Q4 Traveling is Fun

locked

There are n cities numbered from 1 to n. Two cities, x and y, are connected by a bidirectional road if and only if $gcd(x, y) > g$, where gcd is the greatest common divisor of x and y. Julia is planning a long vacation and wants to know whether a path exists from city x to city y.

Complete the $connectedCities$ function; it has four parameters:

Name	Type	Description
n	integer	The number of cities.
g	integer	Cities x and y are connected if and only if $gcd(x, y) > g$.
originCities	integer array	Each index i (where $0 \leq i \leq q$) describes x for the i^{th} pair of cities.
destinationCities	integer array	Each index i (where $0 \leq i \leq q$) describes y for the i^{th} pair of cities.

The function must return an array of q integers where the value at each index i (where $0 \leq i < q$) is 1 if a path exists from city $originCities_i$ to city $destinationCities_i$; otherwise, it's 0 instead.

Input Format

The code to read the inputs from stdin and to pass it to the function $connectedCities$ is provided for you. The below is documentation in case you need to create custom testcases.

- The first line contains an integer denoting n.
- The second line contains an integer denoting g.
- The third line contains an integer, q, denoting the total number of elements in $originCities$.
- Each line i of q subsequent lines (where $i \leq 0 < q$) contains an integer describing $originCities_i$.
- The next line contains an integer, q, denoting the total number of elements in $destinationCities$.
- Each line i of q subsequent lines (where $i \leq 0 < q$) contains an integer describing $destinationCities_i$.

Constraints

- $2 \leq n \leq 2 \times 10^5$
- $0 \leq g \leq n$
- $1 \leq q \leq min(n \times (n-1)/2, 10^5)$
- $1 \leq originCities_i, destinationCities_i \leq n$, where $0 \leq i < q$
- $originCities_i \neq destinationCities_i$, where $0 \leq i < q$

Output Format

Return an array of q integers where the value at each index i (where $0 \leq i < q$) is 1 if a path exists from city $originCities_i$ to city $destinationCities_i$; otherwise, it's 0 instead.

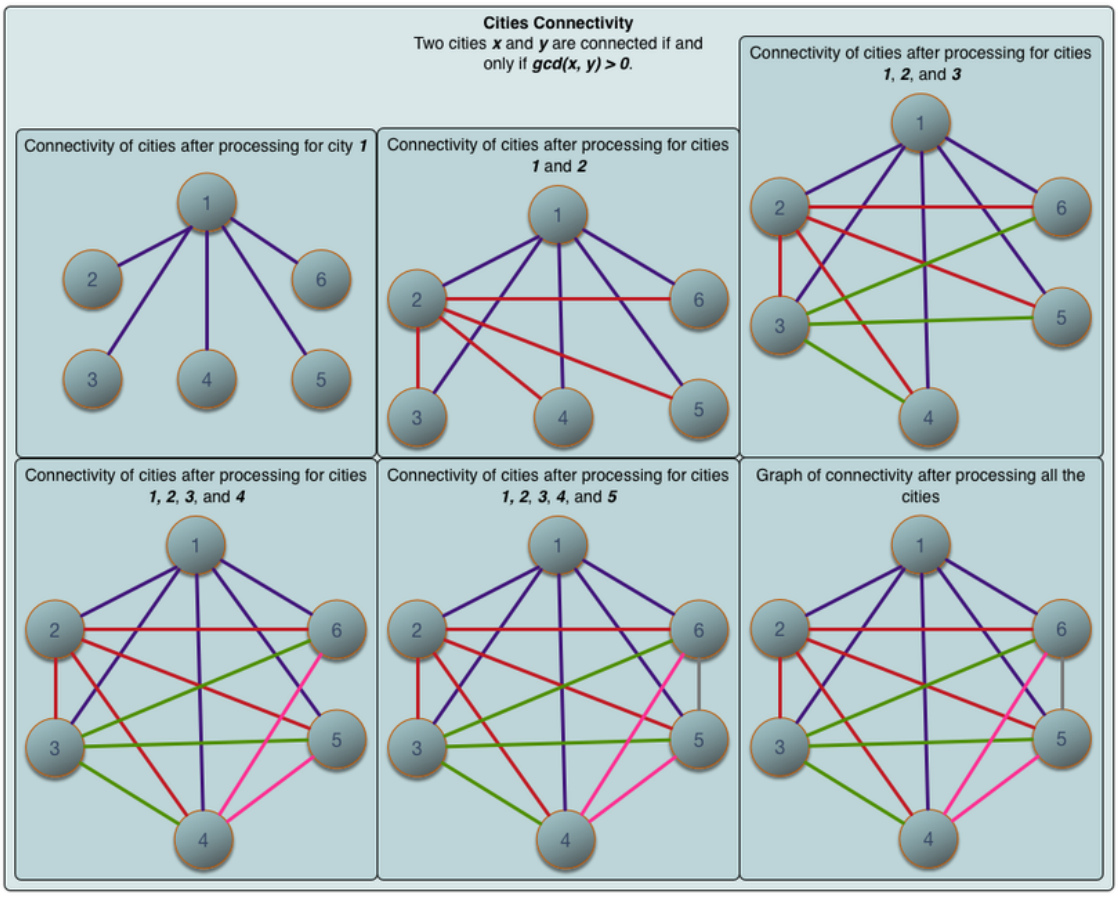
Sample Input 0

```
6
0
4
1
4
3
6
4
3
6
2
5
```

Sample Output 0

```
1
1
1
1
```

Explanation 0



There are $n = 6$ cities and, given $g = 0$, we know that two cities x and y are connected if and only if $gcd(x, y) > 0$. Julia wants to know whether any path exists from:

- City 1 to city 3
- City 4 to city 6
- City 3 to city 2
- City 6 to city 5

Let the return array be $paths$, then:

- $paths_0 = 1$ because a path exists from city 1 to city 3. Julia can follow path 1 → 3.
- $paths_1 = 1$ because a path exists from city 4 to city 6. Julia can follow path 4 → 6.
- $paths_2 = 1$ because a path exists from city 3 to city 2. Julia can follow path 3 → 2.

- $paths_3 = 1$ because a path exists from city 6 to city 5. Julia can follow path 6 → 5.

Thus, we return $paths = [1, 1, 1, 1]$ as our answer.

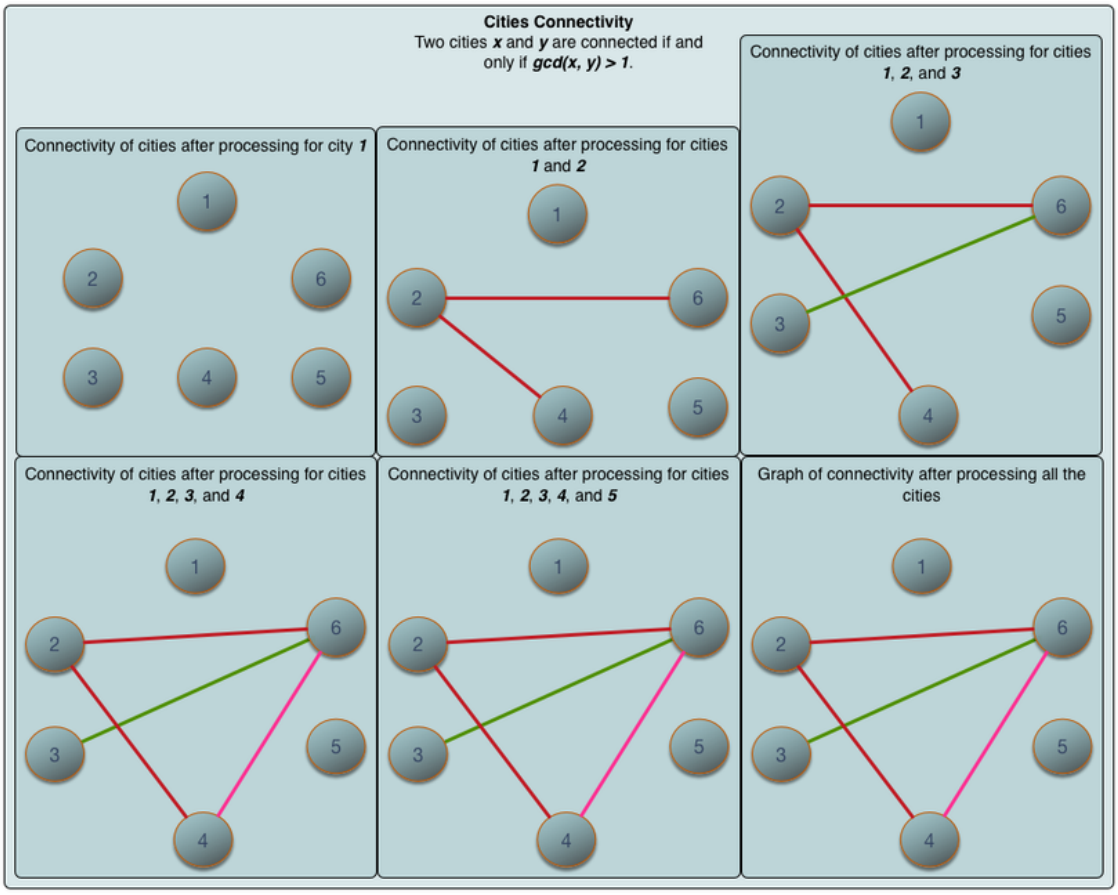
Sample Input 1

```
6
1
4
1
2
4
6
4
3
3
3
4
```

Sample Output 1

```
0
1
1
1
```

Explanation 1



There are $n = 6$ cities and, given $g = 1$, we know that two cities x and y are connected if and only if $gcd(x, y) > 1$. Julia wants to know whether any path exists from:

- City 1 to city 3
- City 2 to city 3
- City 4 to city 3
- City 6 to city 4

Let the return array be $paths$, then:

- $paths_0 = 0$ because it's impossible to reach any city from city 1.
- $paths_1 = 1$ because a path exists from city 2 to city 3. Julia can follow path 2 → 6 → 3.
- $paths_2 = 1$ because a path exists from city 4 to city 3. Julia can follow path 4 → 6 → 3.
- $paths_3 = 1$ because a path exists from city 6 to city 4. Julia can follow path 6 → 4.

Thus, we return $paths = [0, 1, 1, 1]$ as our answer.

Sample Input 2

```
10
1
4
10
4
3
6
4
3
6
2
9
```

Sample Output 2

```
1
1
1
1
```

Submissions: 39
Max Score: 100
Difficulty: Hard

Rate This Challenge:

More

C++14

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


int main() {
    /* Enter your code here. Read input from STDIN. Print output to STDOUT */
    return 0;
}

```

Line: 1 Col: 1

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