D. Lucky Segments

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Petya loves lucky numbers. Everybody knows that lucky numbers are positive integers whose decimal representation contains only the lucky digits **4** and **7**. For example, numbers **47**, **744**, **4** are lucky and **5**, **17**, **467** are not.

Petya has n number segments $[l_1; r_1]$, $[l_2; r_2]$, ..., $[l_n; r_n]$. During one move Petya can take any segment (let it be segment number i) and replace it with segment $[l_i + 1; r_i + 1]$ or $[l_i - 1; r_i - 1]$. In other words, during one move Petya can shift any segment to the left or to the right by a unit distance. Petya calls a number full if it belongs to each segment. That is, number x is full if for any i $(1 \le i \le n)$ the condition $l_i \le x \le r_i$ is fulfilled.

Petya makes no more than k moves. After that he counts the quantity of full lucky numbers. Find the maximal quantity that he can get.

Input

The first line contains two integers n and k $(1 \le n \le 10^5, 1 \le k \le 10^{18})$ — the number of segments and the maximum number of moves. Next n lines contain pairs of integers l_i and r_i $(1 \le l_i \le r_i \le 10^{18})$.

Please do not use the `%lld` specificator to read or write 64-bit integers in C++. It is preferred to use the `%I64d` specificator.

Output

Print on the single line the single number — the answer to the problem.

Examples

input
4 7 1 4 6 9 4 7 3 5

output
1

input
2 7 40 45 47 74

output
2

Note

In the first sample Petya shifts the second segment by two units to the left (it turns into $[4; 7]$), after that number 4 becomes full.

In the second sample Petya shifts the first segment by two units to the right (it turns into $[42; 47]$), and shifts the second segment by three units to the left (it turns into $[44; 71]$), after that numbers 44 and 47 become full.