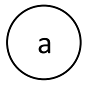
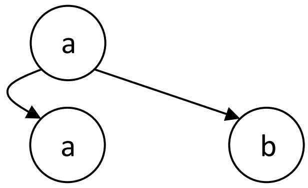
Алгоритм

Обозначим 2 цвета буквами A и B.

Рассмотрим ситуацию, когда первый стул выкрашен цветом A:

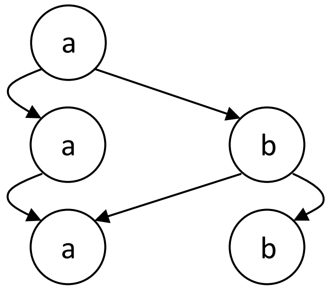


Так как 1 стул – нечетное кол-во, то следующий стул может быть либо такого же цвета, либо другого:

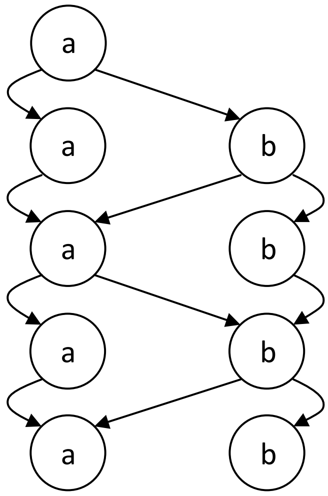


Здесь и далее на графах **i**-я трока соответствует **i**-му стулу.

Заметим, что если второй стул цвета A, то получается нечетное кол-во стульев цвета A, а значит третий стул будет такого же цвета. Если же второй стул имел цвет B, то выходит ситуация, аналогичная ситуации с 1м стулом, тогда получаем следующее:



Аналогично для 5ти первых стульев:



Пусть стулья, после которых можно красить стул в другой цвет, называются «нечетными». А остальные стулья – «четными».

Для краткости графы будем изображать в виде таблицы, где «точками» будут отмечены «нечетные» стулья. Тогда соответствующая таблица для последнего графа будет выглядеть так:

a	●		
a		●	b
a	●		b
a		●	b
a	●		b

Однако стоит заметить, что если будет N=5, то последний стул не может иметь цвет B, т.к. в таком случае кол-во стульев подряд цвета B будет четным, что противоречит условию. Следовательно, **последний стул должен быть «нечетным».**

Следуя этом правилу, таблицы для N=3 и N=4 будут выглядеть так:

№	**цвет1**			**цвет2**
1	a	●		
2	a		●	b
3	a	●		

№	**цвет1**			**цвет2**
1	a	●		
2	a		●	b
3	a	●		b
4			●	b

Итак, мы получили граф раскрашивания стульев. Т.е. путь, по которому мы пройдем от первого верхнего стула до нижнего однозначно определит способ раскрашивания этой последовательности стульев так, чтобы это удовлетворяло условию.

Каждый новый путь соответствует новой раскраске, следовательно, кол-во путей равно кол-во способов раскрасить эти N стульев в соответствии с условием. Осталось лишь посчитать кол-во путей в графе.

Пусть «ai» - кол-во путей из 1го стула к **i**-му стулу, раскрашенному в цвет A. Тогда «bi» - кол-во путей из 1го стула к **i**-му стулу, раскрашенному в цвет B.

Заметим, что a1 = 1 по предположению.

b2 = a1, так как в b1 ведет только путь из a1.

a2 = a1 по той же причине.

a3 = a2+b2, так как в него можно попасть из a2 и из b2.

и так далее…

В общем виде алгоритм подсчета выглядит так:

Если i-й стул цвета A «четный», то: ai=a(i-1)+b(i-1) и bi=b(i-1)

В противном случае: bi=b(i-1)+a(i-1) и ai=a(i-1)

Так как вес (кол-во путей) «четного» стула равен весу предыдущего стула того же цвета, то получаем следующее:

$x_i = x(i-1) + x(i-2)$, где x_j – вес j-го стула такого цвета, чтобы он был «нечетным».

Так как известно, что последний стул должен быть «нечетным», то следует просто последовательно вычислять все x_j для $j=1..N$. И тогда x_N будет искомым значением. Стоит заметить, что $x_1=a_1=1$, $x_2=b_2=1$.

Теперь следует вспомнить, что по предположению первый цвет мы принимаем за A. Если же первый цвет будет B, то получим аналогичную ситуацию с x_N вариантами раскраски, причем множество этих раскрасок не пересекается с множеством, где первый цвет был A.

Тогда искомый ответ будет $2*x_N$.

Оценка

Так как при подсчете для вычисления очередного **xi** нам необходимо помнить только 2 предыдущих значения, а не всю последовательность, то сложность по памяти будет O(1).

Очевидно, что сложность по времени будет O(n).

Диапазона значений типа данных long не хватает уже после N=80, следовательно для получения точного ответа для больших N удобно использовать BigInteger из пакета java.math.