Reverse LIS

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1.5 seconds
Memory limit:	512 megabytes

Nikuniku loves long, non-decreasing subsequences. She has a 0-1 string s and wants to make the longest non-decreasing subsequences of s as long as possible.

0-1 string consists of only two kinds of characters, '`0`' and '`1`', in which '`0`' is considered to be smaller than '`1`'. A subsequence of a string is a new string formed from the original string by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (i.e., "`010`" is a subsequence of "`0110`" while "`001`" is not).

Nikuniku decided to make some changes to the string. Each change reverses an arbitrary substring of s (i.e. make $s[l, r] = s_r s_{r-1} \dots s_l$). She would like to know how long is the longest possible non-decreasing subsequence after at most k changes (or revlis(s, k) in short), for q different k-s. Note that the q queries are independent.

As the string s could be very long, Nikuniku decided to present it in run-length encoded form. A run-length encoded string contains n parts, each part has p_i copies of the same character c_i. For example, string "`001111`" has two parts, with p_1 being 2, c_1 being '`0`', p_2 being 4, and c_2 being '`1`'.

Input

The first line contains an integer n ($1 \le n \le 2 \cdot 10^5$), the number of parts in the string s.

The following n lines describe the parts. Each line contains a character c_i ($c_i \in \{$'`0`', '`1`'$\}$) and an integer p_i ($1 \le p_i \le 10^9$), separated by a space. It is guaranteed that $c_i \neq c_{i+1}$ for $1 \le i \le n - 1$.

The next line contains an integer q ($1 \le q \le 2 \cdot 10^5$), denoting the number of queries. For the next q lines, each line contains an integer k_i ($0 \le k_i \le n$).

Output

For each query, output one line with an integer – the value of revlis(s, k_i).

Examples

standard input	standard output
4 0 1 1 2 0 3 1 4 2 0 1	8 10
5 1 2 0 5 1 2 0 5 1 2 3 5 0 5	16 12 16
7 0 1 1 3 0 7 1 6 0 6 1 6 0 6 2 1 5	26 35