jdelafon / **hierholzer.py**

Created 3 years ago • Report abuse

Hierholzer's algorithm for Eulerian graphs

hierholzer.py

```
# Hierholzer's algorithm for Eulerian graphs #
"""
EXAMPLE
V = [1,2,3,4,5,6]
E = [(1,2),(2,3),(3,4),(4,5),(5,6),(6,1),(2,6),(6,4),(4,2)]
returns [1, 2, 6, 4, 2, 3, 4, 5, 6, 1]

V = ["AA","AB","BC","CD","DE","EF"]
E = [("AA","AB"),("AB","BC"),("BC","CD"),("CD","DE"),("DE","EF"),("EF","AA"),("AB","EF"),("EF","CD"),("CD","AB")]
returns ['AA', 'AB', 'EF', 'CD', 'AB', 'BC', 'CD', 'DE', 'EF', 'AA']
"""

import time

def outgoing(vertex, edges):
    """Returns the list of edges from *edges* entering into node *vertex*."""
    return [edge for edge in edges if edge[0] == vertex]

def incoming(vertex, edges):
    """Returns the list of edges from *edges* coming from node *vertex*."""
    return [edge for edge in edges if edge[1] == vertex]

def walk(vertex, edges):
    """From node *vertex*, walk along edges *edges*, never taking an already
    used one. Return the chosen *path* and the remaining edges.
    If the graph is Eulerian, *path* is a cycle."""
    path = [vertex]; adj = outgoing(vertex,edges)
    while adj:
        e = adj[0]
        edges.remove(e)
        path.append(e[1])
        vertex = e[1]
        adj = outgoing(vertex,edges)
    return path, edges

def hierholzer(vertices, edges):
    """Finds an Eulerian cycle in a connected Eulerian graph defined
    by the set *vertices* of its vertices and the set *edges* of its edges.
    The cycle is returned as a list of vertices."""
    err = "Your graph is either not Eulerian, or connected, or cyclic."
    assert all([(len(outgoing(v,edges))+len(incoming(v,edges)))%2==0 for v in vertices]), err
    t1 = time.time()
    v = vertices[0]
    cycle, edges = walk(v, edges)
    assert cycle[0] == cycle[-1], err
    notvisited = set(cycle)
    while len(notvisited) != 0:
        v = notvisited.pop()
        if len(outgoing(v, edges)) != 0:
            i = cycle.index(v)
            sub, E = walk(v, edges)
            assert sub[0] == sub[-1], err
            cycle = cycle[:i]+sub[:-1]+cycle[i:]
            notvisited.update(sub)
    t2 = time.time()
    print "Running time: %s" % (t2-t1,)
    return cycle

```