GOBOSORT

Problem

Alice has an array a[1,...,2n]a[1, . . . , 2^n]a[1,...,2n] of integers, but unlike most CS students, she has no interest in sorting it. Instead, she performs a sequence of operations on the array, and after each operation, she is curious how unsorted the array is. That is, she wants to know the number of pairs (i,j)(i,j)(i,j) such that i<ji < ji<j and a[i]>a[j]a[i] > a[j]a[i]>a[j].

The kkk-th operation that Alice performs on the array is given by a parameter qkq_kqk and proceeds as follows: First, she splits the array into 2n−qk2^{n-q_k}2n−qk subarrays of length 2qk2^{q_k}2qk. Next, she reverses each of the individual subarrays. Finally, she concatenates the subarrays into a single array (in the same order) and sets the concatenated array to be the new array aaa.

Given the initial array aaa and a sequence q1,...,qmq_1, . . . , q_mq1,...,qm of mmm such operations, your task is to output how unsorted aaa is after each operation.

Constraints

There are two sample inputs, each worth 0 points. (See “Sample Input” below.) These tests serve to verify that the server is assessing your code correctly, in the event that all of the other test cases are reported as Incorrect Output.

For all other test cases, worth a total of 100 points, it is guaranteed that 0≤n≤170 \le n \le 170≤n≤17, 1≤m≤20001\le m\le 20001≤m≤2000, and 0≤qk≤n0 \le q_k \le n0≤qk≤n for all kkk. These test cases are further divided into three batches, with the following additional constraints:

- in the first batch of test cases, worth 15 points, it is guaranteed that m=1m = 1m=1 and n≤10n\le 10n≤10;
- in the second batch, worth an additional 40 points, it is guaranteed that m=1m = 1m=1;
- the third batch, with no additional constraints, is worth the remaining 45 points.

Time Limit

The time limit is 100ms per test case in the first batch (worth a total of 15 points). For all other test cases, the time limit is 350ms per test case. (These time limits are 3x for Java and 10x for Python.)

Input Format

The input consists of there lines as follows:

- line 1 consists of two space-separated integers, nnn and mmm (the log of the length of aaa and the number of operations, respectively);
- line 2 consists of 2n2^n2n space-separated integers a[1],…,a[2n]a[1],\ldots, a[2^n]a[1],…,a[2n], with 1≤a[i]≤1091 \le a[i] \le 10^91≤a[i]≤109 for all iii;
- line 3 consists of mmm space-separated integers q1,…,qmq_1, \ldots, q_mq1,…,qm (the sequence of operations that Alice performs).

Output Format

Output mmm lines, with the kkk-th line containing the unsortedness of array aaa after kkk operations.

Sample Cases

Sample Input 1

```
2 1
2 1 4 3
1
```

Sample Output 1

```
0
```

Sample Input 2

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


Sample Output 2

```
3
3
3
3
```

Details

In the first sample, the operation makes the array sorted, so the answer is 000.

In the second sample, the array becomes 4 1 2 3 after the first operation. The unsortedness of this new array is 333, with pairs (1,2)(1,2)(1,2), (1,3)(1,3)(1,3), and (1,4)(1,4)(1,4) contributing to the unsortedness.

View submissions

Test cases

Input

```
2 1
2 1 4 3
```

Input

```
1


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



```
17 2000
1 1 2 1 2 1 1 1 2 3 1 1 3 2 1 3...
```

Input truncated for display.

```
8 1
4360 1851 260 9316 5497 9478 4354 4848 4204 3206 3304 1545 2047 6989 6193 1200 2997 4852 2
8229 8509 5152 4063 2210 273 7467 2472 4316 672 8884 9939 9412 9706 5031 8003 7021 6019 70
3687 2209 2159 9185 8073 5921 7983 3463 5546 2638 8952 9140 9292 4198 9777 5402 9178 6085
2


10 1
56 8 71 84 30 13 52 57 90 44 90 2 13 5 21 25 6 10 45 70 3 15 46 46 65 22 28 36 68 50 60 92
58 73 89 49 81 85 35 75 78 67 90 92 79 64 29 37 89 56 55 20 83 20 24 73 21 79 96 98 77 86
```

Input

```
39 78 41 24 11 97 44 76 86 35 97 95 73 71 42 85 14 5 3 6 23 75 36 31 79 52 45 16 48 98 52
93 6 23 82 26 83 91 3 35 19 6 68 72 91 42 72 74 79 48 9 51 44 42 92 30 77 91 21 67 24 79 9
54 100 100 83 89 58 9 20 23 93 86 43 89 62 49 21 23 33 77 35 7 40 17 3 31 67 4 17 100 61 6
45 94 87 74 49 43 17 39 30 75 64 11 23 79 80 50 13 8 64 83 69 40 65 11 10 18 32 86 7 18 86
9
```

```
10 1
9671 9007 5473 1727 9727 8557 71...
```

Input truncated for display.

```
10 1
238363353 [illegible] 934211692 892...
```

Input truncated for display.


```
5 1
5 10 8 10 1 2 4 10 2 3 1 4 2 4 4 7 4 10 6 9 5 4 7 6 3 5 9 3 1 6 7 10
2
```

```
16 1
76326 227391 780095 319045 43850...
```

Input truncated for display.

```
17 1
873627 11117 968759 239494 86939...
```

Input truncated for display.

```
16 1
```

Input

```
11139168 391337048 538883744 535...
```

Input truncated for display.

```
17 1
828199235 320792352 22282234 531...
```

Input truncated for display.

```
17 1
893 948 332 210 822 65 42 371 10...
```

Input truncated for display.


```
16 2000
154198 446323 740217 873584 2633...
```


Input truncated for display.

Input

```
17 2000
777878 607479 237595 859628 8244...
```

Input truncated for display.


```
16 2000
551842459 989875543 830179652 70...
```

Input truncated for display.

Input


```
17 2000
911411059 873575358 192088036 63...
```

Input truncated for display.

```
17 2000
181 69 20 665 464 195 725 927 42...
```

Input truncated for display.

Input

Download

Inspired by the "Ultra Cool Programming Contest Control Centre" by Sonny Chan.
Modified for CS 124 by Neal Wu, with design help from Martin Camacho.
Further refined by Nikhil Benesch.