

SEGSQRSS - Sum of Squares with Segment Tree

#tree

Segment trees are extremely useful. In particular "Lazy Propagation" (i.e. [see here, for example](#)) allows one to compute sums over a range in $O(\lg(n))$, and update ranges in $O(\lg(n))$ as well. In this problem you will compute something much harder:

The sum of squares over a range with range updates of 2 types:

1) increment in a range

2) set all numbers the same in a range.

Input

There will be T ($T \leq 25$) test cases in the input file. First line of the input contains two positive integers, N ($N \leq 100,000$) and Q ($Q \leq 100,000$). The next line contains N integers, each at most 1000. Each of the next Q lines starts with a number, which indicates the type of operation:

2 $st\ nd$ -- return the sum of the squares of the numbers with indices in $[st, nd]$ {i.e., from st to nd inclusive} ($1 \leq st \leq nd \leq N$).

1 $st\ nd\ x$ -- add " x " to all numbers with indices in $[st, nd]$ ($1 \leq st \leq nd \leq N$, and $-1,000 \leq x \leq 1,000$).

0 $st\ nd\ x$ -- set all numbers with indices in $[st, nd]$ to " x " ($1 \leq st \leq nd \leq N$, and $-1,000 \leq x \leq 1,000$).

Output

For each test case output the "Case <caseno>:" in the first line and from the second line output the sum of squares for each operation of type 2. Intermediate overflow will not occur with proper use of 64-bit signed integer.

Example

Input:

```
2
4 5
1 2 3 4
2 1 4
0 3 4 1
2 1 4
1 3 4 1
2 1 4
1 1
1
2 1 1
```

Output:

```
Case 1:
30
7
13
Case 2:
1
```