Разбор задач

Задача 1. Ракета на старт

В задаче было 4 подзадачи, в каждой из которых требовалось найти наибольшую возрастающую подпоследовательность данной последовательности (НВП).

Для решения задачи можно было воспользоваться так называемой жадной стратегией (например, каждый раз брать очередной подходящий элемент последовательности) вместе с перебором вариантов.

С помощью жадного алгоритма и перебора первого элемента в подпоследовательности можно было получить следующие ответы на подзадачи $1-3$ (возможны и другие варианты правильных ответов):

1. 3 4
2. 1 3 5 8
3. 4 5 6 7 8

В последней подзадаче перебирать только первые элементы в ответе, действуя дальше жадно, недостаточно, а перебрать все варианты для каждого из следующих элементов было бы слишком долго. Чтобы получить ответ максимальной длины, можно было рассуждать так:

- Возьмем 2 первым элементом в нашу НВП (этот элемент первый и минимальный, с него можно начать подпоследовательность с любыми другими элементами).
- Вместо того, чтобы дальше брать 5, возьмем следующую после пятерки 3. Этот ход мотивирован тем, что любой элемент, потенциально следующий после 5 в нашей НВП, также может следовать и после 3. Однако, не каждый элемент после 3 может встать и после 5. Таким образом, выбирая между соседними тройкой и пятеркой, взять тройку выгоднее.
- Далее можно было бы взять 4, потом 7, затем 8 и 9.
- Получаем желаемую подпоследовательность длины шесть: 2 3 4 7 8 9.

Задача 2. Межпланетные грузовые перевозки

В этой задаче необходимо найти максимальную массу перевозимого груза. За одну операцию можно было увеличить грузоподъемность всех уже заказанных кораблей, либо заказать еще один корабль, грузоподъемность которого будет равна y. Можно было сделать не более k операций, причем изначально было заказано x кораблей с начальной грузоподъемностью y.

Решение задачи сильно облегчает следующее наблюдение: поскольку операция увеличения грузоподъемности затрагивает только корабли, заказанные до её выполнения, выгодно сначала заказать определенное количество кораблей, и только после этого увеличивать их грузоподъемность. Таким образом, заказывая k_1 кораблей, можно будет увеличить грузоподъемность каждого из них на $k_2 = k - k_1$.

Тогда максимальная масса перевозимого груза будет равна $(x+k_1)\cdot(y+k_2)$. Остается выбрать k_1, при котором данное произведение будет максимально. Заметим, что сумма $s = x+k_1+y+k_2 = x+y+k$ не зависит от выбора k_1. Обозначим $a = x + k_1$, $b = y + k_2$. Покажем, что произведение a и b максимально, когда значения a и b отличаются друг от друга как можно меньше, а их сумма не меняется. Если s четная, тогда можно считать, что $a = \frac{s}{2} + z$, $b = \frac{s}{2} - z$. Их произведение тем больше, чем меньше величина z. В этом несложно убедиться при перемножении указанных величин. Аналогичный вывод можно сделать и для нечетной суммы (в этом случае оптимально, чтобы перемножаемые величины отличались только на 1).

Тогда мы получаем следующие ответы для каждой из предложенных ситуаций:

Номер ситуации	x	y	k	k_1	k_2	Грузоподъёмность
1	1	1	2	1	1	4
2	3	4	4	2	2	30
3	6	6	7	3	4	90
4	2	8	8	7	1	81

Задача 3. Яблочный пирог

Для того, чтобы составить формулу подсчета количества заготовок яблок, необходимо количество заготовок, расположенных вдоль одной стороны n, умножить на количество заготовок, расположенных вдоль другой стороны m. А чтобы знать, сколько заготовок помещается вдоль каждой стороны, необходимо длину этой стороны поделить на диаметр заготовки. Таким образом формула примет вид $(n/k) \cdot (m/k)$.

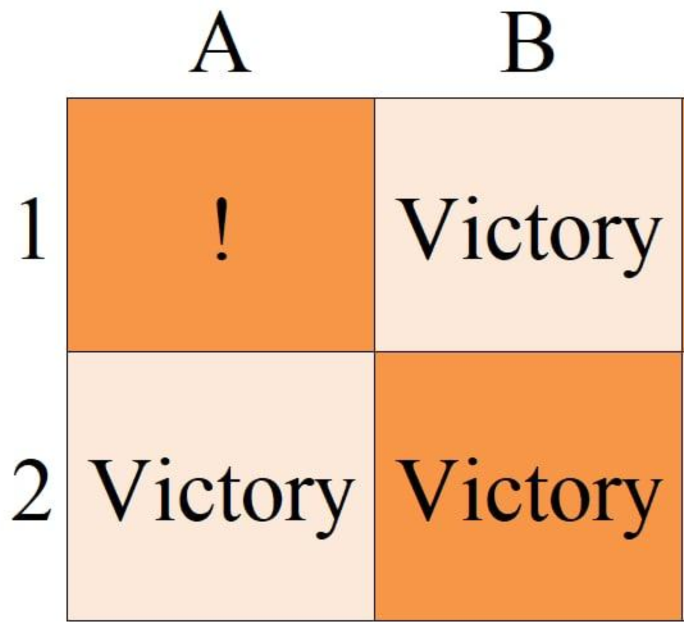
Для расчета количества слив, необходимых для пирога, можно заметить, что слив в каждом ряду и в каждом столбце всегда на 1 меньше, чем заготовок яблок. Ведь сливы помещаются строго между яблоками. Следовательно, формула примет вид $(n/k - 1) \cdot (m/k - 1)$

Значит, один из возможных ответов на задачу следующий:

- $(n/k) * (m/k)$
- $(n/k - 1) * (m/k - 1)$

Задача 4. Петя, Ваня и Леон

Для того, чтобы понять, какие же клетки нижней строки могут привести Петю к победе независимо от игры Вани, необходимо понять, из каких клеток Петя может всегда выиграть, а из каких способен проиграть. Если мы оставим ту же финишную клетку $A1$ и представим, что игра происходит на поле размером 2×2, то увидим, что Петя способен выиграть своим первым же ходом. Посмотрим для этого на рисунок



Отсюда делаем вывод, что клетки $A2$, $B1$, $B2$ – выигрышные для ходящего игрока.

Теперь представим ситуацию с полем размером 3×3. На поле такого размера отметим уже известные нам заранее выигрышные для Пети клетки $A2$, $B1$, $B2$. И, посмотрев на поле, заметим, что есть клетки, из которых можно сходить только в отмеченные выигрышные клетки. Это клетки $C1$ и $A3$. Отмечаем их как проигрышные клетки для ходящего игрока. Соответственно клетки $C2$

и $B3$ становятся выигрышными, так как из них есть ход в проигрышные клетки. А клетка $C3$ становится проигрышной по тому же алгоритму (из неё возможны ходы только в выигрышные клетки). Это можно представить в виде рисунка.

	A	B	C
1	!	Victory	Defeat
2	Victory	Victory	Victory
3	Defeat	Victory	Defeat

Аналогичным образом разбираем решение и для доски 7×7. Если из клетки ход возможен только в выигрышные клетки, то считаем её проигрышной. Если же из клетки есть шанс совершить ход в проигрышную клетку, то считаем её выигрышной. Изобразим наше решение в виде рисунка.

	A	B	C	D	E	F	G
1	!	Victory	Defeat	Victory	Defeat	Victory	Defeat
2	Victory	Victory	Victory	Victory	Victory	Victory	Victory
3	Defeat	Victory	Defeat	Victory	Defeat	Victory	Defeat
4	Victory	Victory	Victory	Victory	Victory	Victory	Victory
5	Defeat	Victory	Defeat	Victory	Defeat	Victory	Defeat
6	Victory	Victory	Victory	Victory	Victory	Victory	Victory
7	Defeat	Victory	Defeat	Victory	Defeat	Victory	Defeat

Из рисунка становится понятно, что в последнем ряду выигрышным клеткам соответствуют столбцы B, D, и F. Это и является правильным ответом.

Задача 5. Поезд

В задаче требовалось найти, сколько раз Василию придется перейти из одного вагона в соседний для того, чтобы встретиться с Петром. Заметим, что это число равно разности между номером вагона Петра и номером вагона Василия.

Как найти номер вагона, в котором находится место с номером p, зная, что в каждом вагоне K мест, и места имеют сквозную нумерацию?

Для этого необходимо понимать, что места с номерами $1 \ldots K$ находятся в первом вагоне, места с номерами $K+1 \ldots 2K$ — во втором, и так далее. Если же нумеровать вагоны с нуля, то номер вагона, в котором находится место p, равно $\lfloor \frac{p-1}{K} \rfloor$.

Тогда ответ на задачу равен $\lfloor \frac{Y-1}{K} \rfloor - \lfloor \frac{X-1}{K} \rfloor$.

Код на языке программирования Python3, соответсвующий решению на 100 баллов:

```python
k = int(input())
x = int(input())
y = int(input())
a = (x - 1) // k
b = (y - 1) // k
print(b - a)
```

Стоит отметить, что в задаче присутствует содержательная группа номер 3, в которой все числа не превосходят 100. Она имеет разве что только косвенное отношение к полному решению, и может быть решена с помощью моделирования действий Василия.

Задача 6. Странное устройство

Для начала разберем решение для подгруппы, где $N \leqslant 20$. В этой подзадаче можно было перебрать все возможные варианты почти любым способом. Для прохождения третьей группы нужно было каким-то образом ускорить перебор (например, используя запоминание ответов для уже найденных меньших значений).

Рассуждая над решением подзадачи номер 1, где $K = 2$, можно было заметить следующее: посмотрим на чётность числа N. Если оно нечетно, то последнее действие, которое было сделано с устройством — это нажатие первой кнопки, увеличивающей число на дисплее на 1. В противном случае, если N чётно, несложно понять, что в оптимальном решении последним действием была нажата вторая кнопка. Таким образом, следующее решение проходит первую подгруппу:

- Если N равно 0, то нажимать на кнопки не требуется.
- Если N нечетно, уменьшаем N на единицу, потому что предыдущим действием могла быть нажата только первая кнопка.
- Если N четно, делим N на 2, поскольку в оптимальном решение предыдущим действием была нажата вторая кнопка.

Ниже приведение код, реализующий идею выше:

```
n = int(input())
k = int(input())
ans = 0
while n > 0:
    if n % 2 == 0:
        n //= 2
    else:
        n -= 1
    ans += 1
print(ans)
```

Поскольку каждое второе действие делит N на 2 (в худшем случае), то будет сделано не более 60 операций.

Для полного решения требовалось обобщить идею, работающую только при $K = 2$. По аналогии, если N не делится на K, то предыдущие $N\ mod\ K$ действий были прибавлением 1 (где $a\ mod\ b$ — остаток при делении a на b). Если же N кратно K, то в оптимальном решении последним действием было умножение на K. Таким образом, алгоритм, решающий задачу на 100 баллов следующий:

- Если N равно 0, то ничего делать не требуется.
- Если N не кратно K, то нужно прибавить к ответу $N\ mod\ K$, а из N вычесть $N\ mod\ K$.
- Если N кратно K, то нужно к ответу прибавить 1, а N поделить на K.

Ниже приведен код, реализующий данный алгоритм.

```
n = int(input())
k = int(input())
ans = 0
while n > 0:
    if n % k == 0:
        ans += 1
        n //= k
    else:
        ans += n % k
        n -= n % k
print(ans)
```

Задача 7. 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 a_{min} (если их несколько, выбирем любой), и удаляем одного из его соседей. При этом к ответу добавляется a_{min}.

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 $a_{min} \cdot (n-1)$.

Таким образом, решение на 100 баллов имеет следующий вид:

```
n = int(input())

mn = 10 ** 9
for i in range(n):
        cur = int(input())
        mn = min(mn, cur)

print(mn * (n - 1))
```