Challenge: Graph validity

These problems are not officially a part of lab. While you do not need to continue, as it's the first lab of the semester, there is obviously more to learn! Further, you may find this useful as you test your code.

Exercise 1 - Graph Validity

One interesting observation about our dictionary representation of graphs is that each edge in the graph is in essence recorded twice. For example in the case of a two node, one edge graph:

```
graph = { "0" : set(["1"]),
          "1" : set(["0"]) }
```

we could test to see if an edge connects nodes `0` and `1` by either examining node `0`'s adjacency set looking for `1` **or** by examining `1`'s adjacency set looking for `0`. If an edge connects these two nodes together then it doesn't matter which of the two sets we choose to examine--either will give us the answer. Node `0`'s adjacency set containing `1` implies that `1`'s set contains `0` and vice versa. Restated, if the answer to the question "Is node 0 connected to node 1?" is "Yes" then the answer to the question "Is node 1 connected to node 0?" must also be "Yes." That is unless we have made a *mistake* when creating the data structure representation of our graph. For example, what if we forgot to add node `0` to node `1`'s adjacency set?

```
illogical_graph = { "0" : set(["1"]),
                    "1" : set([]) }
```

It would be useful to create a function to check our graph structures for such mistakes. As such, write a function

```
def is_undirected_graph_valid(graph):
    """
    Tests whether the given graph is logically valid.

    Asserts for every unordered pair of distinct nodes {n1, n2} that
    if n2 appears in n1's adjacency set then n1 also appears in Â
    n2's adjacency set.  Also asserts that no node appears in Â
    its own adjacency set and that every value that appears in
    an adjacency set is a node in the graph.

    Arguments:
    graph -- The graph in dictionary form to test.

    Returns:
    True if the graph is logically valid.  False otherwise.
    """
    ...
```

that tests whether a given graph is logically valid.

Assert your function produces the following output:

```
>>> graph1 = { "0" : set(["1","2"]),
               "1" : set(["0","2"]),
               "2" : set(["1","0"]) }
>>> is_undirected_graph_valid(graph1)
True
```

```
>>> graph2 = { "0" : set(["1","2"]),
               "1" : set(["0","2"]),
               "2" : set(["1"]) }
>>> is_undirected_graph_valid(graph2)
False
```

```
>>> graph3 = make_complete_graph(100)
>>> is_undirected_graph_valid(graph3)
True
```

```
>>> graph4 = { "0" : set(["1","2"]),
               "1" : set(["0","2"]),
               "2" : set(["1","3"]) }
>>> is_undirected_graph_valid(graph4)
False
```

```
>>> graph5 = { "0" : set(["0"]) }
>>> is_undirected_graph_valid(graph5)
False
```

It is good practice to routinely check the validity of your data structures as you test. You may find this function invaluable as you develop graph algorithms throughout the course.
