```
"""
Read graphs in Open Street Maps osm format

Based on osm.py from brianw's osmgeocode
http://github.com/brianw/osmgeocode, which is based on osm.py from
comes from Graphserver:
http://github.com/bmander/graphserver/tree/master and is copyright (c)
2007, Brandon Martin-Anderson under the BSD License
"""
import xml.sax
import copy
import networkx

def download_osm(left,bottom,right,top):
    """ Return a filehandle to the downloaded data."""
    from urllib import urlopen
    fp = urlopen( "http://api.openstreetmap.org/api/0.6/map?
bbox=%f,%f,%f,%f"%(left,bottom,right,top) )
    return fp

def read_osm(filename_or_stream, only_roads=True):
    """Read graph in OSM format from file specified by name or by
stream object.

    Parameters
    ----------
    filename_or_stream : filename or stream object

    Returns
    -------
    G : DiGraph

    Examples
    --------
    >>> G = read_osm(download_osm(-122.33,47.60,-122.31,47.61))
    >>> networkx.write_adjlist(G, "test.adjlist")

    """
    osm = OSM(filename_or_stream)
    G = networkx.DiGraph()

    for w in osm.ways.values():
        if only_roads and 'highway' not in w.tags:
            continue
        if only_roads and 'highway' in w.tags and (w.tags[u'highway']
== 'steps' or w.tags[u'highway'] == 'footway'):
            continue
        if only_roads and 'highway' in w.tags and (w.tags[u'highway']
== 'cycleway' or w.tags[u'highway'] == 'bridleway'):
            continue
        if only_roads and 'highway' in w.tags and w.tags[u'highway']
== 'path':
            continue
        if 'oneway' not in w.tags:
            G.add_path(w.nds, id=w.id, data=w)
        elif w.tags[u'oneway'] == '-1':
            G.add_path(reversed(w.nds), id=w.id, data=w)
        else:
            G.add_path(w.nds, id=w.id, data=w)
        if 'oneway' not in w.tags and 'junction' not in w.tags:
            G.add_path(reversed(w.nds), id=w.id, data=w)
        if 'oneway' not in w.tags and 'junction' in w.tags and 
w.tags[u'junction'] != 'roundabout':
            G.add_path(reversed(w.nds), id=w.id, data=w)
    for n_id in list(G.nodes()):
        n = osm.nodes[n_id]
        G.nodes[n_id]['data'] = n
    return G


class Node:
    def __init__(self, id, lon, lat):
        self.id = id
        self.lon = lon
        self.lat = lat
        self.tags = {}

class Way:
    def __init__(self, id, osm):
        self.osm = osm
        self.id = id
        self.nds = []
        self.tags = {}

    def split(self, dividers):
        # slice the node-array using this nifty recursive function
        def slice_array(ar, dividers):
            for i in range(1,len(ar)-1):
                if dividers[ar[i]]>1:
                    #print "slice at %s"%ar[i]
                    left = ar[:i+1]
                    right = ar[i:]

                    rightsliced = slice_array(right, dividers)

                    return [left]+rightsliced
            return [ar]

        slices = slice_array(self.nds, dividers)

        # create a way object for each node-array slice
        ret = []
        i=0
        for slice in slices:
            littleway = copy.copy( self )
            littleway.id += "-%d"%i
            littleway.nds = slice
            ret.append( littleway )
            i += 1

        return ret
```

```
class OSM:
    def __init__(self, filename_or_stream):
        """ File can be either a filename or stream/file object."""
        nodes = {}
        ways = {}

        superself = self

        class OSMHandler(xml.sax.ContentHandler):
            @classmethod
            def setDocumentLocator(self,loc):
                pass

            @classmethod
            def startDocument(self):
                pass

            @classmethod
            def endDocument(self):
                pass

            @classmethod
            def startElement(self, name, attrs):
                if name=='node':
                    self.currElem = Node(attrs['id'],
float(attrs['lon']), float(attrs['lat']))
                elif name=='way':
                    self.currElem = Way(attrs['id'], superself)
                elif name=='tag':
                    self.currElem.tags[attrs['k']] = attrs['v']
                elif name=='nd':
                    self.currElem.nds.append( attrs['ref'] )

            @classmethod
            def endElement(self,name):
                if name=='node':
                    nodes[self.currElem.id] = self.currElem
                elif name=='way':
                    ways[self.currElem.id] = self.currElem

            @classmethod
            def characters(self, chars):
                pass

        xml.sax.parse(filename_or_stream, OSMHandler)

        self.nodes = nodes
        self.ways = ways

        #count times each node is used
        node_histogram = dict.fromkeys( self.nodes.keys(), 0 )
        for way in self.ways.values():
            if len(way.nds) < 2:       #if a way has only one node,
delete it out of the osm collection
                del self.ways[way.id]
            else:
                for node in way.nds:
                    node_histogram[node] += 1

        #use that histogram to split all ways, replacing the member
set of ways
        new_ways = {}
        for id, way in self.ways.items():
            split_ways = way.split(node_histogram)
            for split_way in split_ways:
                new_ways[split_way.id] = split_way
        self.ways = new_ways
```