E. Huffman Coding on Segment

Alice wants to send an important message to Bob. Message $a = (a_1, ..., a_n)$ is a sequence of positive integers (*characters*).

To compress the message Alice wants to use binary Huffman coding. We recall that *binary Huffman code*, or *binary prefix code* is a function f, that maps each letter that appears in the string to some binary string (that is, string consisting of characters '0' and '1' only) such that for each pair of different characters a_i and a_j string $f(a_i)$ is not a prefix of $f(a_j)$ (and vice versa). The result of the encoding of the message $a_1, a_2, ..., a_n$ is the concatenation of the encoding of each character, that is the string $f(a_1)f(a_2)...f(a_n)$. Huffman codes are very useful, as the compressed message can be easily and uniquely decompressed, if the function f is given. Code is usually chosen in order to minimize the total length of the compressed message, i.e. the length of the string $f(a_1)f(a_2)...f(a_n)$.

Because of security issues Alice doesn't want to send the whole message. Instead, she picks some substrings of the message and wants to send them separately. For each of the given substrings $a_{l_i}... a_{r_i}$ she wants to know the minimum possible length of the Huffman coding. Help her solve this problem.

Input

The first line of the input contains the single integer n ($1 \le n \le 100\,000$) — the length of the initial message. The second line contains n integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 100\,000$) — characters of the message.

Next line contains the single integer q ($1 \le q \le 100\,000$) — the number of queries.

Then follow q lines with queries descriptions. The i-th of these lines contains two integers l_i and r_i ($1 \le l_i \le r_i \le n$) — the position of the left and right ends of the i-th substring respectively. Positions are numbered from 1. Substrings may overlap in any way. The same substring may appear in the input more than once.

Output

Print q lines. Each line should contain a single integer — the minimum possible length of the Huffman encoding of the substring $a_{l_i}... a_{r_i}$.

Example

input
```
7
1 2 1 3 1 2 1
5
1 7
1 3
3 5
2 4
4 4
```

output
```
10
3
3
5
0
```

Note

In the first query, one of the optimal ways to encode the substring is to map 1 to "0", 2 to "10" and 3 to "11".

Note that it is correct to map the letter to the empty substring (as in the fifth query from the sample).