Paths, Cycles, Trees

1 Eulerian Circuits

1.1 Definition

A circuit through a graph G so that we start and finish at the same vertex and traverse each edge exactly once

1.2 Theorem

A connected graph with at least two vertices has an Eulerian circuit iff each of its vertices has an even degree

1.3 Idea of Proof

Necessity (⇒): each time this circuit passes through a vertex v, it contributes 2 to deg(v). Since each edge is used exactly once, deg(v) must be even
Sufficiency (⇐): Induction on the number of vertices in G.
Induction base: G=K_3, the claim is obvious. Induction step:

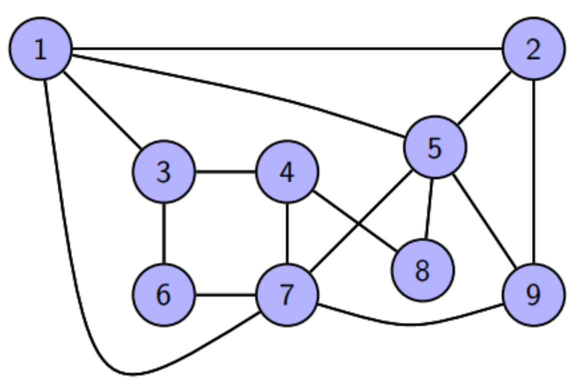
- Start walking from any vertex u along the untraversed edges, marking off the traversed edges
- Stop when you arrive at a vertex where you can't continue (all edges leading from it are already traversed). This vertex must be u again
- Hence we have a circuit C. Delete all edges in C from G to obtain a smaller graph H where all degrees are also even
- By induction hypothesis, each connected component of H has an Eulerian circuit
- Combine C and these circuits to obtain the required circuit for G

2 Hamiltonian Cycles

2.1 Definition

A cycle where we start and finish at the same vertex and visit each vertex exactly once

2.2 Example



- Does this graph has a Eulerian circuit? - No
- Does it have a Hamiltonian cycle? - Yes
- Detecting Eulerian circuits algorithmically is easy (How?) - compute degrees
- Detecting Hamiltonian cycles is hard (NP-Complete)

3 Travelling Salesman Problem

The TSP is the following problem:

- A salesman should visit cities $c_1, c_2, ..., c_n$ in some order, visiting each city exactly once are returning to the starting point
- A (positive integer) cost d(i,j) of travel between each pair (c_i, c_j) is known
- Goal: find an optimal (i.e. cheapest) route for the salesman

Given a graph G with a set V of vertices ($|V| = n$) and a set E of edges

- for each vertex v, create a city c_v
- For each pair of distinct $u, v \in V$, set $d(c_u, c_v) = 1$ if $uv \in E$ and $d(c_u, c_v) = 2$ otherwise

Then detecting a hamiltonian cycle in G can be viewed as TSP:

- If G has a Hamiltonian cycle then the cycle is a route of cost exactly n
- If there is a route of cost n then it can't use pairs with cost 2 and so goes through edges of G and hence is a Hamiltonian cycle

4 Trees

4.1 Definitions

Forest - An acyclic graph, i.e. graph **without cycles**
Tree - A connected forest, i.e. a connected acyclic graph

4.2 Spanning trees

$$G' = (V', E') \text{ of a graph } G = (V, E) \text{ is spanning if } V' = V$$

4.2.1 Theorem

Every connected graph contains a spanning tree, i.e., a spanning subgraph that is a tree.

4.2.2 Proof

Let G be a connected graph:

- If G contains no cycles, it is a tree, and hence a spanning tree of itself
- If G contains a cycle, we can remove one edge from the cycle
- The new graph is still connected (Why?)
- Repeating this we can destroy all cycles and end up with a spanning tree

It follows that trees are the smallest connected structures

4.3 Leaves

A left in a tree is a vertex of degree 1

4.3.1 Lemma

Every tree on at least 2 vertices contains a leaf

4.3.2 Proof

By contraposition:

- Assuming that every vertex has degree 0 or at least 2, we will show that the graph is not a tree
- If a vertex has degree 0, then the graph (which contains at least 2 vertices) is not connected, hence not a tree
- If every vertex has degree at least 2, just start at a vertex, go to one if its neighbours, from there go to another neighbour etc
- Since the vertex set is finite, at some stage we encounter a vertex we have already visited
- This implies that the graph contains a cycle, so is not a tree

4.4 Edges of trees

How many edges does a tree on n vertices have?

4.4.1 Theorem

A connected graph on n vertices is a tree iff it has n-1 edges

4.4.2 Proof

(⇒) Show, by induction on n, that a tree on n vertices has n-1 edges:

- For small n the lemma holds: a tree on one vertex has no edges; a tree on two vertices has one edge
- Suppose that each tree on n-1 vertices has n-2 edges (induction hypothesis)
- Take a tree on n vertices, for some $n \geqslant 3$
- T contains a leaf v. Consider the graph T-v, it has one vertex less and one edge less than T
- T-v is still connected and (still) acyclic
- T-v is a tree with n-1 vertices, by induction hypothesis it has n-2 edges
- T has one edge more, so n-1 edges

(⇐)

- Assume that G is a connected graph with n vertices and n-1 edges
- Then, as we proved before, G contains a spanning tree T
- By the first part of the proof, T contains exactly n-1 edges
- T is a subgraph of G, and it has the same number of edges as G
- Hence, T and G are the same
- In particular, G is a tree

4.5 Paths in trees

Since a tree is a connected graph, between any two vertices in a tree there is a path, can there be more than one path between two vertices in a tree?

4.5.1 Lemma

Let T be a tree and $u, v \in V(T)$ with $u \neq v$
Then there is a unique path in T between u and v

4.5.2 Proof

By contradiction:

- There is a path between u and v in T, since T is connected
- Suppose there are two paths P and Q in T between u and v, and derive a contradiction
- Let x and y in V(T) be distinct and chosen in such a way that x and y are both P and Q, but between x and y the vertices on P and Q are disjoint. (It is possible that x=u and y=v, but this is not necessarily the case)
- Then the segments of P and Q between x and y together form a cycle
- This contradicts that T is a tree. Hence there is a unique (u,v) path in T

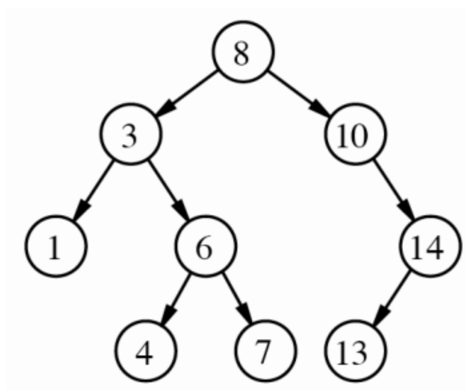
4.6 Rooted trees

4.6.1 Definition

A rooted tree is a tree in which one vertex is fixed as the root(vertex) (and every edge is directed away from this root)

4.6.2 Example

We usually draw a rooted tree in (horizontal) levels, starting with the root (level 0), then the neighbours of the root (level 1), etc



4.7 Rooted trees, children and parents

Let v be a vertex in a rooted tree T

- The neighbours of v in the next level are called the **children** of v
- The (unique) neighbour of v in the previous level (if v is not the root) is called the **parent** of v
- If v has no children then it is called a **leaf** of T
- If v has children, then it is called an **internal vertex**