(input())
print(max((N - X) * A, (N - Y) * B))
```

Задача B. Лунтик-бухгалтер

Заметим, что если бы Лунтику выдали пять купюр в 1 рубль, то их можно бы было заменить на одну купюру в 5 рублей и уменьшить общее количество купюр, не изменив общую сумму. Значит, купюр в 1 рубль не более четырёх. Аналогично, купюр в 5 рублей не более одной, в 10 рублей — не более трёх, в 40 рублей — не более четырёх. Значит, при оптимальной выдаче денег (наименьшим количеством купюр) купюры в 1, 5, 10 и 40 рублей в сумме дают не более 4 + 5 + 30 + 160 = 199 рублей.

Отсюда следует, что купюр в 200 рублей среди выданных ровно N div 200 (на большее количество не хватит денег, а если их меньше, то оставшимися придётся набрать хотя бы 200 рублей, что, как мы поняли, невозможно). Повторяя рассуждения, оставшиеся $N - (N \text{ div } 200) = N \bmod 200$ рублей набираются максимально возможным количеством купюр в 40 рублей и т.д.

Такие алгоритмы, которые впритык набирают сначала самые большие объекты, потом поменьше, а в самом конце самые маленькие, называются *жадными алгоритмами*, и работают, на самом деле, не всегда. Поэтому в этой задаче и приведено обоснование, почему конкретно в этом случае так делать можно.

Пример решения на языке Python:

```
N = int(input())
ans = N // 200
N = N % 200
ans += N // 40
N = N % 40
ans += N // 10
N = N % 10
ans += N // 5
N = N % 5
ans += N
print(ans)
```

Задача C. Лунтик-математик

Заметим, что Лунтик будет сидеть не более суток — в 00 : 00 время точно станет удовлетворять условию (на самом деле сидит Лунтик вообще не более часа). Значит, можно просто перебрать все значения минут и часов, начиная с текущего, пока не будет выполнено описанное в задаче условие. Значение цифр в двузначном числе — это остаток при делении на 10, а значение десятков — целая часть от деления на 10.

Будем увеличивать количество минут до тех пор, пока либо условие не выполнится, либо минут не станет 60. Во втором случае количество минут сделает равным нулю, а количество часов увеличим на 1. Если их станет 24, то сделаем их равным нулю. Пример решения на языке Python:

```
h = int(input())
m = int(input())

ans = 0
while (h % 10) * (h // 10) != (m % 10) * (m // 10):
    m += 1
    if m == 60:
        m = 0
        h += 1
        if h == 24:
            h = 0
    ans += 1
print(ans)
```

Задача D. Лунтик-садовод

Наивным решением будет перебрать все возможные j, для каждого посчитать сумму размеров k сорняков и найти максимум из них всех. Заметим, что j может принимать значения от 1 до $n-k+1$, и для каждого j мы считаем сумму k слагаемых, то есть тратим по крайней мере $(n-k+1)k \approx k(n-k)$ операций. Для k порядка $n/2$ это количество становится равно $n^2/4$. Так как n может быть порядка 10^5, то количество операций уже приближается к миллиарду, что не влезает в секунду, отводимую на решение задачи.

Следующее решение набирает 30 баллов и не проходит по времени на остальных тестах:

```
n, k = map(int, input().split())
A = list(map(int, input().split()))
ans_n = 0
cur = sum(A[:k])
ans = cur
cur_n = 1
while cur_n + k - 1 < n:
    cur = sum(A[cur_n:cur_n + k])
    if cur > ans:
        ans=cur
        ans_n=cur_n
    cur_n += 1
print(ans_n + 1)
```

Вместо этого заметим, что при переходе от j к $j+1$ у нас исчезает крайнее левое растение и добавляется одно растение справа. Поэтому можно явно посчитать только сумму первых k чисел, а затем двигать это "окно" длины k, на каждом шаге вычитая из суммы крайнее левое число и добавляя новое справа. Тогда каждое число мы максимум один раз прибавим и один раз вычтем, то есть количество операций будет порядка $2n$, что проходит по времени.

Следующее решение набирает 100 баллов:

```
n, k = map(int, input().split())
A = list(map(int, input().split()))
ans_n = 0
cur = sum(A[:k])
ans = cur
cur_n = 1
while cur_n + k - 1 < n:
    cur -= A[cur_n - 1]
    cur += A[cur_n + k - 1]
    if cur > ans:
        ans = cur
        ans_n = cur_n
    cur_n += 1
print(ans_n + 1)
```

Задача E. Лунтик-шахматист

Заметим, что в таких задачах, в которых возможен короткий ответ, можно, если нет никаких идей, посылать программу, которая сдаёт только этот ответ. В частности, программа, на любые входные данные отвечающая "NO", проходит 2 теста и набирает 10 баллов. Перейдём теперь к, собственно, решению.

Сразу разберём случаи $A = nm$ — можно оставить всё поле пустым и $A = 0$ — можно всё поле заполнить ладьями.

Предположим теперь, что нам удалось расставить несколько ладей так, чтобы не билось A полей. Пусть ладьи **не** стоят в x строках и в y столбцах. Тогда очевидно, что непобито ровно xy полей вне зависимости от конкретного расположения ладей. Тогда нужно подобрать такие x и y, что $A = xy$. Заметим кроме того, что $1 \leqslant x \leqslant n-1$ и $1 \leqslant y \leqslant m-1$, так как

мы разобрали случаи, когда ни одного или все поля. Поэтому нам достаточно перебрать все возможные делители d числа A, что $d \leqslant n-1$ и $A/d \leqslant m-1$. Если такое d нашлось, выведем таблицу, в которой ладьи стоят на пересечении первых $n-d$ строк и $m-A/d$ столбцов. Если такого d не нашлось, то расставить ладьи невозможно.

Пример решения на языке Python:

```
A, n, m = map(int, input().split())
field = [['.'] * m for i in range(n)]
if A == n * m:
    print('YES')
    print(*[''.join(row) for row in field], sep='\n')
elif A == 0:
    print('YES')
    field = [['R'] * m for i in range(n)]
    print(*[''.join(row) for row in field], sep='\n')
else:
    for d in range(1, A + 1):
        if A % d == 0 and d <= n - 1 and A // d <= m - 1:
            for i in range(n - d):
                for j in range(m - A // d):
                    field[i][j] = 'R'
            print('YES')
            print(*[''.join(row) for row in field], sep='\n')
            break
    else:
        print('NO')
```