Cheating a Boolean Tree. Логическое дерево

Имя входного файла: `boolean.in`
Имя выходного файла: `boolean.out`

Рассмотрим разновидность двоичного дерева которую мы назовем логическим деревом. В этом дереве каждый уровень полностью заполнен, за исключением, возможно, последнего (самого глубокого) уровня. При этом, все вершины последнего уровня находятся максимально слева. Дополнительно, каждая вершина дерева имеет ноль или двоих детей.

Каждая вершина дерева имеет связанное с ней логическое значение (1 или 0). Кроме этого, каждая *внутренняя* вершина имеет связанную с ней логическую операцию ("И" или "ИЛИ"). Значение вершины с операцией "И" это логическое "И" значений ее детей. Аналогично, значение вершины с операцией "ИЛИ" это логическое "ИЛИ" значений ее детей. Значения всех листьев задаются во входном файле, поэтому значения всех вершин дерева могут быть найдены.

Наибольший интерес для нас представляет корень дерева. Мы хотим, чтобы он имел заданное логическое значение v, которое может отличаться от текущего. К счастью, мы можем изменять логические операции некоторых внутренних вершин (заменить "И" на "ИЛИ" и наоборот).

Дано описание логического дерева и набор вершин, операции в которых могут быть изменены. Найдите наименьшее количество вершин, которые следует изменить, чтобы корень дерева принял заданное значение v. Если это невозможно, то выведите строку "`IMPOSSIBLE`" (без кавычек).

Формат входного файла

В первой строке входного файла находятся два числа n и v ($1 \le n \le 10\,000, 0 \le v \le 1$) — количество вершин в дереве и требуемое значение в корне соответственно. Поскольку все вершины имеют ноль или двоих детей, то n – нечетное. Следующие n строк описывают вершины дерева.

Первые $(n-1)/2$ строк описывают внутренние вершины. Каждая из них содержит два числа — g и c, которые принимают значение либо 0, либо 1. Если $g = 1$, то вершина представляет логическую операцию "И", иначе она представляет логическую операцию "ИЛИ". Если $c = 1$, то операция в вершине может быть изменена, иначе нет. Внутренняя вершина с номером i имеет детей $2i$ и $2i+1$.

Следующие $(n+1)/2$ строк описывают листья. Каждая строка содержит одно число 0 или 1 — значение листа.

Формат выходного файла

В выходной файл выведите одно число — ответ на задачу.

Пример

boolean.in	boolean.out
9 1 1 0 1 1 1 1 0 0 1 0 1 0 1	1
5 0 1 1 0 0 1 1 0	IMPOSSIBLE

Brackets. Скобки

Имя входного файла: `brackets.in`
Имя выходного файла: `brackets.out`

Задан шаблон, состоящий из круглых скобок и знаков вопроса. Требуется определить, сколькими способами можно заменить знаки вопроса круглыми скобками так, чтобы получилось правильная скобочная последовательность.

Формат входного файла

Первая строка входного файла содержит заданный шаблон, длиной не более 2 000 символов.

Формат выходного файла

Выведите искомое количество способов по модулю 301 907.

Пример

brackets.in	brackets.out
????(?	2

Numbers. Числа

Имя входного файла: `numbers.in`
Имя выходного файла: `numbers.out`

Дана последовательность чисел a_1, a_2, ..., a_n. За одну операцию разрешается удалить любое (кроме крайних) число, заплатив за это штраф, равный произведению этого числа на сумму соседних. Требуется удалить все числа, кроме крайних, с минимальным суммарным штрафом.

Пример начальной последовательности:
1 50 51 50 1
удаляем четвертое число, штраф $50 \cdot (1 + 51) = 2600$, получаем
1 50 51 1
удаляем третье число, штраф $51 \cdot (50 + 1) = 2601$, получаем
1 50 1
удаляем второе число, штраф $50 \cdot (1 + 1) = 100$.
Итого, штраф 5301.

Формат входного файла

В первой строке входного файла расположено одно число n ($1 \le n \le 100$) — количество чисел в последовательности.

Во второй строке находятся n целых чисел a_1, a_2, ... a_n; никакое из чисел не превосходит по модулю 100.

Формат выходного файла

Выведите в выходной файл одно число — минимальный суммарный штраф.

Пример

numbers.in	numbers.out
5 1 50 51 50 1	5301