Sortowanie przez xorowanie **(sor-xor)**

Memory limit: 128 MB Time limit: 1.00 s

Jasio poznał ostatnio wiele różnych ciekawych metod na sortowanie ciągu liczb takich jak sortowanie przez scalanie, sortowanie bąbelkowe albo sortowanie przez zliczanie.

Zainspirowany tą tematyką postanowił wymyślić swój własny algorytm sortujący, który ma jednak parę wad i ograniczeń.

Ciągi, których sortowanie rozważa Jasio składają z liczb całkowitych o wartościach A_i z przedziału $[0, 2^K)$ zapisanych na dokładnie k bitach. Algorytm Jasia pozwala na wykonywanie dowolnie wiele razy następującej operacji:

- Wybierz niepusty spójny fragment ciągu od l-tej do r-tej pozycji włącznie i każdej liczbie A_i ($l \leq i \leq r$) zamień wszystkie bity na przeciwne (zamień A_i na $A_i \oplus (2^K - 1)$).

Twoim zadaniem jest napisać program, który obliczy jaka jest minimalna liczba operacji konieczna do posortowania ciągu niemalejąco, albo wypisze, że nie jest to możliwe.

Wejście

W pierwszym wierszu wejścia znajdują się dwie liczby całkowite N i K oznaczające długość ciągu oraz liczbę bitów, za pomocą których da się zapisać wszystkie elementy ciągu.

W drugim wierszu wejścia znajduje się N liczb całkowitych pooddzielanych pojedynczymi odstępami oznaczających kolejne elementy ciągu A_i.

Wyjście

W pierwszym (jedynym) wierszu wyjścia powinna się znaleźć się minimalna liczba operacji potrzebna do tego, żeby ciąg stał się niemalejący, albo jedna liczba `-1` jeżeli posortowanie nie jest możliwe.

Ograniczenia

$1 \leq N \leq 500\,000$, $1 \leq K \leq 20$.

Przykład

Input	Output	Explanation
`4 3` `5 2 4 1`	`2`	Możemy wybrać przedziały pozycji $[1, 3]$ oraz $[2, 4]$ otrzymując na końcu ciąg $(2, 2, 4, 6)$.

Input	Output
`7 2` `0 1 0 1 0 1 0`	`-1`