Sparse Arrays | HackerRank

hackerrank.com/challenges/sparse-arrays/problem?isFullScreen=true

Verify it's you | New Chrome available

HackerRank | Prepare | Data Structures | Arrays | Sparse Arrays | Exit Full Screen View

Problem | Submissions | Leaderboard | Discussions

There is a collection of input strings and a collection of query strings. For each query string, determine how many times it occurs in the list of input strings. Return an array of the results.

Example

$stringList = ['ab', 'ab', 'abc']$

$queries = ['ab', 'abc', 'bc']$

There are 2 instances of 'ab'. 1 of 'abc', and 0 of 'bc'. For each query, add an element to the return array: $results = [2, 1, 0]$.

Function Description

Complete the function $matchingStrings$ with the following parameters:

- $string\ stringList[n]$: an array of strings to search
- $string\ queries[q]$: an array of query strings

Returns

- $int[q]$: the results of each query

Input Format

The first line contains and integer n, the size of $stringList[]$.

Each of the next n lines contains a string $stringList[i]$.

The next line contains q, the size of $queries[]$.

Each of the next q lines contains a string $queries[i]$.

Constraints

$1 \leq n \leq 1000$

Change Theme | Language C++11

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;

int main() {
    int n;
    cin >> n;

    vector<string> strings(n);
    for (int i = 0; i < n; i++) {
        cin >> strings[i];
    }

    int q;
    cin >> q;

    vector<string> queries(q);
    for (int i = 0; i < q; i++) {
        cin >> queries[i];
    }

    vector<int> result(q, 0);

    for (int i = 0; i < q; i++) {
```

Line: 9 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

Search | Rainy days ahead 25°C | ENG IN | 23:26 21-11-2025

Sparse Arrays | HackerRank

hackerrank.com/challenges/sparse-arrays/problem?isFullScreen=true

Verify it's you | New Chrome available

HackerRank | Prepare | Data Structures | Arrays | Sparse Arrays | Exit Full Screen View

Problem | Submissions | Leaderboard | Discussions

There is a collection of input strings and a collection of query strings. For each query string, determine how many times it occurs in the list of input strings. Return an array of the results.

Example

$stringList = ['ab', 'ab', 'abc']$

$queries = ['ab', 'abc', 'bc']$

There are 2 instances of 'ab'. 1 of 'abc'. and 0 of 'bc'. For each query, add an element to the return array: $results = [2, 1, 0]$.

Function Description

Complete the function $matchingStrings$ with the following parameters:

- $string\ stringList[n]$: an array of strings to search
- $string\ queries[q]$: an array of query strings

Returns

- $int[q]$: the results of each query

Input Format

The first line contains and integer n, the size of $stringList[]$.

Each of the next n lines contains a string $stringList[i]$.

The next line contains q, the size of $queries[]$.

Each of the next q lines contains a string $queries[i]$.

Constraints

$1 \leq n \leq 1000$

Upload Code as File | Test against custom input | Run Code | Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

- Sample Test case 0
- Sample Test case 1
- Sample Test case 2

Input (stdin) | Download

```
4
aba
baba
aba
xzxb
3
aba
xzxb
ab
```

Your Output (stdout)

```
2
```

Search | NZ - WI In 7 hours | ENG IN | 23:27 21-11-2025

Left Rotation | HackerRank

hackerrank.com/challenges/array-left-rotation/problem?isFullScreen=true

Verify it's you | New Chrome available

HackerRank | Prepare | Data Structures | Arrays | Left Rotation | Exit Full Screen View

Problem | Submissions | Leaderboard | Discussions

A *left rotation* operation on a circular array shifts each of the array's elements 1 unit to the left. The elements that fall off the left end reappear at the right end. Given an integer d, rotate the array that many steps to the left and return the result.

Example

$d = 2$

$arr = [1, 2, 3, 4, 5]$

After 2 rotations, $arr' = [3, 4, 5, 1, 2]$.

Function Description

Complete the *rotateLeft* function with the following parameters:

- *int* d: the amount to rotate by
- *int* $arr[n]$: the array to rotate

Returns

- $int[n]$: the rotated array

Input Format

The first line contains two space-separated integers that denote n, the number of integers, and d, the number of left rotations to perform.

The second line contains n space-separated integers that describe $arr[]$.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq d \leq n$
- $1 \leq a[i] \leq 10^6$

Change Theme | Language C++11

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n, d;
    cin >> n >> d;

    vector<int> arr(n);
    for(int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    // Reduce unnecessary rotations
    d = d % n;

    vector<int> result(n);

    for(int i = 0; i < n; i++) {
        result[i] = arr[(i + d) % n];
    }

    for(int x : result) {
        cout << x << " ";
    }
```

Line: 15 Col: 15

Upload Code as File | Test against custom input | Run Code | Submit Code

Search | Rain coming In about 2.5 hours | ENG IN | 23:31 21-11-2025

Left Rotation | HackerRank

hackerrank.com/challenges/array-left-rotation/problem?isFullScreen=true

Verify it's you | New Chrome available

HackerRank | Prepare | Data Structures | Arrays | Left Rotation | Exit Full Screen View

Problem | Submissions | Leaderboard | Discussions

A *left rotation* operation on a circular array shifts each of the array's elements 1 unit to the left. The elements that fall off the left end reappear at the right end. Given an integer d, rotate the array that many steps to the left and return the result.

Example

$d = 2$

$arr = [1, 2, 3, 4, 5]$

After 2 rotations, $arr' = [3, 4, 5, 1, 2]$.

Function Description

Complete the *rotateLeft* function with the following parameters:

- *int* d: the amount to rotate by
- *int* $arr[n]$: the array to rotate

Returns

- $int[n]$: the rotated array

Input Format

The first line contains two space-separated integers that denote n, the number of integers, and d, the number of left rotations to perform.

The second line contains n space-separated integers that describe $arr[]$.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq d \leq n$
- $1 \leq a[i] \leq 10^6$

```
13
14      // Reduce unnecessary rotations
15      d = d % n;
16
17      vector<int> result(n);
18
19      for(int i = 0; i < n; i++) {
20          result[i] = arr[(i + d) % n];
21      }
22
23      for(int x : result) {
24          cout << x << " ";
25      }
```

Line: 30 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) | Download

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

Search | Rainy days ahead 25°C | ENG IN | 23:25 21-11-2025

Left Rotation | HackerRank

hackerrank.com/challenges/array-left-rotation/problem?isFullScreen=true

Verify it's you | New Chrome available

HackerRank | Prepare | Data Structures | Arrays | Left Rotation | Exit Full Screen View

Problem | Submissions | Leaderboard | Discussions

A *left rotation* operation on a circular array shifts each of the array's elements 1 unit to the left. The elements that fall off the left end reappear at the right end. Given an integer d, rotate the array that many steps to the left and return the result.

Example

$d = 2$

$arr = [1, 2, 3, 4, 5]$

After 2 rotations, $arr' = [3, 4, 5, 1, 2]$.

Function Description

Complete the *rotateLeft* function with the following parameters:

- *int* d: the amount to rotate by
- *int* $arr[n]$: the array to rotate

Returns

- *int*$[n]$: the rotated array

Input Format

The first line contains two space-separated integers that denote n, the number of integers, and d, the number of left rotations to perform.
The second line contains n space-separated integers that describe $arr[]$.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq d \leq n$
- $1 \leq a[i] \leq 10^6$

```
27      return 0;
28  }
29
30
```

Line: 15 Col: 15

Upload Code as File | Test against custom input | Run Code | Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) — Download

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

Your Output (stdout)

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

Expected Output — Download

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

Search | Rain coming In about 2.5 hours | ENG IN | 23:31 21-11-2025

- Declare a 2-dimensional array, arr, with n empty arrays, all zero-indexed.
- Declare an integer, $lastAnswer$, and initialize it to 0.

You need to process two types of queries:

1. Query: $1\ x\ y$
   - Compute $idx = (x \oplus lastAnswer)$.
   - Append the integer y to $arr[idx]$.
2. Query: $2\ x\ y$
   - Compute $idx = (x \oplus lastAnswer)$.
   - Set $lastAnswer = arr[idx][y\%size(arr[idx])]$.
   - Store the new value of $lastAnswer$ in an answers array.

Notes:

- $\oplus$ is the bitwise XOR operation, which corresponds to the ^ operator in most languages.

Learn more about it on Wikipedia.

- % is the modulo operator.

- Finally, $size(arr[idx])$ is the number of elements in $arr[idx]$.

Function Description

Complete the $dynamicArray$ function with the following parameters:

- $int\ n$: the number of empty arrays to initialize in arr

- $int\ queries[q][3]$: 2-D array of integers

Returns

- $int[]$: the results of each type 2 query in the order they are presented

Input Format

Change Theme Language C++11

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n, q;
    cin >> n >> q;

    vector<vector<int>> seqList(n);
    int lastAnswer = 0;

    while (q--) {
        int type, x, y;
        cin >> type >> x >> y;

        int idx = (x ^ lastAnswer) % n;

        if (type == 1) {
            seqList[idx].push_back(y);
        }
        else if (type == 2) {
            int size = seqList[idx].size();
            lastAnswer = seqList[idx][y % size];
            cout << lastAnswer << endl;
        }
```

Line: 29 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

- Declare a 2-dimensional array, arr, with n empty arrays, all zero-indexed.
- Declare an integer, $lastAnswer$, and initialize it to 0.

You need to process two types of queries:

1. Query: $1\ x\ y$
   - Compute $idx = (x \oplus lastAnswer)$.
   - Append the integer y to $arr[idx]$.
2. Query: $2\ x\ y$
   - Compute $idx = (x \oplus lastAnswer)$.
   - Set $lastAnswer = arr[idx][y\%size(arr[idx])]$.
   - Store the new value of $lastAnswer$ in an answers array.

Notes:

- $\oplus$ is the bitwise XOR operation, which corresponds to the ^ operator in most languages.

Learn more about it on Wikipedia.

- % is the modulo operator.

- Finally, $size(arr[idx])$ is the number of elements in $arr[idx]$.

Function Description

Complete the $dynamicArray$ function with the following parameters:

- $int\ n$: the number of empty arrays to initialize in arr

- $int\ queries[q][3]$: 2-D array of integers

Returns

- $int[]$: the results of each type 2 query in the order they are presented

Input Format

```
vector<vector<int>> seqList(n);
int lastAnswer = 0;

while (q--) {
    int type, x, y;
    cin >> type >> x >> y;

    int idx = (x ^ lastAnswer) % n;

    if (type == 1) {
        seqList[idx].push_back(y);
    }
    else if (type == 2) {
        int size = seqList[idx].size();
        lastAnswer = seqList[idx][y % size];
        cout << lastAnswer << endl;
    }
```

Line: 29 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

- Declare a 2-dimensional array, arr, with n empty arrays, all zero-indexed.
- Declare an integer, $lastAnswer$, and initialize it to 0.

You need to process two types of queries:

1. Query: $1\ x\ y$
   - Compute $idx = (x \oplus lastAnswer)$.
   - Append the integer y to $arr[idx]$.
2. Query: $2\ x\ y$
   - Compute $idx = (x \oplus lastAnswer)$.
   - Set $lastAnswer = arr[idx][y\%size(arr[idx])]$.
   - Store the new value of $lastAnswer$ in an answers array.

Notes:

- $\oplus$ is the bitwise XOR operation, which corresponds to the ^ operator in most languages.

Learn more about it on Wikipedia.

- % is the modulo operator.
- Finally, $size(arr[idx])$ is the number of elements in $arr[idx]$.

Function Description

Complete the $dynamicArray$ function with the following parameters:

- $int\ n$: the number of empty arrays to initialize in arr
- $int\ queries[q][3]$: 2-D array of integers

Returns

- $int[]$: the results of each type 2 query in the order they are presented

Input Format

Upload Code as File | Test against custom input | Run Code | Submit Code

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) — Download

```
2 5
1 0 5
1 1 7
1 0 3
2 1 0
2 1 1
```

Your Output (stdout)

```
7
3
```

Expected Output — Download