Разбор задач

Задача 1. Порядок во всём

Если очередное число уже больше или равно предыдущего числа (после дописывания цифр к предыдущему числу), то его нужно оставить без изменений. Если новое число является префиксом (началом) предыдущего числа, то нужно дописать цифры так, чтобы оно стало равно предыдущему числу. Во всех остальных случаях нужно дописывать нули, чтобы получить число такой же длины или на 1 большей длины. Ответ:

48
50
67
300
820
820
6300
7010
54600
54600

Задача 2. Треугольники

Внутри каждого единичного квадрата можно выбрать 4 треугольника площади $\frac{1}{2}$ и 4 треугольника площади $\frac{1}{4}$. Также внутри двух соседних квадратов можно выбрать 2 больших треугольника площади 1. Пару соседних квадратов можно выбрать $n-1$ способом. Итого $4n+4n+2(n-1)$.

Ответ (можно записать в виде любого эквивалентного выражения): $8*n+2*(n-1)$.

Задача 3. Электронное табло

В первом задании необходимо выполнить две операции «+», чтобы получить цифру 2 на последнем месте, затем поменять их местами, получится 20, затем получить 23. Ответ на первое задание (получить 23): $++*++$.

Во втором задании вторая цифра числа большая, поэтому лучше получить число 38 из числа 40 вычитанием числа 2. Ответ на второе задание (получить 38): $++++*--$.

Используя соображение, что для получения больших цифр лучше использовать операцию вычитания, можно получить ответы и для оставшихся случаев. При этом в некоторых случаях, как, например, для получения числа 84, лучше получить число 48 из числа 50, затем переставить цифры числа в обратном порядке.

Ответ на третье задание (получить 65): $+*---*-----$.

Ответ на четвёртое задание (получить 84): $++++++*--*$.

Ответ на пятое задание (получить 99: $+*-*-*+$.

Задача 4. 90 минут

Добавив в таблицу первую строку для записи заголовков. Будем использовать несколько вспомогательных столбцов.

В столбце C посчитаем время поездки в минутах относительно начала для удобства вычисления разности времён двух поездок. Это можно сделать при помощи формулы =LEFT(A2;LEN(A2)−3) ∗ 60+RIGHT(A2;2).

Поездки будут разбиваться на группы, соответствующие одному тарифу. Для того чтобы тарифицировать одну поездку, нам нужно знать время первой поездки в этой группе (будем записывать его в столбце D) и количество поездок на метро в этой группе (в столбце E). В следующих двух столбцах будут записаны логические выражения, определяющее вид поездки. В столбце F будем записывать TRUE для первой поездки в группе (то есть для поездок по тарифу 57 рублей), в столбце G будем записывать TRUE для второй поездки по тарифу «90 минут» (то есть для поездок по

тарифу 28 рублей). Строку 2 таблицы можно заполнить явно, записав в F2 значение TRUE, а в G2 — FALSE.

Поездка будет оплачена по тарифу «единый», то есть будет первой поездкой в группе, при выполнении хотя бы одного условия: после времени начала первой поездки предыдущего билета прошло более 90 минут или количество поездок на метро в предыдущем билете вместе с этой поездкой стало больше 1. Поэтому в ячейку C3 можно написать формулу =OR(C3−D2>90;E2+(B3="M")>1).

Поездка будет оплачена по тарифу 28 рублей, если предыдущая поездка была оплачена по тарифу «единый», а для этой поездки условие оплаты её по тарифу «единый» не было выполнено. Поэтому в ячейку G3 можно записать формулу =AND(F2;NOT(F3)).

Эти формулы используют данные из столбцов D и E предыдущей строки. Теперь пересчитаем эти значения в текущей строке. Они зависят от того, была ли эта поездка первой поездкой по тарифу, то есть от значения в столбце F. В ячейку D3 запишем формулу =IF(F3;C3;D2), в ячейку E3 запишем формулу =IF(F3;0;E2)+(B3="M").

Наконец, посчитаем стоимость поездки. Она будет равна 57 для поездок, у которых записано TRUE в столбце F, или 28 рублей для поездок, у которых записано TRUE в столбце G. Для вычисления этого значения можно записать формулу =F3∗57+G3∗28 в ячейку H3.

Наконец, формулы из ячеек C3:H3 можно скопировать в строки 4–1001 таблицы. После этого числа из столбца H будут ответом на задание.

Задача 5. Очень большая кольцевая линия

Расстояние между станциями с номерами a и b равно $|a-b|$, если не проезжать участок от станции n до станции 1. Если же поехать в другом направлении, то расстояние будет равно $n - |a - b|$. Из этих двух значений нужно выбрать наименьшее.

```
n = int(input())
a = int(input())
b = int(input())
d = abs(a - b)
print(min(d, n - d))
```

Задача 6. Речные прогулки

Набрать 60 баллов можно при помощи перебора по ответу. Переберём все пристани с номерами от 2 до $n - 1$, и для каждой из них посчитаем разность между продолжительностью пути вверх и вниз.

Пусть рассматриваемая пристань имеет номер x, тогда продолжительность пути до пристани 1 равна $a(x - 1)$, а вниз — $b(n - x)$. Нужно найти такую пристань x, для которой модуль разности этих величин будет наименьшим.

Такое решение имеет сложность $O(n)$. Пример такого решения.

```
n = int(input())
a = int(input())
b = int(input())

def ans(x):
    return abs((x - 1) * a - (n - x) * b)

d = 2
for i in range(3, n):
    if ans(i) < ans(d):
        d = i
print(d)
```

Для решения на полный балл заметим, что если некоторая пристань x будет ответом, то значения $a(x - 1)$ и $b(n - x)$ будут близки. Приравняем их, откуда получим решение уравнения

$x = (bn + a)/(a + b)$. Это число было бы ответом, если задача решалась в действительных числах, когда началом маршрутов может быть любая точка. Но мы рассматриваем только целочисленные значения ответа, поэтому ответом может быть одно из двух целых чисел: указанное значение x, округлённое вниз и вверх. Выберем из этих значений такое, для которого модуль разности времени пути до верхней и нижней пристани будет наименьшим, учтя, что ответ не может равняться 1 и n. Такое решение имеет сложность $O(1)$.

```
n = int(input())
a = int(input())
b = int(input())

def ans(pos):
    return abs((pos - 1) * a - (n - pos) * b)

d = max(2, (b * n + a) // (a + b))
if d + 1 < n and ans(d + 1) < ans(d):
    d += 1

print(d)
```

Также полный балл можно было набрать при помощи двоичного или троичного поиска по ответу.

Задача 7. Благоустройство

Задача решается при помощи «жадного алгоритма». Пусть x — координата очередной точки, в которую можно посадить дерево, а предыдущее дерево было посажено в точке с координатой prev. Тогда если x − prev >= d, то посадим дерево в точку x, обновив значение prev = x.

```
d = int(input())
n = int(input())
prev = -d
for i in range(n):
    x = int(input())
    if x - prev >= d:
        print(x)
        prev = x
```