Разбор задач

Задача 1. Защитный экран

Прежде всего необходимо упорядочить данные числа, получим 1, 2, 3, 8, 17, 32, 37.

Используя пластины 1, 2, 3, можно собрать экран любой толщины от 1 до 6. Экран толщины 7 собрать не получится, а используя пластину 8 и меньшие пластины, можно собрать экраны толщины от 8 до 14. Так как следующая пластина имеет толщину 17, экраны толщиной 15 и 16 собрать не получится, а используя пластину 17 и меньшие пластины, можно собрать экраны толщиной от 17 до $17 + 14 = 31$, кроме $17 + 7 = 24$.

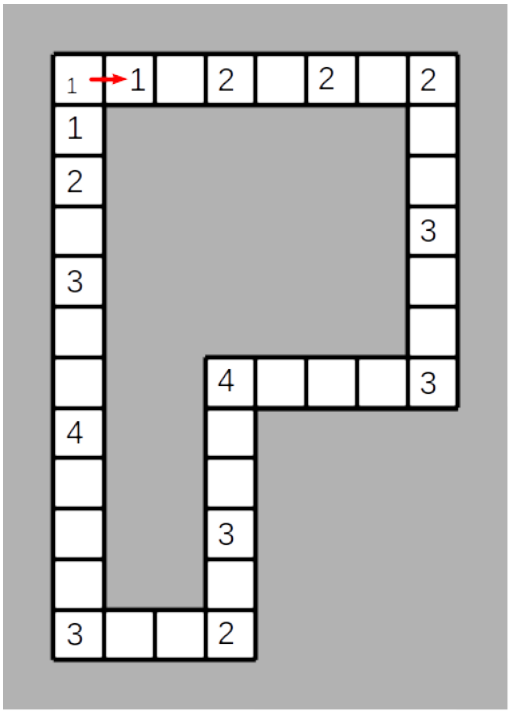
Теперь рассмотрим пластину 32. Вместе с меньшими пластинами с её помощью можно собрать все экраны от 32 до $32 + 31 = 63$, кроме $32 + 7 = 39$, $32 + 15 = 47$, $32 + 16 = 48$ и $32 + 24 = 56$. Итого, при использовании пластин 1, 2, 3, 8, 17, 32 «невозможными» будут комбинации 7, 15, 16, 24, 39, 47, 48, 56.

Теперь рассмотрим пластину 37. С её помощью можно собрать экраны толщиной от 37 до $37 + 63 = 100$, кроме $37 + 7 = 44$, $37 + 15 = 52$, $37 + 16 = 53$, $37 + 24 = 61$, $37 + 39 = 76$, $37 + 47 = 84$, $37 + 48 = 85$ и $37 + 56 = 93$. При этом заметим, что мы смогли получить экраны толщиной 39, 47, 48, 56, которые раньше считались невозможными, а экраны толщиной 44, 52, 53, 61, которые мы смогли получить раньше, всё так же возможны.

Ответ: 7, 15, 16, 24, 76, 84, 85, 93.

Задача 2. Гоночная трасса

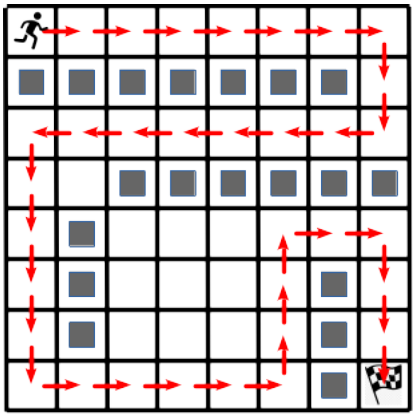
Самый быстрый способ прохождения трассы изображён на рисунке.



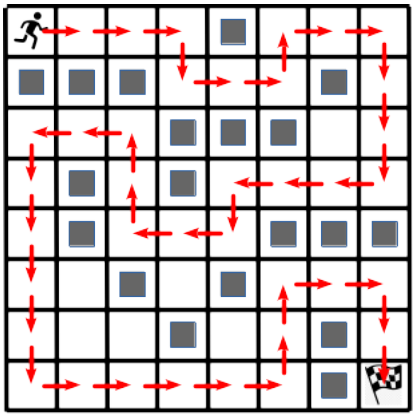
В ответе нужно записать числа 1, 2, 2, 2, 3, 3, 4, 3, 2, 3, 4, 3, 2, 1, 1.

Задача 3. Бег по пересечённой местности

Довольно несложно построить путь из 35 клеток, используя 19 блоков. Такое решение получит 70 баллов.



Но это решение можно улучшить, добавив дополнительные изгибы в тех местах, где происходит движение по прямой. Построим путь из 39 клеток, используя 19 блоков:



Задача 4. Конференция

Задача решается про помощи «жадного алгоритма». Выберем докладчика, которых хочет закончить свой доклад раньше всех. Вторым выберем докладчика, который готов закончить раньше всех из тех докладчиков, которые выступают после первого. Затем выберем докладчика, который заканчивает раньше всех из тех, кто начинает после второго выбранного и т.д.

Для реализации этого решения отсортируем входные данные по столбцу C, то есть по времени окончания доклада. Теперь в ячейку D2 запишем формулу `=IF(B2>=MAX($D$1:D1);C2;"")`. Эта формула запишет в ячейку время окончания выступления этого докладчика (C2), если время начала его выступления (B2) не меньше, чем максимальное время выступления ранее выбранных докладчиков (формула с функцией MAX), иначе это будет пустая строка. Затем эту формулу скопируем в ячейки D3:D1001. Тем самым для докладчиков, которых мы выберем, в столбце D будет записано время окончания их выступления, а для остальных докладчиков — пустая строка.

Осталось только при помощи фильтра выбрать строки с непустыми значениями в столбце D и взять из столбца A номера выбранных докладчиков.

Ответ: 249, 296, 468, 512, 523, 640, 777, 799, 804, 819, 855, 876, 915. Возможны и другие варианты ответа, в лучшем решении выбраны 13 докладчиков.

Задача 5. Пара-тройка конфет

Заметим, что упаковки конфет могут быть двух типов: с двумя шоколадными + одной карамельной или одной шоколадной + двумя карамельными. Также заметим, что ответ не превосходит n, так как в каждой упаковке есть хотя бы одна конфета каждого вида.

При $n \leqslant 10^3$ можно перебрать количество купленных коробок и убедиться, что при любом распределении конфет в купленных упаковках найдётся не менее n конфет одного вида.

Пусть k — количество купленных коробок, из них k_1 — первого типа (две шоколадные и одна карамельная), тогда $k_2 = k - k_1$ — число коробок второго типа (одна шоколадная и две карамельные).

Конфет одного вида будет $\max(2k_1 + k_2, k_1 + 2k_2)$. Для выбранного k переберём значения k_1 такие, что $0 \leqslant k_1 \leqslant k$ и если для всех k_1 верно, что $\max(2k_1 + k_2, k_1 + 2k_2) \geqslant n$, то данное значение k подходит. Минимальное подходящее k является ответом.

Такое решение имеет сложность $O(n^2)$ и набирает 25 баллов. Пример такого решения.

```
n = int(input())

k = 1
ok = False
while not ok:
    ok = True
    for k1 in range(k + 1):
        k2 = k - k1
        if max(2 * k1 + k2, k1 + 2 * k2) < n:
            ok = False
    k += 1
print(k - 1)
```

При $n \leqslant 10^6$ всё ещё получится перебирать значения k, но необходимо избавиться от вложенного цикла, перебирающего возможное распределение упаковок. Заметим, что в худшем случае (чтобы минимизировать количество конфет одного вида), значение $k_1 \approx k_2$, то есть конфет каждого вида окажется примерно поровну.

Заметим, что в двух упаковках точно найдётся три конфеты одного вида. Если мы покупаем чётное число k упаковок, то среди них точно будет $3k/2$ конфет одного вида. Если k нечетное, то количество конфет одного вида будет не меньше, чем $\lfloor 3k/2 \rfloor + 2$. Переберём значения k, пока не найдём минимальное подходящее.

Такое решение имеет сложность $O(n)$ и набирает 50 баллов. Пример такого решения.

```
n = int(input())
k = 0
while k // 2 * 3 + (k % 2) * 2 < n:
    k += 1
print(k)
```

Полное решение имеет сложность $O(1)$. Если n делится на 3, то нужно купить $2n/3$ упаковок (в каждых двух упаковках есть три конфеты одного вида). Если же купить на одну упаковку больше, то в лишней упаковке окажутся 2 одинаковые конфеты. Поэтому для тех значений n, которые не делятся на 3, нужно будет купить ещё одну дополнительную упаковку.

Пример такого решения.

```
n = int(input())
if n % 3 == 0:
    print(n // 3 * 2)
else:
    print(n // 3 * 2 + 1)
```

Задача 6. Речной бой

Будем стрелять последовательно так, чтобы не пропустить корабль. Для этого нужно стрелять, оставляя между выстрелами $k-1$ пустую клетку, то есть в клетки с номерами k, $2k$, $3k$ и т.д., до тех пор, пока не получим результат «ранен», либо такие клетки не кончатся. Мы сделаем не более, чем $\lfloor n/k \rfloor$ выстрелов (целочисленное частное от деления n на k). Получив на каком-то выстреле ответ «ранен», начнём стрелять в соседние клетки в одну сторону, а при получении результата «мимо» — в другую сторону. Тем самым мы сделаем не более одного дополнительного промаха, то есть для попадания в $k-1$ клетку корабля понадобятся не более k выстрелов, и общее число выстрелов будет равно $\lfloor n/k \rfloor + k$.

Но если n делится на k, то последний выстрел из первой последовательности придётся в последнюю клетку поля, и тогда с одной стороны от этой клетки больше не останется клеток. В этом случае достаточно только $k-1$ дополнительных выстрелов, то есть при n, делящемся на k, ответ равен $\lfloor n/k \rfloor + k - 1$.

Используя целочисленное деление, оба этих случая можно записать одной формулой.

```
n = int(input())
k = int(input())
print((n - 1) // k + k)
```

Задача 7. Сломанный индикатор

Для начала формализуем условие задачи. Некий индикатор состоит из 9 сегментов, каждый из которых либо гарантированно работает (зажигается какой-то цифрой из данного набора), либо имеет неизвестное состояние (не зажигается ни одной из данных цифр). Задача заключается в нахождении всех цифр, которые состоят только из гарантированно рабочих индикаторов.

Пронумеруем сегменты индикатора сверху вниз, слева направо, то есть самый верхний сегмент будет иметь номер 1, а самый нижний — 9. Для каждой возможной цифры перечислим сегменты, из которой она состоит. Например, цифра 0 состоит из сегментов 1, 2, 4, 6, 8, 9.

Если какая-то цифра могла быть отображена на экране, то соответствующие ей сегменты заведомо исправны. Если несколько цифр могли быть отображены на экране, то исправны сегменты, которые входят в хотя бы одну данную цифру. То есть нужно сделать объединение множеств сегментов, соответствующих данным цифрам. Это удобно реализовать при помощи структуры данных «множество» в языке Python, но можно использовать и массивы из нулей и единиц (где единица будет соответствовать исправному сегменту индикатора) или битовые операции, но тогда реализация станет сложнее.

Для проверки, что какую-то цифру гарантированно удастся показать на индикаторе, множество сегментов этой цифры должно быть подмножеством заведомо исправных сегментов. Для этого в Python есть операция `<=` для множеств.

Приведём пример решения на языке Python. Здесь segments — это список множеств, соответствующих цифрам, то есть segments[i] — это список сегментов, входящих в цифру i.

После нахождения объединения множеств сегментов переберём все цифры от 0 до 9 и выведем подходящие из них.

```
segments = [
{1, 2, 4, 6, 8, 9},
{3, 4, 8},
{1, 4, 7, 9},
{1, 3, 5, 7},
{2, 4, 5, 8},
{1, 2, 5, 8, 9},
{3, 5, 6, 8, 9},
{1, 3, 6},
{1, 2, 4, 5, 6, 8, 9},
{1, 2, 4, 5, 7}]

res = set()
n = int(input())
for i in range(n):
    d = int(input())
    res |= segments[d]

for d in range(0, 10):
    if segments[d] <= res:
        print(d)
```

Для получения частичного балла при $2 \leqslant a_i \leqslant 4$ достаточно было вручную исследовать, какие ответы возникают на 7 возможных тестах этой группы, занести их в программу и выводить нужный ответ в зависимости от ввода.