Advanced Graph Algorithms 19.04.2012.

Eulerian graphs

1. Definition. *A graph is Eulerian if it has an Eulerian circuit.*

Application: Chinese postman problem: postman has to visit every street (edges), how to optimize the route?

Euler circuits

1. Theorem. *A graph G is Eulerian* $\Leftrightarrow$ *the degree of every vertex in G is even.*

1. Proof. *(*$\Rightarrow$*) : Let C circuit, pick a vertex v in V(C)*
deg(v) = 2 #(visits)* $\Rightarrow$ *even*

($\Leftarrow$*) : By induction on m:*

- *m = 3 ok. (fully connected garph with 3 vertices)*
- *G with m+1 edges: let's assume it has no cycle, so it's a forest, but every tree has a leaf (vertex of degree 1, which is odd), but in G every vertex must be even*
  $\Rightarrow$ *G contains a cycle*
  let C a cycle in G
  $G_1 \cup G_2 \cup \cdots \cup G_N$ *= G* $\setminus$ *E(C) where* G_i *are the components*
  G_i *still have only even vertices (from each component 2 edges were removed)*
  $\Rightarrow G_i$ *Eulerian (#(vertices)* $\leq$ *m)*
  $\Rightarrow C_1 \cup C_2 \cup \cdots \cup C_N \cup C$ *Euler circuit in G*

Euler trails

2. Theorem. *G contains Euler trail* $\Leftrightarrow$ *G has 0 or 2 odd-degree vertices.*

2. Proof. *(*$\Rightarrow$*) : let s start, t end of the trail, if s=t then we have 0 odd vertices, else 2*

($\Leftarrow$*) : 0 : easy*
2 : s and t are odd. connect s and t with an additional edge e. we got a graph with only even vertices.
previous theorem $\Rightarrow$ *it has an Euler circuit.*
remove e. Euler circuit minus one edge is Euler trail. so G has an Euler trail.

To check if the graph has an Euler circuit: $O(n+m)$
How to find this circuit?

2. Definition. *A bridge in a graph is an edge whose deletion increases the number of components.*

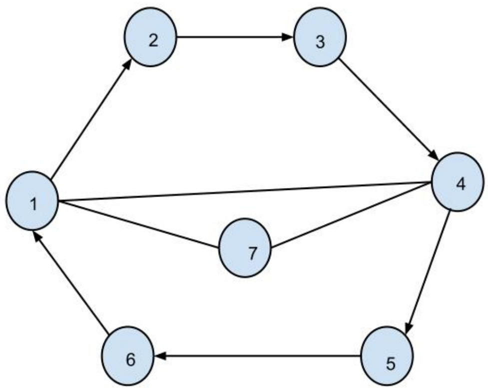
1. Observation. *In G e is bridge* $\Leftrightarrow$ *G doesn't contain a cycle through e.*

1. Corollary. *An Eulerian graph does not contain any bridge.*

Algorithm (first try, which fails because it does not deal with bridges)

1. start with an arbitrary vertex v, $C = \{v\}$
2. choose iteratively incident edge e to v, s.t. e is not in C
3. add e to C
4. repeat with setting v to the end point of e

Counterexample, where this algorithm fails:



Algorithm (Fleury 1883)

1. start with arb. vertex v (for Euler trail v is odd degree vertex if exists), $C = \{v\}$
2. as long as $G \setminus E(C)$ contains incident edges to v :
   1. choose incident edge $e = vw$ that is no bridge in $G \setminus E(C)$ unless there is no alternative
   2. add e to C, set $v := w$

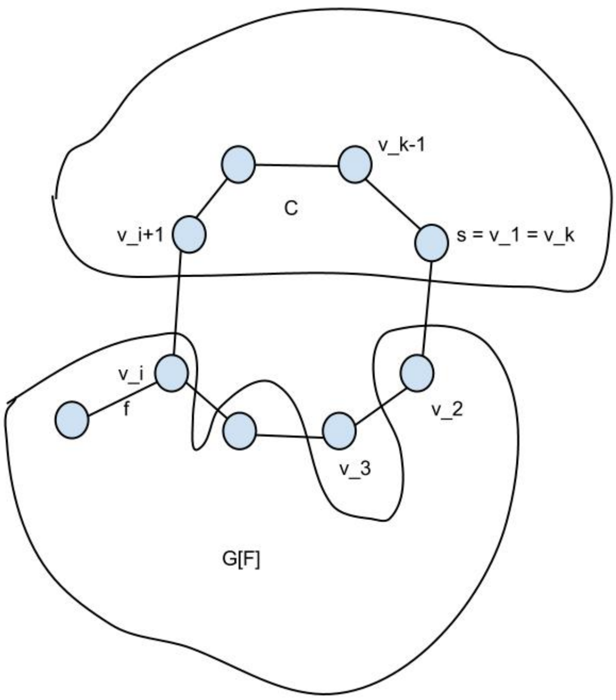
3. Definition. *Let X subset of V. The (vertex-) induced subgraph $G[X]$ is the subgraph of G with vertex set X and with every edge in G having both end points in X.*

4. Definition. *Let X subset of E. The edge-induced subgraph is the subgraph of G with edge set X and every vertex in G which is an end point of some x in X.*

3. Proof. *(Correctness of Fleury's algorithm):*

- *C is a walk*
- *C is a trail: we are not visiting any edge twice (we don't take from C)*
- *C ends at start vertex (closed trail): can't stop before, because that would mean that there is an odd vertex, so there is another edge going out (2.1.)*
- *C is Eulerian:*
  Assume C is not Eulerian:

  Consider $F = E(G) \setminus E(C)$ and $G[F]$
  F is not empty
  s (starting vertex) is not in $G[F]$ (alg. ends when no more incident edges are found so every edge incident to s is in E(C))
  let v_i be the last visited vertex on C that is in $G[F]$
  $v_i v_{i+1}$ is a chosen edge by algorithm but since it is in $G[F]$ there must be another edge which we didn't visit, call it f
  $v_{i+1} \dots v_k$ are not incident to any edge in $G[F]$
  so $v_i v_{i+1}$ must be a bridge when chosen by the algorithm
  $\Rightarrow$ f is also bridge (alg choses bridge only if no other alternative) in $G \setminus (v_1 v_2, v_2 v_3, v_{i-1} v_i)$
  but $G[F]$ is Eulerian (all edges have even minus even degrees) $\Rightarrow$ contradiction (f is a bridge, and Eulerian must not contain a bridge)



Running time: $O(m^2)$
with dynamically checking bridges (with complicated data structures): $O(m \log^3 m)$
There is an even better algorithm: $O(m)$:

Algorithm (Hierholzer 1873)

1. choose starting vertex s, follow an arbitrary trail back to s
2. add to C_1 those edges which were visited
3. let C denote the to-be-constructed Euler circuit. first, set $C := C_1$
4. as long as there is some incident edge to some vertex v_i in C_i (until the last constructed circuit):
    1. ignore the edges of C_i and continue with the remaining graph G_i
    2. start another closed trail C_{i+1} from v_i using the edges of G_i
    3. glue together C_{i+1} with C in the following way:
        1. start with the edges of C until v_i is reached
        2. insert the edges of C_{i+1} after the node v_i
        3. continue with the remaining edges of C
        4. set C to this new circuit

C will be an Eulerian circuit.