Tax

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

JB received his driver's license recently. To celebrate this fact, JB decides to drive to other cities in Byteland. There are n cities and m bidirectional roads in Byteland, labeled by $1, 2, \ldots, n$. JB is at the 1-st city, and he can only drive on these m roads. It is always possible for JB to reach every city in Byteland.

The length of each road is the same, but they are controlled by different engineering companies. For the i-th edge, it is controlled by the c_i-th company. If it is the k-th time JB drives on an edge controlled by the t-th company, JB needs to pay $k \times w_t$ dollars for tax.

JB is selecting his destination city. Assume the destination is the k-th city, he will drive from city 1 to city k along the shortest path, and minimize the total tax when there are multiple shortest paths. Please write a program to help JB calculate the minimum number of dollars he needs to pay for each possible destination.

Input

The input contains only a single case.

The first line of the input contains two integers n and m ($2 \le n \le 50$, $n-1 \le m \le \frac{n(n-1)}{2}$), denoting the number of cities and the number of bidirectional roads.

The second line contains m integers $w_1, w_2, \ldots, w_m$ ($1 \le w_i \le 10\,000$), denoting the base tax of each company.

In the next m lines, the i-th line ($1 \le i \le m$) contains three integers u_i, v_i and c_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$, $1 \le c_i \le m$), denoting denoting an bidirectional road between the u_i-th city and the v_i-th city, controlled by the c_i-th company.

It is guaranteed that there are at most one road between a pair of city, and it is always possible for JB to drive to every other city.

Output

Print $n-1$ lines, the k-th ($1 \le k \le n-1$) of which containing an integer, denoting the minimum number of dollars JB needs to pay when the destination is the $(k+1)$-th city.

Example

standard input	standard output
5 6 1 8 2 1 3 9 1 2 1 2 3 2 1 4 1 3 4 6 3 5 4 4 5 1	1 9 1 3