Разбор задач

Задача 1. Разрезание полоски

Чем короче получившиеся числа, тем меньше их вклад в сумму. При этом порядок цифр менять нельзя: можно только выбирать места разрезов.

Сумма всех цифр полоски равна $5 + 2 + 1 + 3 + 8 + 7 + 4 + 9 + 6 = 45$.

Если объединить две соседние цифры a и b в двузначное число $\overline{ab}$, то вместо суммы $a + b$ получится число $10a + b$. Сумма увеличится на $(10a + b) - (a + b) = 9a$.

Поэтому выгодно объединять соседние цифры, у которых первая цифра как можно меньше.

Всего нужно получить шесть частей из девяти цифр. Если три части будут двузначными, а остальные три — однозначными, то сумма будет минимальной среди разбиений такого вида. В качестве первых цифр двузначных чисел по возможности надо выбрать наименьшие цифры.

Выберем пары 21, 38 и 49. Получим разбиение 5, 21, 38, 7, 49, 6.

Для двузначных чисел мы выбрали в качестве первых цифры 2, 3, 4 с суммой 9. Меньшую сумму получить нельзя, потому что если мы выберем двузначное число 13, то тогда нельзя будет выбрать ни цифру 2, ни цифру 3 первыми цифрами двузначных чисел. Двумя другими минимальными возможными двузначными числами тогда будут числа 52 и 49, а сумма первых цифр двузначных чисел будет равна $1 + 4 + 5 = 10$.

Использовать в ответе трёхзначные и более длинные числа невыгодно, так как число сотен в трёхзначном числе даст дополнительную добавку к общей сумме не менее 90.

Ответ: 5, 21, 38, 7, 49, 6.

Задача 2. Редактирование строки

Будем последовательно преобразовывать исходную строку `ABBCBABCBA` в строку `BCBDBABAAD`.

1. Заменим все буквы `C` на `D`.

В целевой строке на местах, где в исходной строке стоят буквы `C`, должны находиться буквы `D`. Поэтому выполним команду `C > D`:

`ABBCBABCBA` — было;
`ABBDBABDBA` — стало;
`BCBDBABAAD` — нужно получить.

2. Заменим пары `AB` на `BC`.

В начале целевой строки стоят символы `BC`. В текущей строке в начале строки стоит пара `AB`. Выполним `AB > BC`:

`ABBDBABDBA` — было;
`BCBDBBCDBA` — стало;
`BCBDBABAAD` — нужно получить.

Теперь первые пять символов совпадают с целевой строкой. Осталось преобразовать окончание строки.

3. Заменим пары `BA` на `AD`.

В конце текущей строки находится пара `BA`. Заменим её на `AD`:

`BCBDBBCDBA` — было;
`BCBDBBCDAD` — стало;
`BCBDBABAAD` — нужно получить.

4. Заменим пары `CD` на `BA`.

В текущей строке есть пара `CD`, а в целевой строке на соответствующих позициях должны стоять буквы `BA`. Выполним `CD > BA`:

`BCBDBBCDAD` — было;
`BCBDBBBAAD` — стало;
`BCBDBABAAD` — нужно получить.

Теперь в строке осталось исправить одну пару BB: вместо неё должны стоять символы BA.

5. Заменим пары BB на BA.

Выполним команду BB > BA:

BCBDBBBAAD — было;
BCBDBABAAD — стало;
BCBDBABAAD — нужно получить.

Требуемая строка BCBDBABAAD получена за пять операций.

Ответ:

```
C > D
AB > BC
BA > AD
CD > BA
BB > BA
```

Задача 3. Мобильный интернет

Пусть у абонента включено в тариф A мегабайт интернета, а стоимость тарифа равна B рублей. Если за месяц использовано не больше A мегабайт, достаточно заплатить только B рублей.

Если трафик M превышает A, необходимо приобрести дополнительные пакеты. В каждом пакете 100 мегабайт, поэтому количество пакетов равно разности $M - A$, делённой на 100 с округлением вверх: $p = \left\lceil \frac{M-A}{100} \right\rceil$.

Округление вверх необходимо потому, что купить часть пакета нельзя. Например, если не хватает даже одного мегабайта, всё равно придётся купить целый пакет.

Таким образом, стоимость интернета равна

$$\begin{cases} B, & M \leqslant A, \\ B + C \left\lceil \dfrac{M-A}{100} \right\rceil, & M > A. \end{cases}$$

Нужно вычислить стоимость для 20 заданных наборов A, B, C, M. Это можно сделать разными способами: используя электронные таблицы, написав программу или выполняя вычисления на калькуляторе.

Решение в электронных таблицах

Вычислим число дополнительных пакетов как частное от деления превышения трафика над включённым объёмом на 100, округлённое вверх. При этом, если трафик не превышает включённый объём, число пакетов должно быть равно нулю. Поэтому в ячейку E2 можно записать формулу: `=B2+ROUNDUP(MAX(0; D2-A2)/100)*C2`. Здесь мы вместо разбора случаев $M \leqslant A$ и $M > A$ используем функцию `MAX(0; D2-A2)`, которая вернёт значение 0, если после вычитания стандартного месячного пакета трафика получится отрицательное число.

В русскоязычном интерфейсе функции называются `ОКРУГЛВВЕРХ` и `МАКС`.

Затем скопируем формулу из E2 в диапазон E2:E21. В этих ячейках появятся ответы для всех 20 наборов данных. Выделим диапазон E2:E21, скопируем его и вставим в поле для ввода ответа.

Решение на языке Python

В Python целое число пакетов можно вычислить с помощью выражения `(M-A+99)//100`, если $M > A$. Чтобы не получать отрицательное число пакетов, когда трафика не больше включённого объёма, используем максимум из нуля и разности $M - A$.

```
packs = max(0, (M - A + 99) // 100)
print(B + packs * C)
```

В этом коде `(M - A + 99) // 100` вычисляет количество пакетов с округлением вверх, когда $M > A$. После этого к стоимости тарифа прибавляется стоимость всех дополнительных пакетов.

Если нужно обработать все 20 наборов данных одной программой, можно считывать четыре числа для каждого набора и выводить рассчитанную стоимость. Например, полностью программа могла бы выглядеть так:

```
for i in range(20):
    A, B, C, M = map(int, input().split())
    packs = max(0, (M - A + 99) // 100)
    print(B + packs * C)
```

На вход этой программе можно было бы подать 20 строк по четыре числа в каждой, скопировав из электронной таблицы блок A2:D21.

Ответ на задание

```
200
320
450
600
1060
2350
5730
11950
33500
82000
133860
246280
636890
1292500
2695770
7945200
25442930
126834150
939658720
997513000
```

Задача 4. Доставка в Простоквашино

Сначала рассмотрим переборное решение. Будем перебирать возможное начальное количество энергии в переменной ans, начиная с нуля. Для каждого значения проверим, сможет ли робот доехать до дома дяди Фёдора. Для этого будем моделировать его движение: сначала уменьшим запас энергии на a, а если он не стал отрицательным, увеличим его на b. Если робот смог совершить все k поездок между соседними зданиями, то текущее значение ans и будет ответом.

Пример такого решения.

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

ans = 0
while True:
    cc = ans
    for i in range(k):
        cc -= a
        if cc < 0:
```

```
            break
        cc += b
    else:
        print(ans)
        break
    ans += 1
```

Это решение перебирает все возможные значения ответа и для каждого из них моделирует движение робота. Если ответ равен x, то время работы составляет $O(xk)$. В худшем случае это слишком долго, поэтому попробуем улучшить алгоритм.

Заметим, что для проверки начального заряда не обязательно моделировать движение робота отдельно для каждого возможного ответа. Вместо этого можно один раз проследить за изменением его энергии, считая, что изначально она равна нулю. При этом в ходе передвижения робота значение его энергии может стать отрицательным, разрешим роботу передвигаться с отрицательной энергией, запоминая минимальное (отрицательное) значение энергии, которое было у робота в ходе поездки.

Введём переменную cnt, которая будет показывать, как изменился запас энергии относительно начального значения. При каждом перемещении робота на расстояние между соседними зданиями будем уменьшать cnt на a, а затем запоминать минимальное значение этой переменной в opt. После этого увеличим cnt на b, поскольку робот может воспользоваться зарядной станцией у достигнутого дома.

Начальный заряд должен быть не меньше модуля минимального значения opt. Действительно, если в некоторый момент значение cnt равно $-x$, то для сохранения неотрицательного запаса энергии к этому моменту необходимо было изначально иметь не менее x единиц энергии.

Пример такого решения.

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

cnt = 0
opt = 0
for i in range(k):
    cnt -= a
    opt = min(opt, cnt)
    cnt += b

print(-opt)
```

В этом решении каждый шаг моделирования выполняется за $O(1)$, поэтому общая сложность составляет $O(k)$. Такое решение проходит тесты с ограничением $k \leqslant 5 \cdot 10^5$, но для полного решения при $k \leqslant 10^9$ его недостаточно.

Рассмотрим теперь решение за $O(1)$.

Если $b \geqslant a$, то после подзарядки запас энергии робота увеличивается или остаётся прежним. Поэтому достаточно изначально зарядить робота на a единиц энергии: он сможет доехать до первого дома, а затем после каждой подзарядки у него будет достаточно энергии для следующего перемещения. Если $a = 0$, робот может добраться до любого дома без начального заряда, что также соответствует этой формуле.

Теперь рассмотрим случай $b < a$. После каждого перемещения и последующей подзарядки запас энергии уменьшается на $a - b$ единиц, то есть заряд робота будет уменьшаться при передвижении к последующему дому. Нам нужно так зарядить робота в начале, чтобы он смог доехать до последнего дома, значит, нужно рассмотреть случай, когда робот доезжает до последнего дома с нулевым зарядом. Суммарно на передвижение ему понадобится ka единиц энергии, но при этом он $k - 1$ раз зарядится на b единиц энергии. Если его первоначальный заряд был равен x, то тогда $x - ka + (k - 1)b = 0$, откуда $x = ka - (k - 1)b$.

Итак, ответ можно вычислить по формуле

$$\text{ans} = \begin{cases} a, & b \geqslant a, \\ ka - (k-1)b, & b < a. \end{cases}$$

Пример полного решения.

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

if b >= a:
    print(a)
else:
    print(k * a - (k - 1) * b)
```

Это решение использует постоянное число арифметических операций, поэтому его временная сложность составляет $O(1)$, а дополнительная память — $O(1)$. По условию $ak \leqslant 2 \cdot 10^9$, поэтому вычисления укладываются в заданные ограничения и решение набирает 100 баллов.

Задача 5. Места в автобусе

В каждом ряду находится четыре места, причём проход разделяет их на две пары. Номера мест в первом ряду — 1, 2, 3, 4, во втором — 5, 6, 7, 8 и так далее. Также заметим, что у прохода находятся места с чётными номерами, а места с нечётными номерами находятся у окна.

Будем считать, что ряды нумеруются с нуля. Тогда номер ряда, в котором находится место с номером x, можно вычислить по формуле (x − 1) // 4, где // — целочисленное деление. Аналогично вычислим номер ряда для места y.

В начале выполнения программы удобно сделать так, чтобы $x < y$. Если изначально $x > y$, поменяем значения переменных местами.

Теперь рассмотрим все четыре категории расположения мест.

1. Идеально удобно.

В этом случае друзья сидят рядом, с одной стороны от прохода. Такие места имеют номера 1 и 2, 5 и 6, 9 и 10 и так далее либо 3 и 4, 7 и 8, 11 и 12 и так далее.

Поскольку $x < y$, для первой пары мест в ряду номер x должен быть нечётным, а номер y должен равняться $x + 1$. Кроме того, оба места должны находиться в одном ряду.

2. Довольно удобно.

В этом случае друзья сидят в одном ряду по разные стороны от прохода, причём оба места находятся у прохода. У прохода находятся места с чётными номерами, поэтому оба значения x и y должны быть чётными, а также места должны находиться в одном ряду.

3. Можно играть.

В этом случае друзья сидят на соседних рядах с одной и той же стороны от прохода. Например, места 1 и 5, а также места 4 и 8.

В этом случае нам неважно, находятся ли места у прохода или у окна, они только должны находиться в соседних рядах и по одну сторону от прохода. Рассмотрим места с номерами x + x % 2 и y + y % 2, где % — операция взятия остатка от деления. В результате преобразования нечётный номер места сменится на номер соседнего с ним места у прохода. То есть места x + x % 2 и y + y % 2 будут находиться оба у прохода, и они будут либо совпадать с местами x и y, либо будут соседними с ними.

Два места у прохода будут находиться друг за другом (то есть в соседних рядах по одну сторону от прохода), если их номера отличаются на 4. То есть в третьем случае нужно проверить условие x + x % 2 + 4 == y + y % 2.

Если ни одно из перечисленных условий не выполняется, расположение мест относится к четвёртой категории.

Полное решение выглядит следующим образом.

```
x = int(input())
y = int(input())
if x > y:
    x, y = y, x
row1 = (x - 1) // 4
row2 = (y - 1) // 4
if row1 == row2 and x % 2 == 1 and y == x + 1:
    print(1)
elif row1 == row2 and x % 2 == 0 and y % 2 == 0:
    print(2)
elif x + x % 2 + 4 == y + y % 2:
    print(3)
else:
    print(4)
```

Задача 6. Кофе помогает от кофе

Обозначим координаты дома через (x_1, y_1), а координаты кофейни — через (x_2, y_2). Требуется найти целочисленные координаты школы (x, y), для которых сумма манхэттенских расстояний от дома до школы и от школы до кофейни равна d:

$$|x - x_1| + |y - y_1| + |x - x_2| + |y - y_2| = d.$$

Если такой точки не существует, необходимо вывести -1.

Для небольших ограничений можно перебрать возможные координаты школы. Если все входные числа не превосходят 500, достаточно рассмотреть координаты x и y от -500 до 1000 включительно.

Действительно, расстояние от дома до школы не может превышать d, поскольку оно является частью суммы, равной d. Поэтому координаты школы отличаются от координат дома не более чем на d. С учётом ограничений на входные данные обе координаты школы лежат в указанном диапазоне.

Переберём все точки (x, y) в этом диапазоне. Для каждой точки вычислим сумму расстояний до дома и кофейни. Если она равна d, выведем координаты найденной точки и завершим программу. Если перебор закончится без результата, выведем -1.

```
import sys

x1 = int(input())
y1 = int(input())
x2 = int(input())
y2 = int(input())
d = int(input())

for x in range(-500, 1001):
    for y in range(-500, 1001):
        dist = abs(x - x1) + abs(y - y1) + abs(x - x2) + abs(y - y2)
        if dist == d:
            print(x, y)
            sys.exit()

print(-1)
```

Перебирается 1501^2 точек. Сложность решения составляет $O(M^2)$, где $M = 1501$ — количество возможных значений каждой координаты.

Можно улучшить перебор, выбрав диапазоны координат в зависимости от входных данных.

Поскольку каждое из четырёх слагаемых в условии неотрицательно, каждое из них не превосходит d. В частности, $|x - x_1| \leqslant d$ и $|x - x_2| \leqslant d$.

Следовательно, координата x должна находиться в диапазоне $\max(x_1, x_2) - d \leqslant x \leqslant \min(x_1, x_2) + d$. Аналогично, для координаты y получаем диапазон $\max(y_1, y_2) - d \leqslant y \leqslant \min(y_1, y_2) + d$.

Переберём все точки в этих диапазонах и проверим условие задачи. Как только найдём подходящую точку, выведем её координаты и завершим программу. Если ни одна точка не подойдёт, выведем -1.

```
import sys

x1 = int(input())
y1 = int(input())
x2 = int(input())
y2 = int(input())
d = int(input())

for x in range(max(x1, x2) - d, min(x1, x2) + d + 1):
    for y in range(max(y1, y2) - d, min(y1, y2) + d + 1):
        dist = abs(x - x1) + abs(x - x2) + abs(y - y1) + abs(y - y2)
        if dist == d:
            print(x, y)
            sys.exit()
print(-1)
```

Сложность решения составляет $O(d^2)$. Такой подход позволяет обрабатывать более крупные значения входных данных, однако при больших ограничениях перебор всё ещё может оказаться слишком медленным.

Решение на 100 баллов. Рассмотрим расстояние между домом и кофейней: $\text{dist} = |x_1 - x_2| + |y_1 - y_2|$.

По неравенству треугольника сумма расстояний от дома до школы и от школы до кофейни не может быть меньше расстояния между домом и кофейней. Поэтому необходимо, чтобы $d \geqslant \text{dist}$.

Есть и второе необходимое условие: числа d и dist должны иметь одинаковую чётность. Действительно, при каждом шаге по горизонтали или вертикали чётность суммы координат меняется. Поэтому чётность длины любого пути между двумя фиксированными точками определяется этими точками. В частности, $d \bmod 2 = \text{dist} \bmod 2$.

Если хотя бы одно из двух условий не выполняется, то ответа не существует, и нужно вывести -1.

Докажем, что при выполнении этих условий ответ всегда существует.

Выберем из дома и кофейни точку с меньшей координатой x. Обозначим её координаты через (x_0, y_0). Если $x_1 \leqslant x_2$, то $(x_0, y_0) = (x_1, y_1)$, иначе $(x_0, y_0) = (x_2, y_2)$.

Если разместить школу в этой точке, сумма расстояний от неё до дома и кофейни будет равна dist. Теперь переместим школу влево на $k = \frac{d - \text{dist}}{2}$ клеток. Это целое неотрицательное число, поскольку $d \geqslant \text{dist}$ и числа d и dist имеют одинаковую чётность.

Так как x_0 не превосходит обеих исходных координат x_1 и x_2, перемещение влево на k клеток увеличит расстояние до каждой из двух точек ровно на k. Координата y при этом не изменится. Следовательно, сумма расстояний увеличится на $2k$ и станет равна $\text{dist} + 2k = \text{dist} + 2 \cdot \frac{d - \text{dist}}{2} = d$.

Таким образом, при выполнении необходимых условий подходящую точку можно построить непосредственно.

```
x1 = int(input())
y1 = int(input())
```

```
x2 = int(input())
y2 = int(input())
d = int(input())
dist = abs(x1 - x2) + abs(y1 - y2)
if d < dist or d % 2 != dist % 2:
    print(-1)
else:
    if x1 <= x2:
        x0, y0 = x1, y1
    else:
        x0, y0 = x2, y2
    x0 -= (d - dist) // 2
    print(x0, y0)
```