# Problem 2316: Count Unreachable Pairs of Nodes in an Undirected Graph

## Problem Information

**Difficulty:** Medium
**Acceptance Rate:** 0.00%
**Paid Only:** No

## Problem Description

You are given an integer

n

. There is an

undirected

graph with

n

nodes, numbered from

0

to

n - 1

. You are given a 2D integer array

edges

where

edges[i] = [a

i

, b

i

]

denotes that there exists an

undirected

edge connecting nodes

a

i

and

b

i

.

Return

the

number of pairs

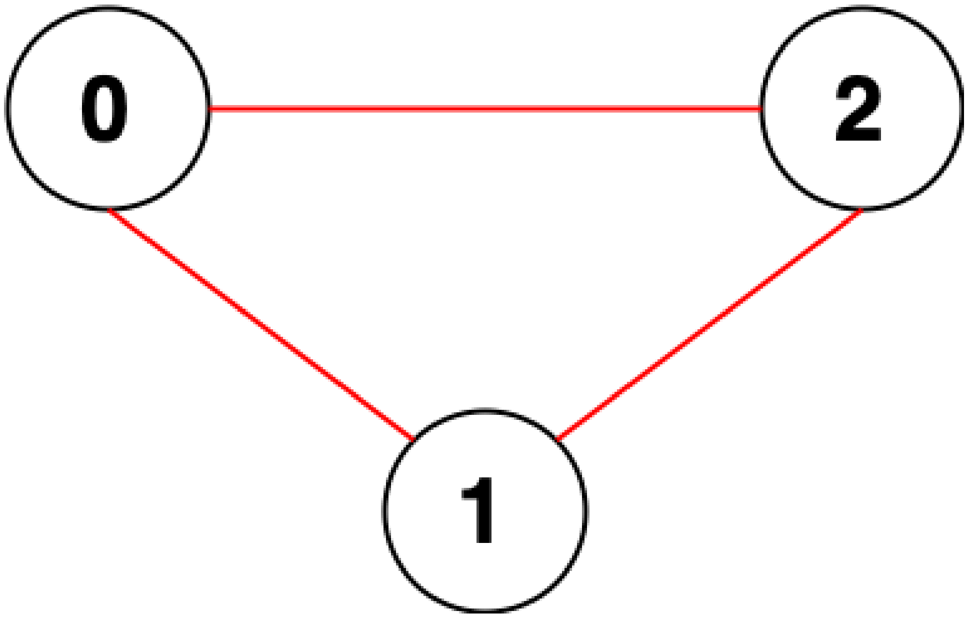
of different nodes that are

unreachable

from each other

.

Example 1:



Input:

n = 3, edges = [[0,1],[0,2],[1,2]]

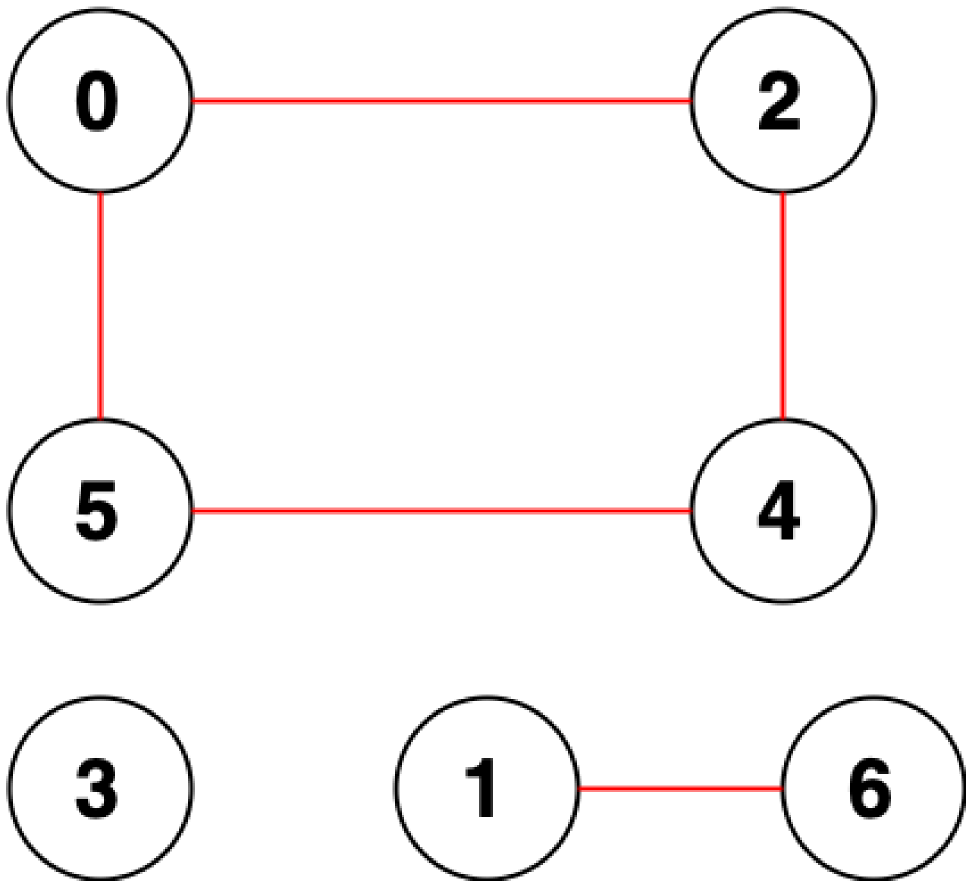
Output:

0

Explanation:

There are no pairs of nodes that are unreachable from each other. Therefore, we return 0.

Example 2:



Input:

n = 7, edges = [[0,2],[0,5],[2,4],[1,6],[5,4]]

Output:

14

Explanation:

There are 14 pairs of nodes that are unreachable from each other: [[0,1],[0,3],[0,6],[1,2],[1,3],[1,4],[1,5],[2,3],[2,6],[3,4],[3,5],[3,6],[4,6],[5,6]]. Therefore, we return 14.

Constraints:

1 <= n <= 10

5

0 <= edges.length <= 2 * 10

5

edges[i].length == 2

0 <= a

i

, b

i

< n

a

i

!= b

i

There are no repeated edges.

## Code Snippets

**C++:**

```
class Solution {
public:
long long countPairs(int n, vector<vector<int>>& edges) {

}
};
```

**Java:**

```
class Solution {
public long countPairs(int n, int[][] edges) {

}
}
```

**Python3:**

```
class Solution:
def countPairs(self, n: int, edges: List[List[int]]) -> int:
```

**Python:**

```
class Solution(object):
def countPairs(self, n, edges):
"""
:type n: int
:type edges: List[List[int]]
:rtype: int
"""
```

**JavaScript:**

```
/**
* @param {number} n
* @param {number[][]} edges
* @return {number}
*/
var countPairs = function(n, edges) {

};
```

**TypeScript:**

```
function countPairs(n: number, edges: number[][]): number {

};
```

**C#:**

```
public class Solution {
public long CountPairs(int n, int[][] edges) {
}
}
```

**C:**

```
long long countPairs(int n, int** edges, int edgesSize, int* edgesColSize) {

}
```

**Go:**

```
func countPairs(n int, edges [][]int) int64 {

}
```

**Kotlin:**

```
class Solution {
fun countPairs(n: Int, edges: Array<IntArray>): Long {

}
}
```

**Swift:**

```
class Solution {
func countPairs(_ n: Int, _ edges: [[Int]]) -> Int {

}
}
```

**Rust:**

```
impl Solution {
pub fn count_pairs(n: i32, edges: Vec<Vec<i32>>) -> i64 {

}
}
```

**Ruby:**

```
# @param {Integer} n
# @param {Integer[][]} edges
# @return {Integer}
def count_pairs(n, edges)

end
```

**PHP:**

```
class Solution {

/**
* @param Integer $n
* @param Integer[][] $edges
* @return Integer
*/
function countPairs($n, $edges) {

}
}
```

**Dart:**

```
class Solution {
int countPairs(int n, List<List<int>> edges) {

}
}
```

**Scala:**

```
object Solution {
def countPairs(n: Int, edges: Array[Array[Int]]): Long = {

}
}
```

**Elixir:**

```
defmodule Solution do
@spec count_pairs(n :: integer, edges :: [[integer]]) :: integer
def count_pairs(n, edges) do
```

```
end
end
```

**Erlang:**

```
-spec count_pairs(N :: integer(), Edges :: [[integer()]]) -> integer().
count_pairs(N, Edges) ->
.
```

**Racket:**

```
(define/contract (count-pairs n edges)
(-> exact-integer? (listof (listof exact-integer?)) exact-integer?)
)
```

## Solutions

### C++ Solution:

```
/*
* Problem: Count Unreachable Pairs of Nodes in an Undirected Graph
* Difficulty: Medium
* Tags: array, graph, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
long long countPairs(int n, vector<vector<int>>& edges) {

}
};
```

### Java Solution:

```
/**
* Problem: Count Unreachable Pairs of Nodes in an Undirected Graph
* Difficulty: Medium
* Tags: array, graph, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public long countPairs(int n, int[][] edges) {

}
}
```

### Python3 Solution:

```
"""
Problem: Count Unreachable Pairs of Nodes in an Undirected Graph
Difficulty: Medium
Tags: array, graph, search

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def countPairs(self, n: int, edges: List[List[int]]) -> int:
# TODO: Implement optimized solution
pass
```

### Python Solution:

```
class Solution(object):
def countPairs(self, n, edges):
"""
:type n: int
:type edges: List[List[int]]
:rtype: int
"""
```

## JavaScript Solution:

```
/**
* Problem: Count Unreachable Pairs of Nodes in an Undirected Graph
* Difficulty: Medium
* Tags: array, graph, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} n
* @param {number[][]} edges
* @return {number}
*/
var countPairs = function(n, edges) {

};
```

## TypeScript Solution:

```
/**
* Problem: Count Unreachable Pairs of Nodes in an Undirected Graph
* Difficulty: Medium
* Tags: array, graph, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function countPairs(n: number, edges: number[][]): number {

};
```

## C# Solution:

```
/*
* Problem: Count Unreachable Pairs of Nodes in an Undirected Graph
* Difficulty: Medium
* Tags: array, graph, search
```

```
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public long CountPairs(int n, int[][] edges) {

}
}
```

### C Solution:

```
/*
* Problem: Count Unreachable Pairs of Nodes in an Undirected Graph
* Difficulty: Medium
* Tags: array, graph, search
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

long long countPairs(int n, int** edges, int edgesSize, int* edgesColSize) {

}
```

### Go Solution:

```
// Problem: Count Unreachable Pairs of Nodes in an Undirected Graph
// Difficulty: Medium
// Tags: array, graph, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func countPairs(n int, edges [][]int) int64 {

}
```

### Kotlin Solution:

```
class Solution {
fun countPairs(n: Int, edges: Array<IntArray>): Long {

}
}
```

### Swift Solution:

```
class Solution {
func countPairs(_ n: Int, _ edges: [[Int]]) -> Int {

}
}
```

### Rust Solution:

```
// Problem: Count Unreachable Pairs of Nodes in an Undirected Graph
// Difficulty: Medium
// Tags: array, graph, search
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn count_pairs(n: i32, edges: Vec<Vec<i32>>) -> i64 {

}
}
```

### Ruby Solution:

```
# @param {Integer} n
# @param {Integer[][]} edges
# @return {Integer}
def count_pairs(n, edges)

end
```

## PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer[][] $edges
* @return Integer
*/
function countPairs($n, $edges) {

}
}
```

## Dart Solution:

```
class Solution {
int countPairs(int n, List<List<int>> edges) {

}
}
```

## Scala Solution:

```
object Solution {
def countPairs(n: Int, edges: Array[Array[Int]]): Long = {

}
}
```

## Elixir Solution:

```
defmodule Solution do
@spec count_pairs(n :: integer, edges :: [[integer]]) :: integer
def count_pairs(n, edges) do

end
end
```

## Erlang Solution:

```
-spec count_pairs(N :: integer(), Edges :: [[integer()]]) -> integer().
count_pairs(N, Edges) ->
.
```

## Racket Solution:

```
(define/contract (count-pairs n edges)
(-> exact-integer? (listof (listof exact-integer?)) exact-integer?)
)
```