Edge counting

Help Center

In an earlier activity, we wrote a simple function to count edges. However, this basic little function can be written numerous ways. It is instructive to look at some possible implementations and explore the trade-offs. Consider the following implementation of edge_count:

```
def ec1(g):
    """
    Count the number of edges in an undirected graph.

    Arguments:
    g -- undirected graph

    Returns:
    The number of edges in g.
    """
    edges = 0
    for s in g:
        for t in g[s]:
            edges += 1
    return edges / 2
```

This implementation iterates through all of the nodes, *s*, in the graph *g*, then iterates through each neighbor, *t*, of node *s*, and adds one to the total edge count, *edges*. When done, the final count is divided by 2, since an undirected edge will appear twice in the graph, once in each direction. This implementation will obviously yield the correct result, but is it good? In today's lab we will discuss this, as well as several other implementations.

Here are some bad implementations. Why?

```
def ecbad1(g):
    """
    Count the number of edges in an undirected graph.

    Arguments:
    g -- undirected graph

    Returns:
    The number of edges in g.
    """
    edges = set()
    for s in g:
        for t in g[s]:
```

```
            edges.add(frozenset([s, t]))
    return len(edges)
```

```
def ecbad2(g):
    """
    Count the number of edges in an undirected graph.

    Arguments:
    g -- undirected graph

    Returns:
    The number of edges in g.
    """
    edges = 0
    for s in range(len(g)):
        for t in g[s]:
            edges += 1
    return edges / 2
```

Created Sun 24 Aug 2014 12:40 AM BST

Last Modified Sun 24 Aug 2014 6:00 AM BST