Helpful Bits

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 mebibytes

Imagine that you are asked to find the length of the shortest path between several pairs of vertices in a weighted undirected graph.

Sounds too easy, doesn't it? We think so, too.

That's why in this problem, your program will be run twice on each test. During the first run, you can read the weights of the edges. During the second run, you will be only given the graph without weights of its edges, and you will be asked to find the weight of the shortest path between several pairs of vertices.

"How would I know the shortest path without weights?" you might ask. That's a perfectly reasonable question, so we'll make it easier for you:

- If you are given m weights during the first run, you can output at most $12m$ bits as a binary string, and they will be passed to the second run of your program as is.
- The shortest path lengths don't have to be precise, but can differ from the actual value X by at most 0.1% of it. In other words, your answer Y has to satisfy the condition $0.999X \le Y \le 1.001X$.

Input

Your program will be run **twice** on each test.

The first line contains one integer T ($1 \le |T| \le 10^5$). Its absolute value denotes the number of test cases. If $T < 0$, it's the first run; otherwise, it's the second run. Then, $|T|$ test cases follow in the following format.

If this is the first run:

The next line contains one integer m ($1 \le m \le 10^5$), denoting the number of edges in the graph.

The next line contains m integers $w_1, w_2, \ldots, w_m$ ($1 \le w_i \le 10^6$), denoting the weights of the edges.

If this is the second run:

The next line contains four integers n, m, q, b ($2 \le n \le 10^5$, $1 \le m \le 10^5$, $1 \le q \le 10^5$, $0 \le b \le 12m$), denoting the number of vertices, the number of edges in the graph, the number of shortest path length queries, and the number of bits you sent during the first run, respectively.

The next line contains a binary string of length b: the bits you printed during the first run.

The i-th of the next m lines contains two integers a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$): the endpoints of the next edge. The weight of this edge is equal to w_i from the first run (but is not given during the second run).

Each of the next q lines contains two integers s, f ($1 \le s, f \le n$), which means that you are asked to find the length of the shortest path from vertex s to vertex f.

It is guaranteed that the value of m given during both runs is the same, that no pair of vertices is directly connected more than once, and that both the sum of n and the sum of $m \cdot q$ over all test cases in a single file do not exceed $3 \cdot 10^5$.

Output

For each test case:

If this is the first run:

Print one integer b ($0 \le b \le 12m$): the number of bits you intend to pass to the second run.

If $b > 0$, print a single binary string of length b in the next line: the bits themselves. They will be passed intact to the second run.

If this is the second run:

For each of the q queries, print the length of the shortest path between two requested vertices, or -1 if such a path does not exist. Your answer doesn't have to be an integer and can be a fractional number with at most 15 digits after the decimal separator. Your answer will be accepted if its **absolute** difference from the correct answer X is at most $X \cdot 10^{-3}$.

Example

standard input	standard output
-2 3 4 5 6 1 1 - - - - - - - - - - - - - - - - - - - 2 4 3 5 36 000000000100000000000101000000000110 1 2 2 3 3 1 1 1 1 2 3 1 2 3 2 4 2 1 1 12 000000000001 1 2 1 2	36 000000000100000000000101000000000110 12 000000000001 - - - - - - - - - - - - - - - - - - - 0.00000 4.00000 6.00000 5.00000 -1.00000 1.00000

Note

In the example, the input and output data are listed for both runs of the program, separated with a dashed line and additional empty lines. Neither dashed lines nor empty lines are present in the actual input and output data.

For both test cases of the example, during the first run, the solution decided to use all available bits and write all the weights as 12-digit binary numbers. Then, during the second run, the solution worked out what the weights of all the edges were and printed all the requested shortest path lengths, including the absence of a path between vertices 2 and 4.

Note that when you view examples in the testing system, input data might not correspond to the exact input data for any of the runs, as those are generated from what is shown in the internal format of tests.