Problem 4 - Cats! (Programming) (13 points)

Problem Description

BB loves cats! There are N cats in his garden, and the cats are labeled from 1 to N. BB's garden can be seen as a straight line, and the i^{th} cat is initially located at x-coordinate x_i.

Every once in a while, BB will place some food somewhere in the garden, which attracts some of the cats nearby. More precisely, there are Q sequential events, the j^{th} of which is of one of the two types:

- BB placed some food at x-coordinate p_j, and attracted all cats within radius r_j. That is, every cat within $[p_j - r_j, p_j + r_j]$ moved to p_j.
- The ${t_j}^{\text{th}}$ cat moved to x-coordinate x'_j

Unfortunately, BB feels like his garden might be a bit too crowded after some of the events. The cats might feel uncomfortable due to this. To try resolving this problem, BB would like to first know the *crowdedness* of his garden. **He define *crowdedness* be the number of pairs (i, j), $i < j$, such that the i^{th} cat and the j^{th} cat are at the same location.**

Please help BB find out the *crowdedness* after each event.

Input

The first line of the input contains two positive integers N and Q, denoting the number of cats on BB's garden and the number of events, respectively.

The second line contains N space-separated integers $x_1, x_2, \ldots, x_N$, denoting the initial x-coordinate of the cats.

Then Q lines follows. The j-th line of which is of one of the following two forms:

- `1` p_j r_j: BB placed some food at p_j with radius r_j.
- `2` t_j x'_j: The ${t_j}^{\text{th}}$ cat moved to x'_j.

Note that the events happen one by one (i.e., the j^{th} event happens under the effect of all previous $j - 1$ events).

- $1 \leq N, Q \leq 10^5$
- $0 \leq x_i, p_j, r_j, x'_j \leq 10^9$
- $1 \leq t_j \leq N$

Output

Please output Q lines, where the j^{th} line denotes the *crowdedness* of BB's garden after the j^{th} event.

Test Group 0 (0 %)

- Sample Input

Test Group 1 (10 %)

- $N, Q \leq 100$

Test Group 2 (20 %)

- $N, Q \leq 5000$

Test Group 3 (30 %)

- There are only events of type 1.

Test Group 4 (40 %)

- No Additional Constraint

Sample Input 1

```
5 6
3 1 4 1 5
2 3 6
1 2 1
2 2 7
1 3 2
2 4 7
1 5 2
```

Sample Output 1

```
1
3
1
3
2
10
```

Sample Input 2

```
8 5
1 1 2 3 5 8 13 21
1 10 4
1 3 1
1 3 2
1 8 4
1 12 9
```

Sample Output 2

```
2
3
11
11
28
```

Hint

1. STL is your good friend:
   - Use `std::map` or `std::set` to maintain the cats in ascending order of their x-coordinate.
   - Use `std::vector` or `std::list` when you need arbitrary length array.
   - Go to Cpp Reference if you don't know how to use an STL container or function.
2. Be careful **NOT** to use `std::endl` since it might be very slow. Use `'\n'` instead.
3. Let `c` be an `std::map` or `std::set`. Be careful **NOT** to use `std::lower_bound(c.begin(), c.end(), x)`. Use `c.lower_bound(x)` instead.
4. You should try to prove the time complexity of your code.
5. The test data in Test Group 1 might be a lot weaker than other test groups. Passing this test group does not guaranteed your code is bug-free.