Assignment 23, Part I: Prim
Due in class Monday, 3/18

March 13, 2019
CS: DS&A
Proof School

The task for Part I is to implement Prim's algorithm using a binary heap implementation of a priority queue to handle edge selection. I've prepared `weighted_graph.py`, which contains basic code for handing weighted graphs. First, some comments on that code:

- Weighted edges are handled using a `WeightedEdge` class, which keeps the vertices in a set (of size 2). In the old `Graph` class (for unweighted graphs), the internal `edge` attribute was a dictionary mapping vertices to sets of vertices; here it is a dictionary mapping vertices to lists of `WeightedEdge` objects.
- You shouldn't have to directly handle the `edge` attribute, though. There are handler functions for returning the set of vertices, the set of edges connected to a given vertex, and the set of all edges.
- There are also some auxiliary functions for generating cyclic and complete graphs, with random weights.

Some comments on your task:

- Your function should be called `MST_Prim`, and it should return an ordered pair. The first element of the pair is a set of `WeightedEdge` objects corresponding to the minimal spanning tree produced by Prim's algorithm. The second element is the sum of the weights in that tree.
- See the next page for a sample input and output. (Your output might be a little different, depending on how your algorithm handles ties.)
- For the priority queue, you will use the `BinaryHeap` class you wrote for assignment 6. This is the implementation that uses an array.
- Please turn in not only `assignment_23.py`, but also whatever binary heap file you're using. (Perhaps it's called `binary_heap.py`, or `assignment_6.py`; it doesn't really matter.) You can also just copy the binary heap code into your `assignment_23.py` file, if you don't want to import it.

Good luck!!!

(Part II will be implementing Kruskal, using the union-find data structure. You can start it now if you want, but it's not due until next Thursday.)

Sample input:

```
data = [ ('A','B',1), ('B','C',2), ('D','E',3), ('E','F',5), ('G','H',3), ('H','I',2),
('A','D',1), ('B','E',1), ('C','F',3), ('D','G',1), ('E','H',2), ('F','I',4) ]

G = weighted_graph.WeightedGraph(data)

output = MST_Prim(G)

print(output[0])

print(output[1])
```

Output:

```
{({'D', 'A'}, 1), ({'B', 'E'}, 1), ({'B', 'A'}, 1), ({'F', 'C'}, 3), ({'G', 'D'},
1), ({'B', 'C'}, 2), ({'H', 'E'}, 2), ({'H', 'I'}, 2)}

13
```