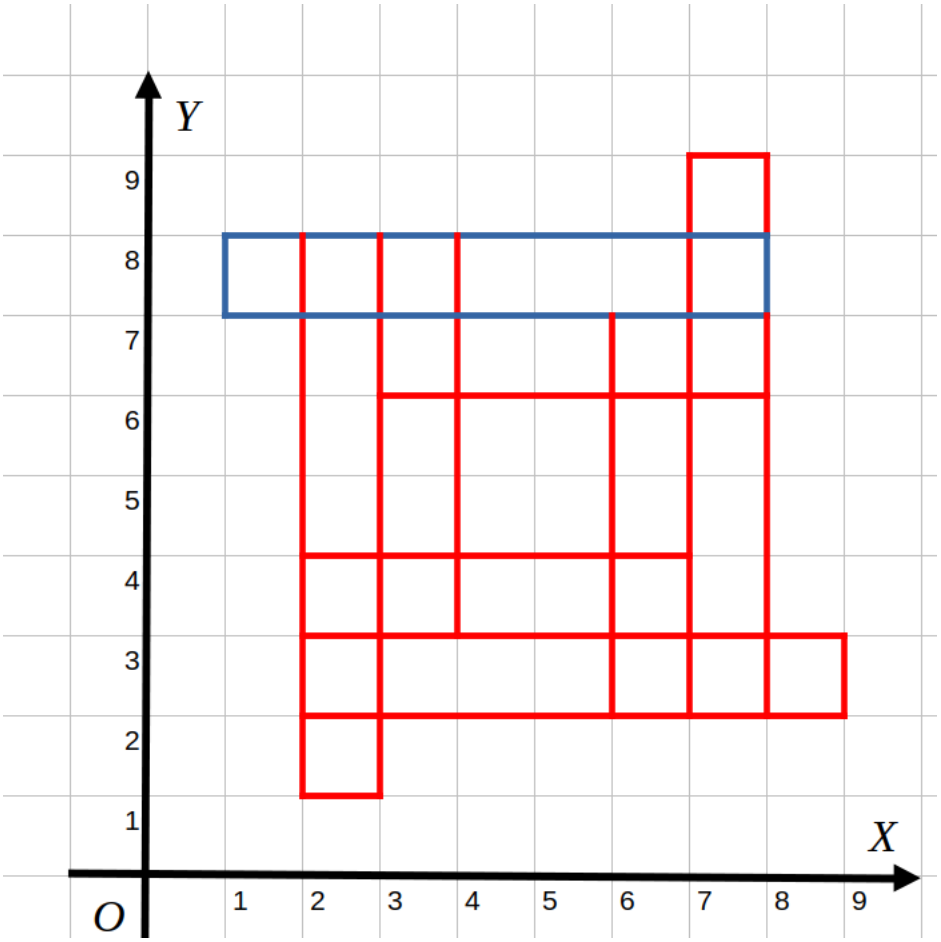
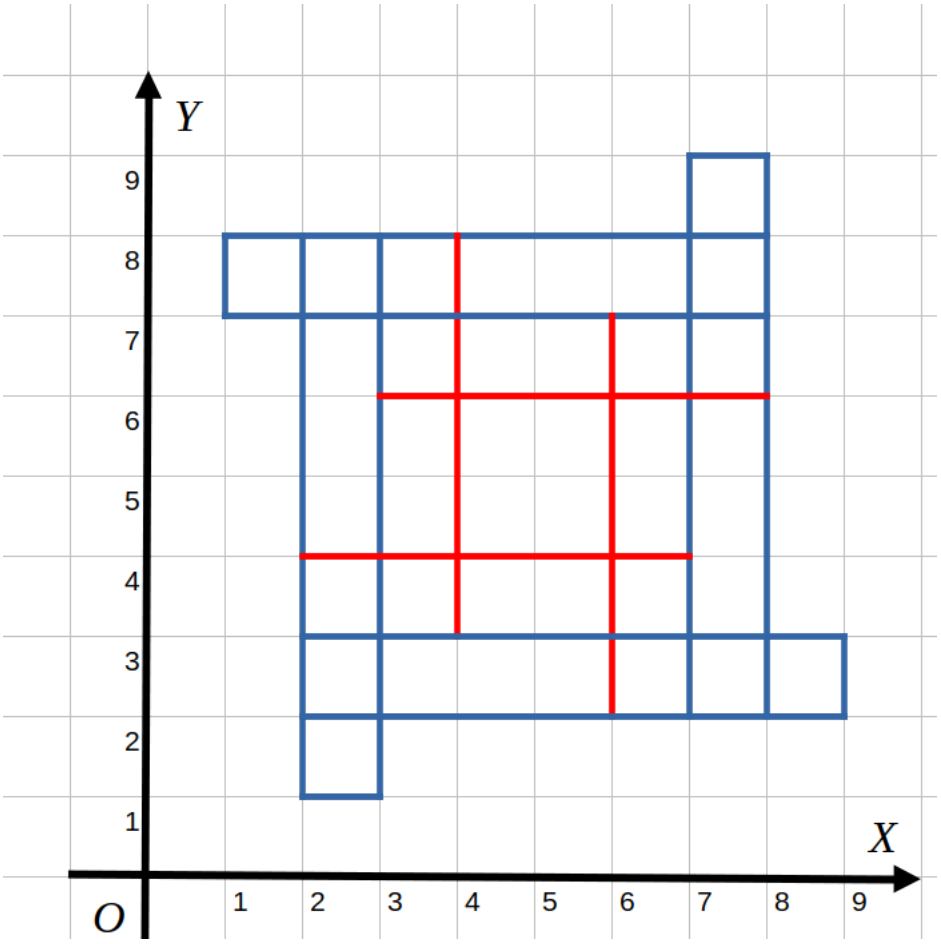
Задача 1. Абстрактный плакат

Можно заметить, что картинка совпадает с собой при повороте на 90°, поэтому количество прямоугольников в ответе будет делиться на 4. Для начала нарисуем выступающий единичный квадрат, сделаем этот прямоугольник, чтобы нарисовать всю верхнюю горизонтальную сторону (без выступающего сверху квадрата).



Нарисуем ещё три прямоугольника, полученные из данного поворотами.



Оставшиеся линии можно нарисовать при помощи четырёх прямоугольников. Всего достаточно 8 прямоугольников.

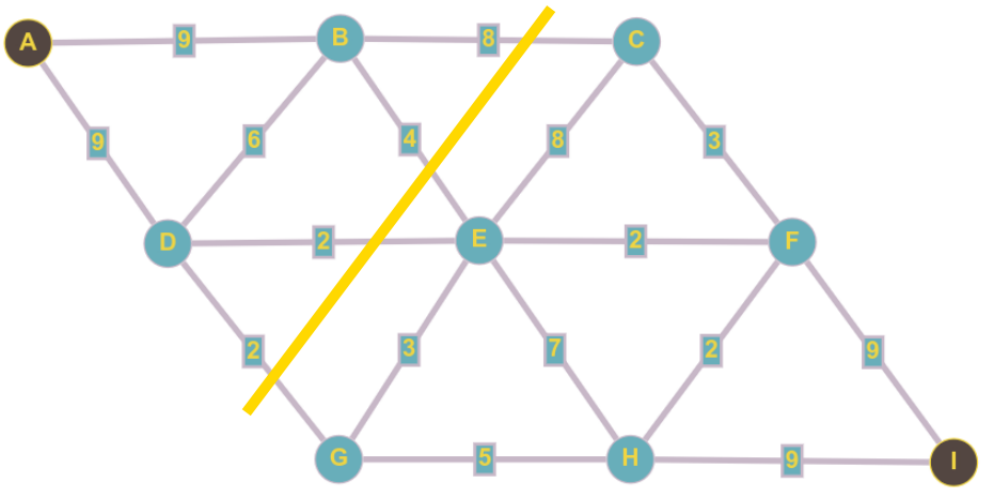
Пример такого ответа:

```
1 7 8 8
7 9 8 2
2 2 9 3
2 1 3 8
3 3 4 8
3 6 8 7
6 2 7 7
2 3 7 4
```

Задача 2. Максимальный поток

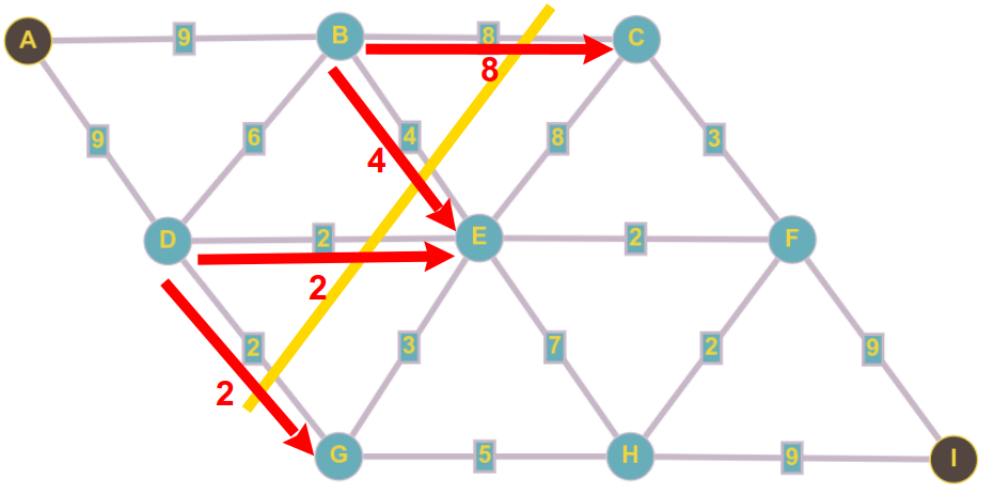
В терминах теории графов узлы будем называть вершинами, трубы — рёбрами, пропускную способность ребра — его весом. А такая задача известна, как задача нахождения максимального потока в графе.

Сумма весов рёбер исходящих из истока A и входящих в сток I равна 18, поэтому поток не может быть больше 18. Но посмотрим жёлтую линию («разрез»), отделяющий вершины B и D от вершин C, E, G. Сумма весов рёбер, которые пересекает этот разрез, равна 16, поэтому величина потока не может быть больше 16.



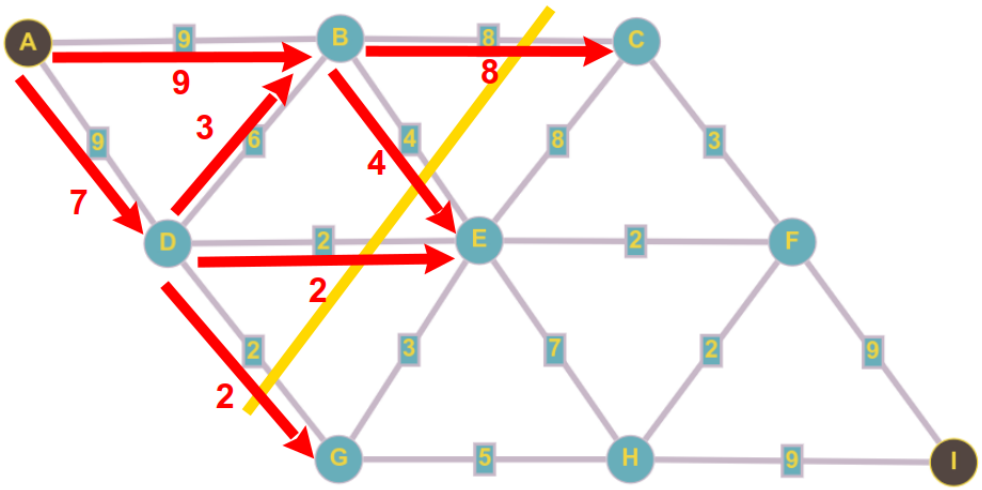
Чтобы получить максимальный результат нужно «насытить» эти рёбра, то есть пустить по ним максимальный объём воды. Получим такую часть ответа.

```
B C 8
B E 4
D E 2
D G 2
```

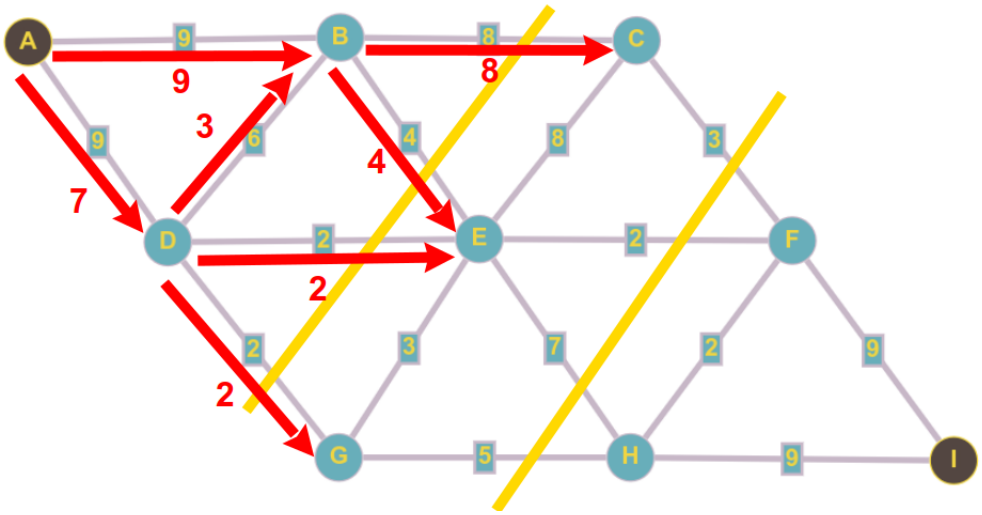


Объём вытекающей из B воды равен 12, поэтому необходимо сделать сумму весов входящих в B рёбер тоже равным 12, для этого нужно будет подать из D в B объём 3 и насытить ребро AB. По ребру AD нужно будет подать 7, из которых 3 пойдёт в вершину B и по 2 — в вершины E и G.

```
A B 9
A D 7
D B 3
```

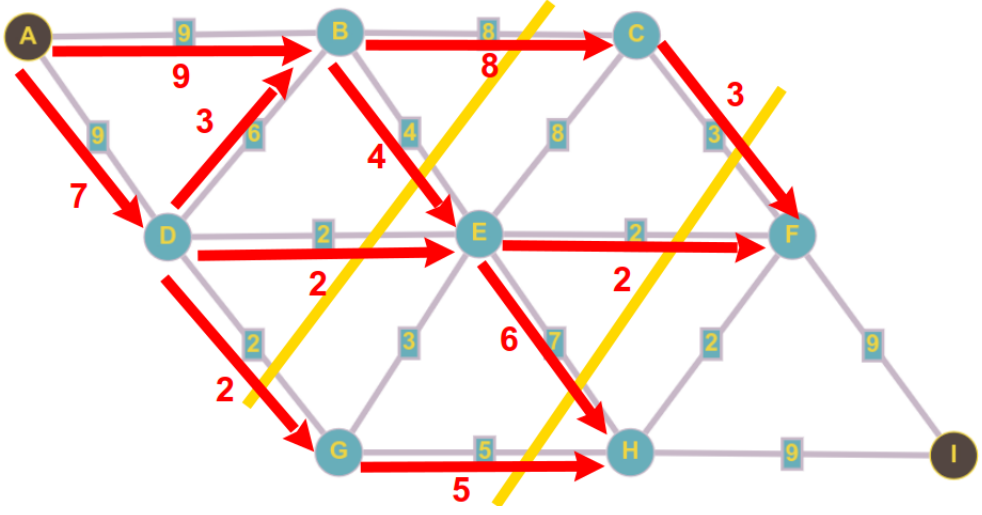


Теперь посмотрим на второй жёлтый разрез, отделяющий вершины C, E, G от вершин F и H.



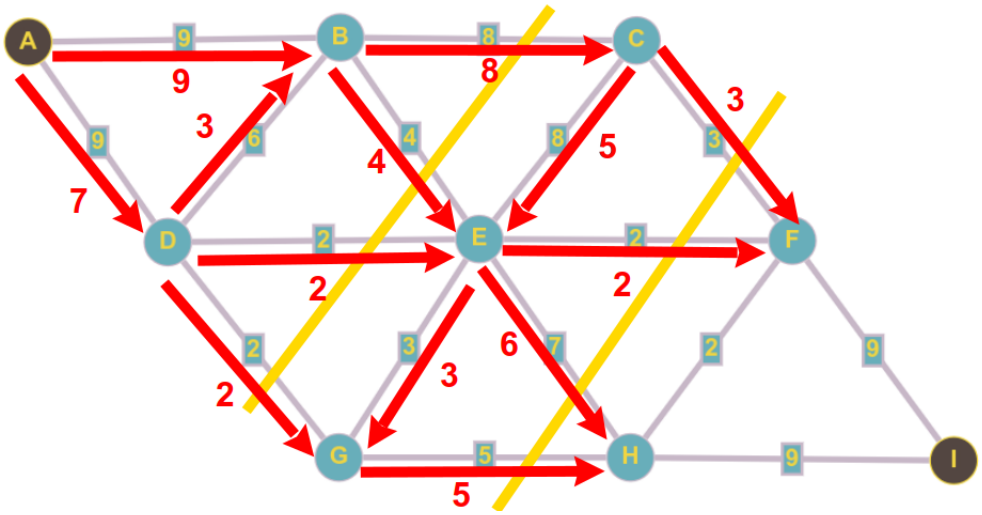
Сумма весов рёбер, пересекающих разрез, равно 17, значит, нужно насытить все рёбра, кроме одного. Например, пустим через ребро EH объём 6.

```
C F 3
E F 2
E H 6
G H 5
```



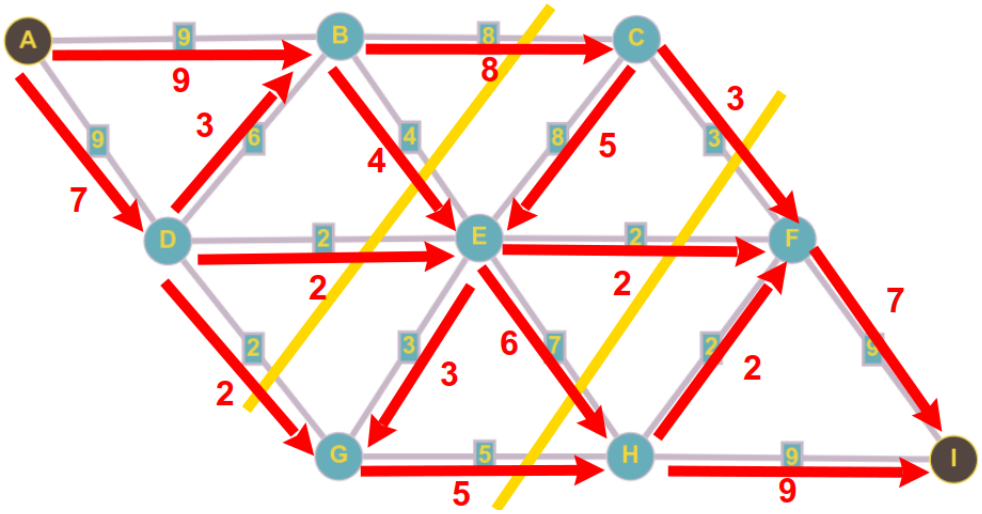
Теперь направим воду из C в E и из E в G так, чтобы обеспечить нужный объём входящей воды в E и G:

```
C E 5
E G 3
```



Наконец, распределим воду по последним рёбрам. Поскольку в H втекло 11, а вытечь может не более 9, то нужно направить 2 единицы воды из H в F.

```
H F 2
H I 9
F I 7
```



Задача 3. Телефонный справочник

Давайте рассматривать символы по одному, начиная с первого. Чтобы получить как можно меньшую строку, мы должны найти самый первый символ, который можно заменить на меньший, то есть правее которого есть меньший.

Рассмотрим первое задание “cfwvfu”. Для символов “c” и “f” правее нет меньшего символа, а для символа “w” — есть, это предпоследний символ “f”. Получим ответ “cffvwu”.

Рассмотрим второе задание “tbzttbetcb”. Здесь самый первый символ “t” можно заменить на меньший. Но чтобы строка была лексикографически наименьшей, мы должны выбрать справа наименьший из возможных символов для замены, это символ “b”. Но и символов “b” в строке несколько, поскольку выбранный символ “b” мы заменим на больший, то нужно использовать для перестановки последний символ “b”. Получим ответ “bbzttbetct”.

В третьем задании дана строка “aefhfifjfkflzhz”. Здесь самый первый символ, правее которого есть меньший — это “h”. Поменяем его с последним вхождением символа “f”, получим ответ “aefffifjfkhlzhz”.

В четвёртом задании “abcdfjhklmnqrtuvwzyx” можно заменить символ “j” на меньший символ ”h”, стоящий правее него. Получим “abcdfhjklmnqrtuvwzyx”.

Задача 4. Большая команда

В первом задании школ мало и ответ можно найти просто изучив входные данные, он равен 14.

Во втором задании школ уже довольно много, поэтому понадобится провести некоторые вычисления. Например, запишем в ячейку D1 количество участников в команде. В ячейку D2 запишем формулу для вычисления количества команд из школы в строке 2: `=INT($A2/D$1)` — нужно поделить количество учащихся в школе на размер команды и взять целую часть. Теперь скопируем эту формулу на весь столбец D. Посчитаем сумму чисел в столбце D, то есть количество получившихся команд. Для этого можно использовать формулу типа `=SUM(C2:C1001)`. Будем менять значение в ячейке D1, пока эта сумма не станет больше или равна необходимого количества команд. Это произойдёт при размере команды, равном 22.

Аналогично можно решить и оставшиеся задания, только сложность будет с перебором значения размера команды. В задании 3 можно заполнить ячейки в строке 1 возможными значениями команды: 1, 2, 3, и т.д. А в столбце ниже него посчитать количество команд, которое получится при данном размере. Максимальный размер команды, при котором наберётся нужное количество команд, будет равен 379.

В четвёртом задании числа столь большие, что такой подход потребует слишком много вычислений. Можно подобрать нужный размер, например, сначала увеличивая размер команды с шагом 1000. Затем определив диапазон для ответа из 1000 чисел, заполнить строку с размером команды с шагом 1. Или можно интерактивно менять значение в ячейке, увеличивая или уменьшая его, сокращая диапазон поиска. Например, сначала определить, что ответ находится от 200 000 до 300 000. Потом подобрать вторую цифру ответа, ответ будет находиться от 210 000 до 220 000. Затем подобрать третью цифру и т.д. Ответ на четвертое задание — 218 976.

Задача 5. Ферзь

Чтобы набрать 40 баллов достаточно написать решение, перебирающее все клетки, на которые можно поставить ферзя. Затем нужно посчитать количество клеток, которые бьёт этот ферзь. Для этого также переберём все оставшиеся клетки доски, и проверим, находятся ли они в одной горизонтали, вертикали или диагонали с выбранной клеткой. Запомним наибольшее количество клеток, которое может бить ферзь. Сложность такого решения будет $O(n^2m^2)$. В примере такого решения строки и столбцы доски нумеруются для удобства с нуля.

```
n = int(input())
m = int(input())
ans = 0
for x in range(n):
    for y in range(m):
        count = 0
        for xx in range(n):
            for yy in range(m):
                if x==xx or y==yy or x-xx==y-yy or x-xx==yy-y:
                    count += 1
        ans = max(ans, count - 1)
print(ans)
```

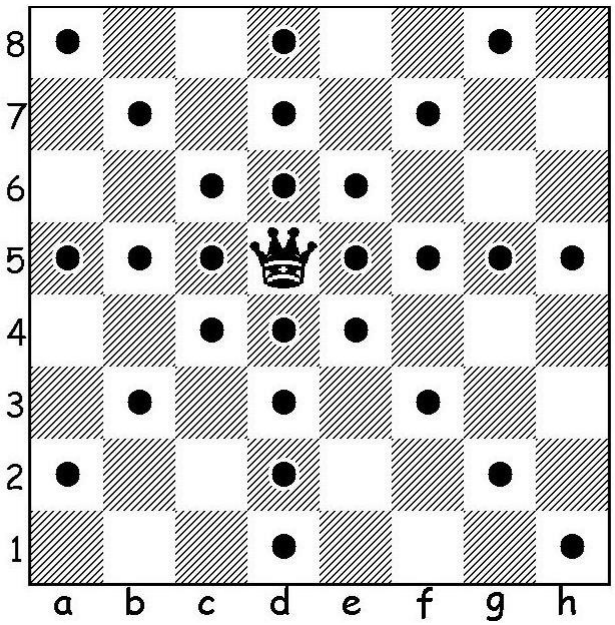
Чтобы набрать 80 баллов, нужно уменьшить сложность решения до $O(nm)$. Для этого можно находить количество клеток, которые бьёт ферзь, без цикла, то есть за $O(1)$. Или, наоборот, заметить, что ферзя нужно поставить в центр доски, и найти циклами количество клеток, которые он бьёт. Пример второго решения, где ферзь ставится в клетку с координатами $(\lfloor n/2 \rfloor, \lfloor m/2 \rfloor)$.

```
n = int(input())
```

```
m = int(input())
x = n // 2
y = m // 2
count = 0
for xx in range(n):
    for yy in range(m):
        if x==xx or y==yy or x-xx==y-yy or x-xx==yy-y:
            count += 1
print(count - 1)
```

Полное решение имеет сложность $O(1)$. Здесь нужно заметить, что ферзь контролирует наибольшее количество клеток, находясь в центре доски, и посчитать их количество, без использования циклов.

Пусть $n \leqslant m$, то есть доска "вытянута" по горизонтали, иначе поменяем значения n и m. Тогда в одной горизонтали с ферзём находятся $m-1$ клетка, а в одной вертикали — $n-1$ клетка. Диагонали, проходящие через ферзя, также могут содержать не более n клеток (поэтому в каждой из них не более $n-1$ клетки, за вычетом клетки, в которой стоит ферзь), поэтому ответ будет равен $(m-1)+3\cdot(n-1)$. Но есть одно исключение: на квадратной доске, сторона которой имеет чётную длину, одна из диагоналей будет короче на одну клетку. Это, например, случай доски 8×8 (первый пример из условия), для которой ответ равен 27, а не 28. Почему так происходит, можно видеть на рисунке.



В этом случае просто вычтем 1 из ответа. Пример такого решения.

```
n = int(input())
m = int(input())
if n > m:
    n, m = m, n
ans = 3 * n + m - 4
if n % 2 == 0 and m == n:
    ans -= 1
print(ans)
```

Задача 6. Рамка для рисунка

Начнём с переборных решений, набирающих частичные баллы. Пусть n_1 и n_2 — количество палочек длины 1 и 2 соответственно.

40 баллов можно набрать, если перебирать две стороны прямоугольника a и b. Проверим, можно ли сложить из имеющихся палочек прямоугольник $a \times b$. Должны выполняться два условия: общая длина всех палочек $n_1 + 2n_2$ должна быть не меньше периметра прямоугольника, равного $2a + 2b$, и каждая сторона нечётной длины должна содержать хотя бы одну палочку длины 1, поэтому значение n_1 должно быть не меньше количества нечётных чисел среди сторон a, b, a, b. Пример такого решения.

```
n1 = int(input())
n2 = int(input())
max_side = (n1 + 2 * n2) // 2
ans = 0
for a in range(1, max_side + 1):
    for b in range(1, max_side + 1):
        if n1 + 2 * n2 >= 2 * a + 2 * b and n1 >= 2 * (a % 2 + b % 2):
            ans = max(ans, a * b)
print(ans)
```

Чтобы набрать 60 баллов нужно перебирать только одну сторону, а не две. Пусть это сторона a. Максимальное значение одной стороны, как и в предыдущем решении, равно $\lfloor \frac{n_1+2n_2}{2} \rfloor$ (целочисленное частное от деления суммы длин всех палочек на 2). Тогда у нас будет две стороны длины a. Проверим условие, что n_1 не меньше, чем $2(a \bmod 2)$, то есть если a — нечётное, то найдётся хотя бы две палочки длины 1.

Теперь определим наибольшее подходящее значение b для данного значения a. Для этого посчитаем длину оставшихся палочек $n_1 + 2n_2 - 2a$ и поделим её на 2. При этом могло оказаться, что $n_1 < 2(a \bmod 2 + b \bmod 2)$, то есть значение n_1 оказалось меньше, чем число нечётных чисел среди значений a, b, a, b, то нам не хватит палочек длины 1 для того, чтобы собрать нечётные отрезки длины a, b, a, b. Но ранее мы проверили, что нам хватает палочек длины 1 для того, чтобы собрать только отрезки a и a, поэтому это возможно только в случае нечётного b. В этом случае уменьшим значение b на 1. Так мы определяем наибольшее значение второй стороны b для ранее выбранной стороны a. Запомним наибольшее из значений площадей прямоугольников $a \times b$.

```
n1 = int(input())
n2 = int(input())

max_side = (n1 + 2 * n2) // 2

ans = 0
for a in range(1, max_side + 1):
    if n1 < 2 * (a % 2):
        continue
    b = (n1 + 2 * n2 - 2 * a) // 2
    if n1 < 2 * (a % 2 + b % 2):
        b -= 1
    ans = max(ans, a * b)
print(ans)
```

Чтобы написать полное решение, нужно избавится и от перебора всех возможных значений одной стороны. Нужно заметить, что среди всех прямоугольников с одинаковым периметром максимальная площадь будет у квадрата или у прямоугольника, стороны которого различаются на 1. Действительно, пусть прямоугольник имеет стороны $a \times b$, при этом $a + 1 < b$. Рассмотрим прямоугольник $(a + 1) \times (b - 1)$ с таким же периметром. Его площадь будет равна $ab + b - a - 1$, то есть больше площади ab.

Поэтому для максимизации периметра в качестве значения одной из сторон нужно выбрать $\frac{1}{4}$ от максимально возможного периметра. Для этого в качестве значения минимальной стороны a возьмём значение $\lfloor \frac{n_1+2n_2}{4} \rfloor$, а значение b подберём наибольшее подходящее значение, как в предыдущем

решении. Но если значение a оказалось нечётным, то для формирования сторон длины a понадобится минимум 2 палочки длины 1, и, возможно, нам не хватит палочек длины 1 для достижения максимально возможного значения b. Поэтому необходимо рассмотреть как чётное, так и нечётное значение a, то есть нужно взять не только $a = \lfloor \frac{n_1+2n_2}{4} \rfloor$, но и значение на 1 меньше — $a = \lfloor \frac{n_1+2n_2}{4} \rfloor$, так как одно из них будет чётным. Для каждого из этих значений a выберем подходящее b и найдём наибольшее значение $a \times b$.

Пример такого решения.

```
n1 = int(input())
n2 = int(input())
ans = 0
side = (n1 + 2 * n2) // 4
for a in range(side - 1, side + 1):
    if n1 < 2 * (a % 2):
        continue
    b = (n1 + 2 * n2 - 2 * a) // 2
    if n1 < 2 * (a % 2 + b % 2):
        b -= 1
    ans = max(ans, a * b)
print(ans)
```

Есть и другие способы решения задачи. Например, можно попробовать конструктивно построить решение так, чтобы две стороны прямоугольника $a \times b$ оказались максимально большими и при этом близки друг к другу. Для этого сначала разложим палочки длины 2 поровну на все стороны. У нас останется 0, 1, 2 или 3 палочки длины 2. Если осталось хотя бы две палочки длины 2, то увеличим на 2 длины двух сторон a. Если после этого осталась хотя бы одна палочка длины 2 и ещё одна палочка длины 2 или две палочки длины 1, то также можно сторону b увеличить на 2. Иначе попробуем используя палочки длины 1 выровнять длины сторон. После выравнивания длин сторон все оставшиеся палочки длины 1 разложим поровну по всем сторонам. После этого останется не более трёх палочек длины 1, если их две или три — то можно длины двух сторон увеличить на 1.

Сложность реализации такого решения в том, что нужно аккуратно рассмотреть все случаи, не пропустив ни одного.

```
n1 = int(input())
n2 = int(input())

a = 2 * (n2 // 4)
b = 2 * (n2 // 4)
n2 %= 4

if n2 >= 2:
    n2 -= 2
    a += 2
if n2 == 1 and n1 >= 2:
    b += 2
    n2 = 0
    n1 -= 2
while a < b and n1 >= 2:
    a += 1
    n1 -= 2
while a > b and n1 >= 2:
    b += 1
    n1 -= 2
a += n1 // 4
b += n1 // 4
```

```
n1 %= 4
if n1 >= 2:
    a += 1
print(a * b)
```

Задача 7. Задачи на печать!

Будем рассматривать задачи последовательно, определяя, в каком режиме должна быть напечатана очередная задача. Например, если в задаче 2 страницы, то её условие должно быть напечатана только в двустороннем режиме. А если в задаче 1 страница, то всё зависит от того, в каком режиме была напечатана предыдущая страница. Если это был односторонний режим, то мы расширим предыдущий диапазон печати на новую задачу, а если двусторонний — то эту одну страницу можно напечатать в двустороннем режиме, однако, следующая страница обязательно должна стать началом нового диапазона печати, который может быть как односторонним, так и двусторонним.

Заведём переменную *mode*, в которой будет храниться текущий режим печати — 1 для односторонней печати и 2 для двусторонней. Значение 0 означает, что очередная страница должна стать началом нового диапазона печати, который может быть любым. В переменной *ans* хранится общее число диапазонов печати. В переменной p хранится количество страниц в текущей задаче.

Далее нужно аккуратно разобрать все случаи. Если $p = 2$ то мы обязательно переходим в режим 2, при этом если ранее режим был другим, то к ответу прибавляем 1.

Если $p = 1$, то в режиме 1 не нужно делать ничего, в режиме 2 эта страница печатается в двустороннем режиме, но нужно перейти в режим 0 для начала нового диапазона со следующей страницы (потому что на обороте этой страницы ничего нельзя печатать), а в режиме 0 нужно перейти в режим 1, начав новый диапазон, то есть добавив к ответу 1.

Наконец, разберём случай $p = 3$. Если до этого был режим 1, то мы печатаем одну страницу односторонней печатью, а ещё две страницы — двусторонней, поэтому нужно перейти в режим 2. Если мы были в режиме 2, то все три страницы можно напечатать в двустороннем режиме, потом нужно перейти в режим 0, потому что придётся начать новый диапазон. Аналогично поступим, когда режим был равен 0 — начнём новый двусторонний режим, напечатаем три страницы, а затем придётся начать новый диапазон, то есть в этом случае нужно просто увеличить значение *ans* на 1, сохранив значение *mode* равным 0.

Пример такого решения.

```
ans = 0
mode = 0
n = int(input())
for i in range(n):
    p = int(input())
    if p == 1:
        if mode == 2:
            mode = 0
        elif mode == 0:
            mode = 1
            ans += 1
    if p == 2:
        if mode != 2:
            ans += 1
            mode = 2
    if p == 3:
        if mode == 1:
            mode = 2
            ans += 1
        elif mode == 2:
            mode = 0
```

```
        else:
            ans += 1
print(ans)
```

Частичные решения можно получить используя полный перебор вариантов.

Задачу можно решить и динамическим программированием, в котором целевой функцией $f(i)$ будет количество диапазонов, необходимое для печати первых i задач. При этом придется ввести и второй параметр, аналогичный по смыслу переменной *mode*.