动态的果子合并

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

在学习算法的过程中，小博领悟了最经典的贪心问题：果子合并问题。

这个问题给出了 n 个果子，其中第 i 个果子具有 a_i 的重量；每次果子合并操作可以任选两个果子合并，假设这两个果子重量分别为 a_i 和 a_j ，那么此次合并消耗的代价是两个果子的重量之和 $a_i + a_j$，合并之后这两个果子消失，取而代之的是一个重量为 $a_i + a_j$ 的果子。 这个问题的目标是求解出把 n 个果子进行 $n-1$ 次合并得到一个果子所消耗的最小总代价。

小博认为这个问题太简单了，于是开始冥思苦想这个问题的动态版本：最开始给出果子重量的多重集 $A = \{a_1, a_2 \cdots a_n\}$ ，然后进行 m 次操作，每次操作是以下两种之一：

1. 添加一个重量为 x 的果子到多重集合 A 中。
2. 从多重集合 A 删除一个重量为 x 的果子（保证 A 中存在重量为 x 的果子）。

每次操作结束后，小博希望能快速求解当前多重集合 A 的果子合并问题。

遗憾的是，他思考良久也没想到优秀的算法。这时，他的朋友小思来帮助他了。小思给出了一个"54二进制好果子"集合 $B = \{2, 4, 8, 16, 32 \ldots 2^{54}\}$ ，形式化的说，B 集合里元素为 $2^k, (1 \le k \le 54)$ ，共 54 个整数。

小思表示，只要固定加上 B 集合里的这些好果子，小博的动态问题就可以解决了，并把这个难题留给了小博。

那么现在请你帮助小博解决下面的问题:

给出一个大小为 n 的多重集合 A，表示 n 个果子的重量；给出 m 个操作，操作形式如上文所述；在每次操作之后，请求出多重集合 $A \cup B$ 的果子合并问题的答案（即，同时考虑当前多重集合 A 中的果子和集合 B 中的果子，然后求解把他们全部合并成一个果子的最小总代价）。

Input

第一行包含两个正整数 $n, m (1 \le n \le 10^5,\ 1 \le m \le 2 \times 10^5)$ ，分别表示集合 A 的初始大小，和接下来进行的操作次数。

第二行包含 n 个正整数 $a_1, a_2 \cdots a_n (1 \le a_i \le 10^9)$，分别表示 A 集合中初始果子的重量。

接下来 m 行，每行首先输入两个整数 $op, x (op \in \{1, 2\},\ 1 \le x \le 10^9)$。

如果 $op = 1$ ，则表示向多重集合 A 中插入一个重量为 x 的果子。

如果 $op = 2$ ，则表示向多重集合 A 中删除一个重量为 x 的果子（保证被删除对象存在）。

注意到，B 集合是一个题目中给定的固定的集合，因此不需要输入。

Output

输出共 m 行。每次操作后输出一行，包含一个整数，表示当前多重集合 $A \cup B$ 的果子合并问题的最小总代价。

Examples

standard input	standard output
1 2 1 2 1 1 2	72057594037927822 72057594037927932
6 10 1 1 4 5 1 4 1 9 1 9 2 1 2 1 2 4 2 5 2 1 2 4 2 9 2 9	72057594037929188 72057594037929672 72057594037929615 72057594037929558 72057594037929338 72057594037929065 72057594037929008 72057594037928788 72057594037928304 72057594037927822