Eulerian path

Eulerian cycle/path, Chinese postman problem

beCP Training

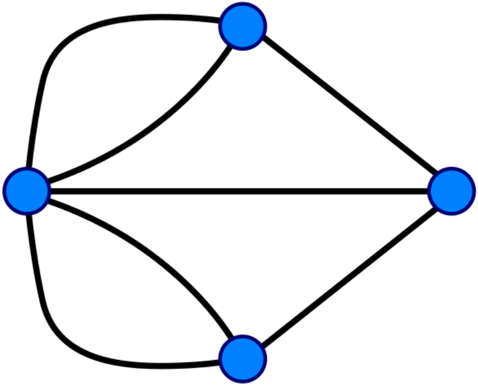
OLYMPIADE BELGE D'INFORMATIQUE
BELGISCHE INFORMATICA-OLYMPIADE

April 15, 2016

Table of Contents

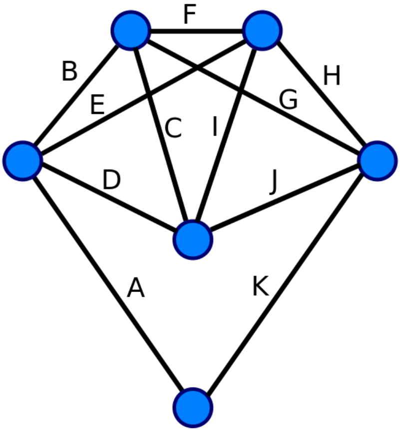
Euler's problem

Can we make a cycle/path that visits every edge exactly once?



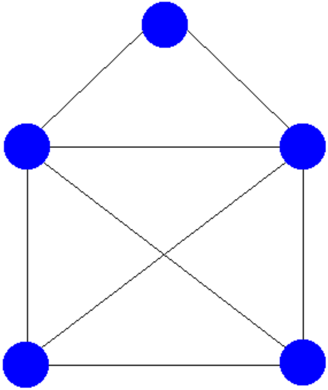
Eulerian graph

A eulerian graph is a graph that has a Eulerian cycle (must come back to the start).



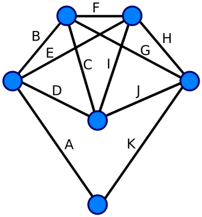
Semi-eulerian graph

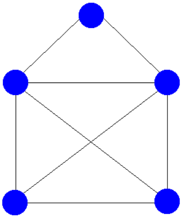
A semi-eulerian graph is a graph that has a Eulerian cycle *or* path (start and end may differ).

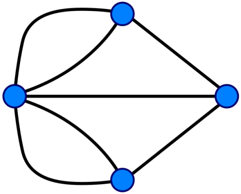


Criteria for undirected graphs

- The graph must be connected
- For each node except the start and end, we use two edges: one for entering and one for leaving!
- So all degrees will be even, except for at most two







Criteria for directed graphs

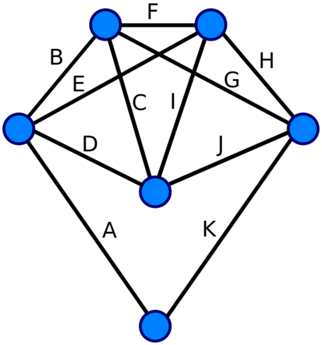
Too complicated...

Just check Wikipedia.

Table of Contents

Traversal strategy

- Start at one of the odd-degree nodes (if they exist)
- Take arbitrary edges and remove them on the way
- The only possible dead-end is the ending node!
- If there are edges left, restart in the middle



Implementation

To erase edges, we need to remember an "edge ID"

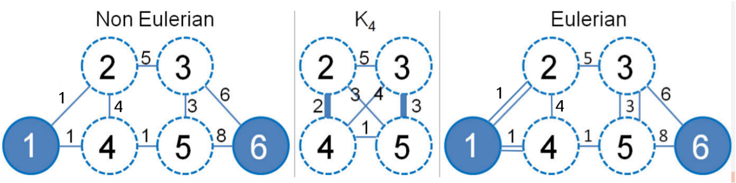
```
vector<int> id[MAXV], neigh[MAXV];
bool visited[MAXE]; // which edges have been taken

// Start on an odd-degree node (if possible)
void euler(int u, vector<int> &s)
{
    for (int i=0; i < (int)neigh[u].size(); i++) {
        if (!visited[id[u][i]]) {
            visited[id[u][i]] = true;
            euler(neigh[u][i], s);
        }
    }
    s.push_back(u);
}
```

Table of Contents

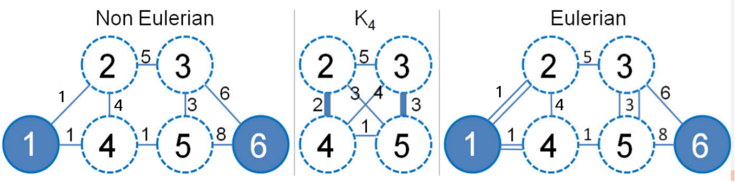
Chinese postman statement

- A postman has to make a cycle through every street at least once, while travelling the smallest possible distance.
- If the graph is Eulerian, then just take a Eulerian cycle
- Otherwise, there are an odd number of odd-degree nodes
- Let's add edges to make it Eulerian!



Choosing the right edges to add

- We have to add an edge to every odd-degree node
- But not change the parity of even-degree node
- So we have to add paths between pairs of odd-degree nodes
- Compute the best paths and put it in a graph
- Find the best pairings with complete search



Sources of figures

- https://commons.wikimedia.org/wiki/File:Königsberg_graph.svg
- https://commons.wikimedia.org/wiki/File:Labelled_Eulergraph.svg
- https://en.wikibooks.org/wiki/File:Eulerian3.png
- CP3 book