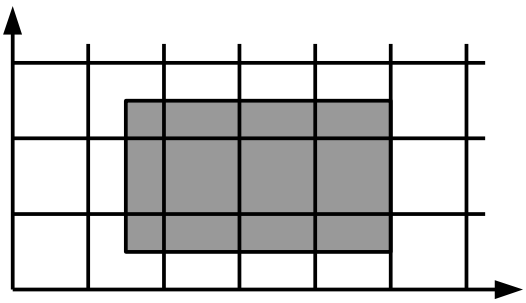
Муниципальный этап всероссийской олимпиады школьников по информатике
Москва, 16 декабря 2018 г.
Разбор заданий для 9–11 классов

Задача 1. Плитка

Стена покрыта квадратной плиткой со стороной M см. На стену повесили картину, известны координаты левого нижнего угла картины, её ширина и высота. Определите количество плиток, которые оказались частично или полностью закрыты картиной.

Первая строка входных данных содержит число M — сторону плитки. Вторая и третья строки содержат числа X и Y — координаты левого нижнего угла картины. Третья и четвёртая строки содержат числа W и H — ширину и высоту картины. Ось OX направлена вправо, ось OY направлена вверх. Все числа целые, не превосходящие 2×10^9, числа M, W, H — положительные, числа X и Y — положительные или равны 0.

Программа должна вывести одно число — количество плиток, полностью или частично закрытых картиной. Плитка считается закрытой картиной, если пересечение картины и плитки имеет ненулевую площадь, то есть касание картины и плитки не считается перекрытием.



Система оценивания

Решение, правильно работающее только для случаев, когда все входные числа не превосходят 100, будет оцениваться в 40 баллов.

Решение, правильно работающее только для случаев, когда все входные числа не превосходят 10^5, будет оцениваться в 70 баллов.

Решение

Разобьём плоскость на плитки (квадраты со стороной M) и пронумеруем столбцы и строки квадратов начиная с 0. Сначала определим, в какой столбец плиток попадёт левая сторона картины, в какой столбец попадёт правая сторона картины, в какой ряд плиток попадёт нижняя сторона картины, в какой ряд плиток попадёт верхняя сторона картины. Для этого нужно координаты сторон плиток поделить на M. Левая сторона картины имеет координату X, правая — $X + W$, нижняя — Y, верхняя — $Y + H$. При этом при определении номера столбца левой стороны и номера строки нижней стороны нужно делить на M с округлением вниз, а при определении номера столбца правой стороны и номера строки верхней стороны нужно делить на M с округлением вверх. Затем определяем, количество столбцов и строк, которое занимает картина и перемножаем два полученных числа.

Отметим, что в языках Pascal, C++, Java для получения полного балла необходимо проводить вычисления с использованием 64-битных целых чисел, т. к. при использовании обычных целых чисел происходит переполнение целочисленной переменной при вычислении.

Пример решения на языке Python.

```
M = int(input())
X = int(input())
Y = int(input())
W = int(input())
H = int(input())
left = X // M
right = (X + W - 1) // M
bottom = Y // M
top = (Y + H - 1) // M
print((right - left + 1) * (top - bottom + 1))
```

Задача 2. Римские числа

Один древнеримский торговец брал несколько раз ссуду в древнеримском банке. Каждый раз банкир записывал размер выданной ссуды на листе пергамента, используя римские числа. Но ввиду дороговизны пергамента, запись производилась плотно и все числа оказались записанными подряд, без разделителей. Когда торговец пришёл возвращать ссуду, оказалось, что невозможно установить разбиение записи на числа.

Например, если на пергаменте записана строка «XIIV», её можно разбить на римские числа разными способами, например, XI + IV = 11 + 4 = 15 или XII + V = 12 + 5 = 17, возможны и другие варианты разбиения.

Торговец хочет вернуть как можно меньше денег, поэтому он хочет так разбить строку цифр на римские числа, чтобы сумма всех чисел была как можно меньше.

Программа получает на вход строку, длина которой не превосходит 250 символов. Строка состоит только из заглавных латинских букв I, V, X, L, C, D, M.

Программа должна вывести единственное число — минимальную возможную сумму, которую можно получить при разбиении данной строки на последовательность корректных римских чисел. Ответ нужно вывести арабскими цифрами в десятичной системе счисления.

Правила записи римских чисел

Римскими цифрами можно записать целые числа от 1 до 3999. Число представляется в виде суммы тысяч, сотен, десятков и единиц. Далее из следующей таблицы берётся по одному элементу, соответствующему тысячам, сотням, десяткам, единицам ровно в таком порядке.

Цифра	Тысячи	Сотни	Десятки	Единицы
1	M	C	X	I
2	MM	CC	XX	II
3	MMM	CCC	XXX	III
4		CD	XL	IV
5		D	L	V
6		DC	LX	VI
7		DCC	LXX	VII
8		DCCC	LXXX	VIII
9		CM	XC	IX

Если число тысяч, сотен, десятков, единиц равно 0, то из соответствующего столбца ничего не берётся. Например, число 1990 записывается, как 1000 + 900 + 90 = MCMXC.

Пример входных и выходных данных

Ввод	Вывод
XIIV	15

Система оценивания

Решение, правильно работающее только для случаев, когда входная строка содержит только символы I, V, X, будет оцениваться в 30 баллов.

Решение, правильно работающее только для случаев, когда входная строка содержит только символы I, V, X, L, C, будет оцениваться в 60 баллов.

Решение

В записи римских чисел в большинстве случаев нет никакой разницы, как разбивать числа на слагаемые. Например, запись XV обозначает 15, а если разбить её на слагаемые X + V, то сумма также будет равна 15. Исключения из этого правила возникают, только когда появляются символы, уменьшающие значение следующего за ними символа. Таких комбинаций 6: IV, IX, XL, XC, CD, CM. Поскольку нам требуется уменьшить сумму, то такие комбинации нужно считать за один символ со значением 4, 9, 40, 90, 400, 900 соответственно.

Но такие комбинации могут склеиваться вместе, например, IXL, IXC, XCD, XCM. Такие тройки символов нужно разбивать следующим образом: IXL = I + XL, IXC = I + XC, XCD = X + CD, XCM = X + CM.

Наконец, возможны последовательности из трёх подряд идущих вычитаний: IXCD и IXMC. Их нужно разбивать так: IXCD = IX + CD, IXCM = IX + CM.

Поэтому можно идти по строке от начала до конца и смотреть на группу символов, начиная с текущей позиции:

а) если от текущей позиции начинается четвёрка IXCD или IXCM, то разбить её на комбинации IX + CD или IX + CM.

б) если от текущей позиции начинается тройка IXL, IXC, XCD, XCM, то разбить их так, как указано выше.

в) если от текущей позиции начинается пара IV, IX, XL, XC, CD, CM, то обработать её как одну цифру.

г) наконец, если ничего из вышеперечисленного не сработало, то обработать один текущий символ, как одну цифру.

Но есть и более элегантная реализация этого алгоритма, не требующая отдельного анализа троек и четвёрок символов. Для этого заметим, что в тройках и в четвёрках мы разбиваем их на пары начиная с конца. Поэтому будем идти по строке с конца. Если встречаем пару символов IV, IX, XL, XC, CD, CM, то обрабатываем их как одну цифру римской записи, иначе обрабатываем один символ, как одну цифру.

В этом решении для сокращения программы используется словарь (ассоциативный массив) digits, который сопоставляет строкам из одной или двух цифр их значение в римской записи. От текущей позиции в этом решении сначала рассматриваются два символа: предыдущий s[i - 1[и текущий s[i], если строка из этих символов содержится в словаре digits, то к ответу прибавляется значение этих цифр и i уменьшается на 2. Иначе к ответу прибавляется значение одной цифры s[i] и i уменьшается на 1.

```
s = input()
ans = 0
digits = {"I": 1, "V": 5, "X": 10, "IV": 4, "IX": 9, "L": 50, "C":
100, "XL": 40, "XC": 90, "D": 500, "M": 1000, "CD": 400, "CM":
900}
i = len(s) - 1
```

```
while i >= 0:
    num2 = s[i – 1] + s[i]
    num1 = s[i]
    if i > 0 and num2 in digits:
        ans += digits[num2]
        i -= 2
    else:
        ans += digits[num1]
        i -= 1
print(ans)
```

Задача 3. Турнир

В турнире участвуют N команд. Турнир проводится по олимпийской системе (команды играют «на вылет», проигравшие команды выбывают из турнира, выигравшие проходят в следующий тур, ничьих не бывает). Число команд в этой задаче будет степенью двойки: $N = 2^k$.

Все команды пронумерованы числами от 1 до N. В первом туре играют команды с номерами 1 и 2, 3 и 4, 5 и 6 и т. д., всего играется $N / 2$ матчей. По результатам этих матчей команды выходят во второй тур. Во втором туре играют победители первой и второй игры первого тура, победители третьей и четвёртой игры первого тура и т. д. Они выходят в третий тур. В третьем круге играют вместе победители первой и второй игры второго тура, победители третьей и четвёртой игры второго тура и т. д.

Вам даны результаты всех матчей. Определите номер команды, которая стала победителем турнира.

В первой строке входных данных записано число N — количество команд, участвовавших в турнире. Оно является степенью двойки и может принимать значения от $2^0 = 1$ до $2^{16} = 65536$. Следующая N - 1 строка содержат результаты всех сыгранных матчей. Первые $N / 2$ строк из них являются результатами матчей первого тура, затем идёт $N / 4$ строк с результатами второго тура, $N / 8$ строк с результатами третьего тура и т. д.

Результат каждого матча является одним из двух возможных чисел: 1 или 2. Число 1 означает, что в матче выиграла первая команда (номер которой меньше), число 2 означает, что в матче выиграла вторая команда (номер которой больше).

Программа должна вывести одно число — номер победившей в турнире команды.

Пример входных и выходных данных

Ввод	**Вывод**
8 1 2 2 1 2 1 1	4

Система оценивания

Решение, правильно работающее только для случаев, когда $N \leq 4$, будет оцениваться в 20 баллов.

Решение, правильно работающее только для случаев, когда $N \leq 8$, будет оцениваться в 40 баллов.

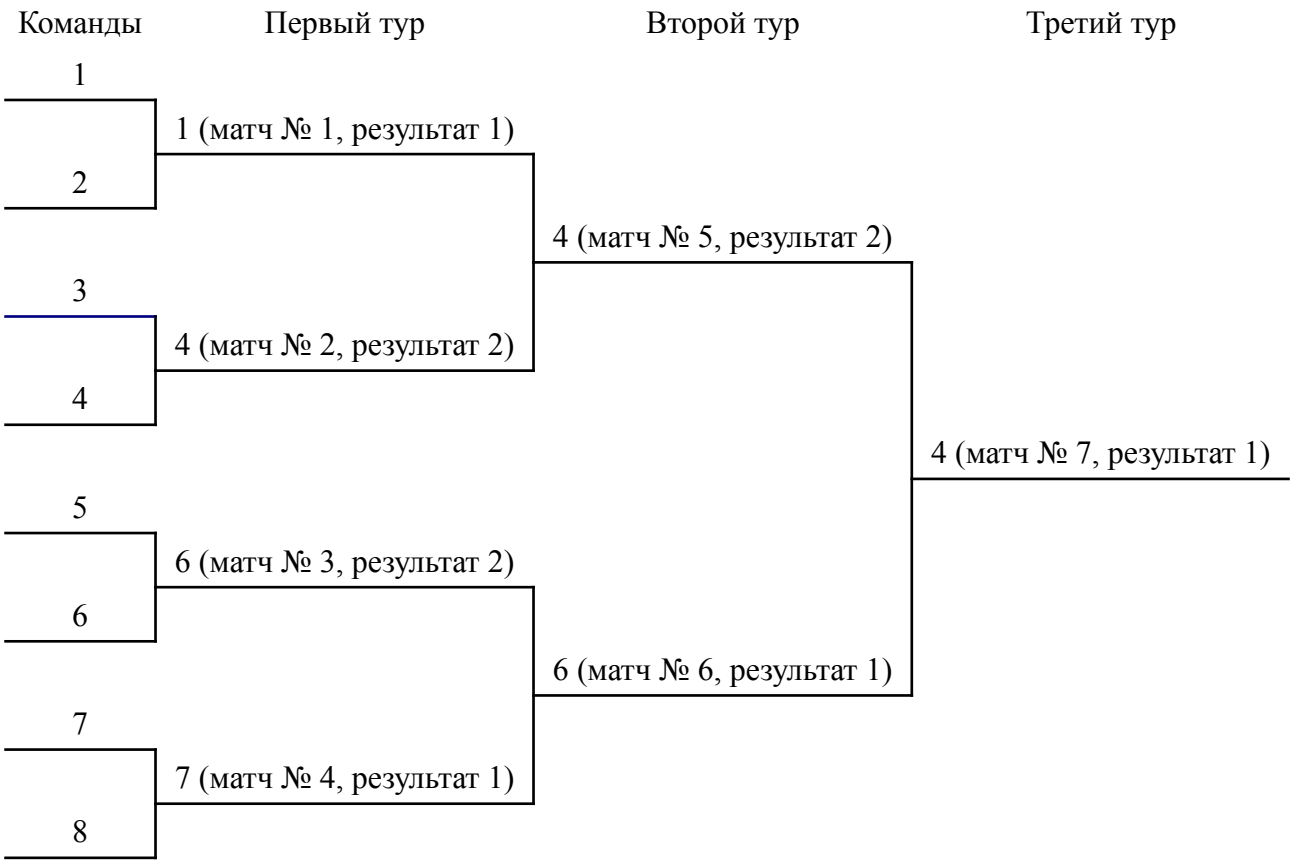
Пояснение к примеру

В следующей таблице нарисована схема турнира для примера из условия. В турнире участвовало 8 команд. Результаты матчей: 1, 2, 2, 1, 2, 1, 1.

В первом туре играли команды 1 и 2, 3 и 4, 5 и 6, 7 и 8. Результаты матчей первого тура: 1, 2, 2, 1, во второй тур вышли команды 1, 4, 6, 7.

Во втором туре играли команды 1 и 4, 6 и 7. Результаты матчей второго тура: 2, 1. В третий тур вышли команды 4 и 6.

В последнем третьем туре играют команды 4 и 6, результат матча: 1, поэтому победителем турнира является команда 4.



Решение

Эта задача является упражнением на использование массивов, в котором нужно смоделировать описанный процесс. Удобно использовать массивы переменной длины, например, в языке C++ удобно использовать контейнеры `vector`, а в языке Python — контейнер `list`.

Создадим массив из номеров команд, участвующих в турнире (teams), заполним его числами от 1 до N. Далее будем моделировать процесс турнира. В моделировании будет два цикла. Внешний цикл — по турам, этот цикл продолжается, пока в турнире не останется одна команда, то есть пока длина массива teams будет больше 1. Внутренний цикл — цикл по всем играм этого тура, это цикл с шагом 2. Длина этого цикла равна половине длины массива `teams`. Внутри этого цикла мы считываем результат игры. В игре участвуют две команды из массива teams: `teams[i]` и `teams[i + 1]`. Если результат игры равен 1, то в следующий тур выходит команда `teams[i]`, иначе — `teams[i + 1]`. Номер победившей команды добавляется в конец массива `left_teams` — это команды, вышедшие в следующий тур.

После окончания тура команды из массива left_teams переносятся в массив teams.

```
n = int(input())
teams = [i + 1 for i in range(n)]
```

```
while len(teams) > 1:
    left_teams = []
    for i in range(0, len(teams), 2):
        if int(input()) == 1:
            left_teams.append(teams[i])
        else:
            left_teams.append(teams[i + 1])
    teams = left_teams
print(teams[0])
```

Задача 4. Лифт в бизнес-центре

В прошлом году на муниципальном этапе была задача про сотрудников бизнес-центра, которые вечером выходят с работы. Теперь решите задачу про сотрудников бизнес-центра, которые утром приходят на работу.

Бизнес-центр представляет собой N-этажное здание, этажи пронумерованы от 1 до N снизу вверх. На каждом этаже работает ровно один сотрудник. Все сотрудники утром приезжают на парковку, которая расположена в подвальном помещении на один этаж ниже первого. Бизнес-центр оборудован лифтом, который вмещает неограниченное число людей, но вредный лифтёр сегодня готов отвезти всех сотрудников только на один какой-то этаж.

У каждого сотрудника есть выбор: он может пойти вверх пешком по лестнице, на подъём на один этаж при этом будет уходить A секунд. Либо он может сесть в лифт, который отвезёт всех сотрудников на какой-то выбранный ими вместе этаж. Выйдя из лифта, сотрудник может подняться до своего этажа (также тратя A секунд на подъём на один этаж), либо спуститься до нужного этажа вниз, тратя B секунд на спуск на один этаж. Лифт тратит C секунд на подъём на один этаж.

Определите минимальное время, за которое все сотрудники разойдутся по своим этажам, если они наилучшим образом выберут этаж, на который едет лифт, и свою стратегию поведения (подниматься по лестнице, или ехать на лифте, а затем идти по лестнице).

Первая строка входных данных содержит число N — количество этажей в бизнес-центре. Следующие три строки содержат числа A, B, C — время, необходимое сотруднику на подъем на один этаж, на спуск на один этаж и время, необходимое лифту на подъём на один этаж. Все числа — целые положительные, не превосходящие 2×10^9, при этом $A \geq B$, $A \geq C$.

Программа должна вывести единственное целое число — минимальное время, за которое все сотрудники могут добраться до своего этажа.

Примеры входных и выходных данных

Ввод	**Вывод**	**Примечание**
6 20 10 5	45	В здании 6 этажей. Сотрудник поднимается на один этаж за 20 секунд, спускается за 10 секунд. Лифт поднимается на один этаж за 5 секунд. Чтобы быстрее всем добраться до мест, лифт едет на 5 этаж за 25 секунд. Сотрудник, который работает на 6 этаже, выходит из лифта и поднимается за 20 секунд, всего его путь занимает 45 секунд. Сотрудник, работающий на 3 этаже, едет на лифте и спускается на 2 этажа, это также занимает 45 секунд. Сотрудники с 4 и 5 этажей также едут на лифте, их путь будет быстрее 45 секунд. На 1 и 2 этажи сотрудники поднимаются пешком по лестнице за 20 и 40 секунд соответственно. Итого все

		сотрудники добираются до своих этажей не более, чем за 45 секунд.

Система оценивания

Решение, правильно работающее только для случаев, когда все входные числа не превосходит 100, будет оцениваться в 40 баллов.

Решение, правильно работающее только для случаев, когда все входные числа не превосходит 10^5, будет оцениваться в 70 баллов.

Решение

Сначала придумает наиболее простое решение перебором. Пусть лифт едет на этаж номер x, а те, кто не едут в лифте, поднимаются пешком на этаж номер y. Тогда затраченное время будет равно.

а) Для тех, кто не едет в лифте — A * y.

б) Для тех, кто едет в лифте, затем спускается — C * x + B * (x - y - 1).

в) Для тех, кто едет в в лифте, затем поднимается — C * x + B * (x - y - 1).

Из этих значений нужно выбрать наибольшее.

Затем двумя вложенными циклами переберём значения x и y ($0 \le y < x \le N$) и выберем из них такую комбинацию, при которой затраченное время будем минимальным.

Пример решения на языке Python.

```
N = int(input())
A = int(input())
B = int(input())
C = int(input())

def time(x, y):
    return max(x * C + (N-x) * A, x * C + (x-y-1) * B, y * A)

ans = N * A
for x in range(1, N + 1):
    for y in range(0, x):
        ans = min(ans, time(x, y))
print(ans)
```

Такое решение имеет сложность $O(N^2)$ и набирает 40 баллов.

Улучшим это решение, избавимся от цикла по переменной y, найдём значение y без перебора. Для этого представим себе группу людей, которые начинают подниматься снизу вверх, а также людей, которые доезжают до этажа x, затем спускаются. Нам нужно минимизировать время, за которое на каждом этаже от 1 до x окажется по человеку. Это то время, когда две группы людей (поднимающихся и спускающихся) встретятся. Составим уравнение, где эти люди встретятся, пусть они встретились на этаже y. Тогда поднимающиеся вверх потратили время A * y, а спускающиеся вниз (с учётом времени на поездку на лифте) — C * x + A * (x - y). Приравняем эти времена и решим уравнение: A * y = C * x + B * (x - y), откуда y = (C + B) * x / (A + B).

Возьмём в качестве y целую часть от этого деления, так как на самом деле поднимающиеся и спускающиеся должны закончить свой маршрут не на одном этаже, а на двух соседних этажах. То есть мы просто заменим нахождение значения y при помощи цикла

на вычисление по формуле. Пример такого решения (оно набирает 70 баллов).

```
N = int(input())
A = int(input())
B = int(input())
C = int(input())

def time(x, y):
    return max(x * C + (N-x) * A, x * C + (x-y-1) * B, y * A)

ans = N * A
for x in range(1, N + 1):
    y = (C + B) * x // (A + B)
    ans = min(ans, time(x, y))
print(ans)
```

Улучшим и это решение, избавившись от цикла и по переменной x. Для этого заметим, что в оптимальном решении все три группы людей (поднимающиеся снизу, едущие на лифте и спускающиеся вниз, едущие на лифте и поднимающиеся вверх) должны достичь конечного этажа одновременно (совсем одновременно не получится, так как задача целочисленная, а не непрерывная). Приравняем эти три времени:

A * y = C * x + B * (x - y - 1) = C * x + B * (x - y - 1)

Мы получим систему из 2 уравнений с 2 неизвестными, которую можно решить. Можно также вместо одного уравнения использовать ранее найденную подстановку y = (C + B) * x / (A + B). Получим решение x = (N * A * A + N * A * B) / (2 * A * B + A * A - C * B). Но поскольку необходимо найти решение, минимизирующее функцию только для целочисленных значений x и y, мы возьмём в качестве значения x два ближайших целых числа, то есть результат деления, округлённый вниз и вверх. Значение y по выбранному значению x определяется однозначно, как в предыдущем решении.

Пример решения, набирающего 100 баллов.

```
N = int(input())
A = int(input())
B = int(input())
C = int(input())

def time(x, y):
    return max(x * C + (N-x) * A, x * C + (x-y-1) * B, y * A)

ans = N * A
X = (N * A * A + N * A * B) // (2 * A * B + A * A - C * B)
for x in (X, X + 1):
    y = (C + B) * x // (A + B)
    ans = min(ans, time(x, y))
print(ans)
```

Также на 100 баллов можно было реализовать решение, где для поиска значения x и y используется не формула, а двоичный поиск.

Задача 5. Дом у озера

Есть озеро, рядом с которым хотят построить дом. Архитектурный проект (форма дома) уже утверждён, можно только выбрать расположение дома так, чтобы он оказался рядом с озером. Желательно выбрать расположение дома рядом с озером так, чтобы у как можно большего числа жителей дома окна выходили на озеро, то есть чтобы длина общей границы дома и озера была максимальной.

План дома и озера задан в виде изображения на клетчатой бумаге, в котором отмечены клетки, принадлежащие дому и озеру. Первая строка входных данных содержит число *N* — количество строк в плане дома и озера. Вторая строка входных данных содержит число *M* — количество столбцов в плане дома и озера. Следующие *N* строк содержат по *M* символов — план дома. Символ «.» в этих строках обозначает пустую клетку, символ «H» обозначает клетку дома. План дома является связной областью и не содержит «дырок» внутри. В плане есть хотя бы одна клетка, принадлежащая дому.

Следующие *N* строк по *M* символов в каждой содержат план озера, в этих строках символ «.» обозначает пустую клетку, символ «W» обозначает клетку, занятую озером. План озера является связной областью и не содержит «дырок» внутри. В плане есть хотя бы одна клетка, принадлежащая озеру.

Программа должна вывести одно число — максимальное значение длины общей границы дома и озера, измеряемую в сторонах клетки. При расположении дома рядом с озером его можно передвигать, но нельзя делать повороты и отражения. Вокруг озера есть неограниченное свободное пространство, дом может располагаться с любой стороны от озера.

Числа *N* и *M* являются целыми, положительными, не превосходят 20.

Примеры входных и выходных данных

Ввод	**Вывод**	**Пример наилучшего размещения**
5 6 ..... .HHHH. .HHHH. .HHHH. ....WWW. ..WWW. ..WWW. ..WWW. ..WWW.	3	. . . . W W W H H H H W W W H H H H W W W H H H H W W W . . . . W W W
5 7 ...HHH. .H...H. .H.HHH. .H.H... .HHH... ....WWWWW WWW...W	11	. W W W W W W W W H H H W W H W . . H W . H W H H H . . H . H . . . . H H H . . .

W.W...W ..W....		

Система оценивания

Решение, правильно работающее только для случаев, когда дом и озеро являются прямоугольниками, будет оцениваться в 40 баллов.

Решение будет оцениваться, только если оно проходит первый тест из условия задачи (прохождение второго теста обязательным не является).

Решение

Переберём все возможные расположения дома относительно озера. Пусть карта озера смещена относительно карты дома на вектор (dx, dy). Будем перебирать все такие смещения, причем для значения dx возможные значения смещения изменяются от -N до N, а для значения dy от -M до M.

Затем для каждого возможного вектора смещения совместим карты дома и озера. Проверим, что ни одна клетка дома не наложилась на клетку озера, а также посчитаем для каждой клетки дома количество соседних с ней клеток озера, и выберем максимальное значение среди всех возможных смещений.

Пример решения на языке Python.

```
N = int(input())
M = int(input())
house = [input() for i in range(N)]
water = [input() for i in range(N)]
ans = 0
for dx in range(-N, N + 1):
  for dy in range(-M, M + 1):
    count = 0
    for house_x in range(N):
      for house_y in range(M):
        if house[house_x][house_y] == "H":
          water_x = house_x + dx
          water_y = house_y + dy
          if 0<=water_x<N and 0<=water_y<M and water[water_x][water_y]=="W":
            count = -10 ** 9
            for neigh_x, neigh_y in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
            water_x = house_x + dx + neigh_x
            water_y = house_y + dy + neigh_y
            if (0 <= water_x < N and 0 <= water_y < M and
                              water[water_x][water_y] == "W"):
              count += 1
    ans = max(ans, count)
print(ans)
```