Breaking String

A certain string-processing language allows the programmer to break a string into two pieces. Since this involves copying the old string, it costs n units of time to break a string of n characters into two pieces. Suppose a programmer wants to break a string into many pieces. The order in which the breaks are made can affect the total amount of time used. For example, suppose we wish to break a 20 character string after characters 3, 8, and 10 (numbering the characters in ascending order from the left-hand end, starting from 1). If the breaks are made in left-to-right order, then the first break cost 20 units of time, the second break costs 17 units of time, and the third breaks costs 12 units of time, a total of 49 units of time (see the sample below). If the breaks are made in right-to-left order, then the first break costs 20 units of time, the second break costs 10 units of time, and the third break costs 8 units of time, a total of 38 units of time.

The cost of making the breaks in left-to-right order:

```
thisisastringofchars      (original)
thi sisastringofchars     (cost:20 units)
thi sisas tringofchars    (cost:17 units)
thi sisas tr ingofchars   (cost:12 units)
                        Total: 49 units.
```

The cost of making the breaks in right-to-left order:

```
thisisastringofchars      (original)
thisisastr ingofchars     (cost:20 units)
thisisas tr ingofchars    (cost:10 units)
thi sisas tr ingofchars   (cost: 8 units)
                        Total: 38 units.
```

Input:

There are several test cases! In each test case, the first line contains 2 integers N (2<=N<=10000000) and M (1<=M<=1000, M<N). N is the original length of the string, and M is the number of the breaks. The following lines contain M integers Mi (1<=Mi<N) in ascending order that represent the breaking positions from the string's left-hand end. Read input till EOF.

(There wont be more than **100** cases)

Output:

For each test case, output in one line the least cost to make all the breakings.

Sample Input:

```
20 3
3 8 10
20 4
2 3 8 10
```

Sample Output:

```
37
40
```