



Solution Review: Count the Number of Edges in an Undirected Graph

This review provides a detailed analysis of the different ways to solve the count of the number of edges in graph challenge.

We'll cover the following ^

- Solution #1: Iteration
 - Time complexity
- Solution #2:
 - Time complexity

Solution #1: Iteration#

main.py

Graph.py

Stack.py

Queue.py

LinkedList.py

Node.py

```
1 from Graph import Graph
2 # We only need Graph for this Question!
3
4
```

```
5 def num_edges(g):
6     # For undirected graph, just sum up the size of
7     # all the adjacency lists for each vertex
8     sum = 0
9     for i in range(g.vertices):
10         temp = g.array[i].head_node
11         while temp is not None:
12             sum += 1
13             temp = temp.next_element
14
15     # Half the total sum as it is an undirected graph
16     return sum//2
17
18 if __name__ == "__main__" :
19
20     g = Graph(9)
21     g.add_edge(0, 2)
22     g.add_edge(0, 5)
23     g.add_edge(2, 3)
24     g.add_edge(2, 4)
25     g.add_edge(5, 3)
26     g.add_edge(5, 6)
27     g.add_edge(3, 6)
28     g.add_edge(6, 7)
```



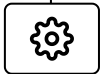
Nothing too tricky going on here. We simply traverse through the complete adjacency list and count the size of each linked list. In an undirected graph, the number of edges is always even as the edges are bidirectional. Hence, to get the number of bidirectional edges, we half the total **sum**.

Time complexity#

$O(V + E)$

Solution #2:#



[main.py](#)[Graph.py](#)[Stack.py](#)[Queue.py](#)[LinkedList.py](#)[Node.py](#)

```
from Graph import Graph
# We only need Graph for this Question!

def num_edges(g):
    # For undirected graph, just sum up the size of
    # all the adjacency lists for each vertex
    return sum([g.array[i].length() for i in range(g.vertices)]) // 2

if __name__ == "__main__" :

    g = Graph(9)
    g.add_edge(0, 2)
    g.add_edge(0, 5)
    g.add_edge(2, 3)
    g.add_edge(2, 4)
    g.add_edge(5, 3)
    g.add_edge(5, 6)
    g.add_edge(3, 6)
    g.add_edge(6, 7)
    g.add_edge(6, 8)
    g.add_edge(6, 4)
    g.add_edge(7, 8)

    g2 = Graph(7)
    g2.add_edge(1, 2)
    g2.add_edge(1, 3)
    g2.add_edge(3, 4)
    g2.add_edge(3, 5)
    g2.add_edge(2, 5)
    g2.add_edge(2, 4)
    g2.add_edge(4, 6)
    g2.add_edge(4, 5)
    g2.add_edge(6, 5)

    print(num_edges(g))

    print(num_edges(g2))
```





Nothing too tricky going on here. It is just a compact version of writing the code. We are using the length function to get the size and we half the total **sum**.

Time complexity#

$O(V + E)$

Interviewing soon? We've partnered with Hired so that companies apply to you instead of you applying to them. [See how](#) ⓘ




← Back

Next →

Challenge 5: Count Number of Edges i...

Challenge 6: Check if a Path Exists Bet...

☒ Mark as Completed

 Report an Issue



