Graph Theory I

Math 7703

Shagnik Das

NTU

Fall 2021

Chapter 2: Trees

1. Definitions
2. Equivalent characterisations
3. Cayley's formula — Prüfer code

§2.1: Definitions

1. Definitions
   - Forests, trees and leaves
   - Finding leaves
2. Equivalent characterisations
3. Cayley's formula — Prüfer code

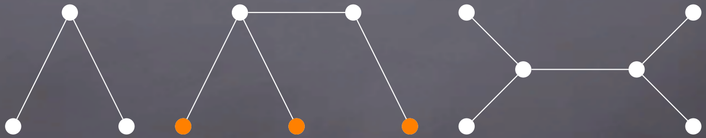
Forests, trees and leaves

Definitions

A forest is a graph without cycles (acyclic).

A tree is a connected acyclic graph.

A leaf (or pendant vertex) is a vertex of degree 1.



A forest

Finding leaves

Lemma

Every finite tree T with at least two vertices has at least two leaves.

Proof

Let P be a longest path in T

- T connected, $v(T) \geq 2 \Rightarrow e(P) \geq 1$
- Longest path $\Rightarrow$ endpoints' neighbours are all on P
- If endpoints have multiple neighbours $\Rightarrow$ cycle in T

$\Rightarrow$ both endpoints are leaves $\square$

Losing leaves

Lemma

Deleting a leaf from an n-vertex tree yields an $(n-1)$-vertex tree.

Proof

Let v be a leaf, $T' = T \setminus v$ the subgraph after deleting v

- T' is connected:
  - Let $u, w \in V(T')$
  - T connected $\Rightarrow$ there is a u, w-path P in T
  - $d(v) = 1 \Rightarrow v$ is not on P

  $\Rightarrow P \subseteq T'$
- T' is acyclic:
  - T is acyclic
  - Deleting a vertex cannot create a cycle

$\Rightarrow T'$ is a tree, and $v(T') = v(T) - 1 = n - 1$ $\square$

§2.2: Equivalent characterisations

1. Definitions
2. Equivalent characterisations
   - Statements
   - Cut-edges
   - Proving equivalence
   - Consequences
3. Cayley's formula — Prüfer code

Statements

Question

Can we easily recognise trees?

Theorem

For an n-vertex simple graph G, the following are equivalent:

(a) *G is connected and acyclic (i.e. G is a tree).*

(b) *G is connected and has $n-1$ edges.*

(c) *G is acyclic and has $n-1$ edges.*

(d) *For every pair $u, v \in V(G)$, there is exactly one u, v-path in G.*

Cut-edges

Definition

An edge of a connected graph is called a cut-edge if deleting it disconnects the graph.

Lemma

An edge in a cycle is not a cut-edge.

Proof

- Let $\{u, v\}$ belong to a cycle, $u, u_1, u_2, \ldots, u_{k-1}, v, u$
- If $\{u, v\}$ not on a path P, path still exists without the edge
- If $\{u, v\}$ is an edge of P:
  - Replace $\{u, v\}$ in the path with $u, u_1, \ldots, u_{k-1}, v$
  - Gives a walk between endpoints avoiding $\{u, v\}$

$\Rightarrow$ Graph is still connected after deleting $\{u, v\}$ $\square$

Proving equivalence

Theorem

For an n-vertex simple graph G, the following are equivalent:

(a) *G is connected and acyclic (i.e. G is a tree).*

(b) *G is connected and has $n-1$ edges.*

(c) *G is acyclic and has $n-1$ edges.*

(d) *For every pair $u, v \in V(G)$, there is exactly one u, v-path in G.*

Strategy

1. Show any two of {'connected', 'acyclic', 'has $n-1$ edges'} implies the third

   $\Rightarrow$ (a), (b) and (c) are equivalent

2. Show (a) and (d) are equivalent

Step 1: (a) $\Rightarrow$ (b), (c)

Claim

If G is connected and acyclic, then $e(G) = n - 1$.

Proof

Induction on $n = v(G)$

- Base case: $n = 1$
  - Cannot have any edges $\Rightarrow e(G) = 0$
- Induction step: $n \geq 2$
  - Lemma: there is a leaf v, and $G \setminus v$ is a tree
  - Induction $\Rightarrow e(G \setminus v) = n - 2$
  - v incident to exactly one edge in $G \Rightarrow e(G) = n - 1$ $\square$

Step 1: (b) $\Rightarrow$ (a), (c)

Claim

If G is connected and $e(G) = n - 1$, then G is acyclic.

Proof

Let G be a connected graph with $n - 1$ edges

- Iteratively remove an edge from every cycle
  - Edges in cycles are not cut-edges
  - $\Rightarrow$ Resulting graph G' is still connected
- G' connected, acyclic $\Rightarrow e(G') = n - 1$
  - $\Rightarrow$ $G' = G$
  - $\Rightarrow$ G is acyclic

$\square$

Step 1: (c) $\Rightarrow$ (a), (b)

Claim

If G is acyclic and $e(G) = n - 1$, then G is connected.

Proof

Suppose G has components $G_1, G_2, \ldots, G_k$ of order $n_1, n_2, \ldots, n_k$

- Each component G_i is connected, acyclic
  $\Rightarrow \; e(G_i) = n_i - 1$

$\Rightarrow \; e(G) = \sum_i e(G_i) = \sum_i (n_i - 1) = \sum_i n_i - k = n - k$

$\Rightarrow \; k = 1$

$\Rightarrow \; G$ is connected $\square$

Step 2: (a) $\Rightarrow$ (d)

Claim

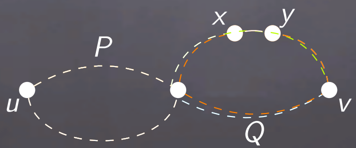
If G is connected and acyclic, then for every pair $u, v \in V(G)$, there is exactly one u, v-path in G.

Proof

- G connected $\Rightarrow$ there is at least one u, v-path
- Suppose there are two distinct paths P and Q
- Let $\{x, y\}$ be an edge in P that is not in Q

  $\Rightarrow$ $P \cup Q \setminus \{\{x, y\}\}$ is an x, y-walk not using $\{x, y\}$

  $\Rightarrow$ contains an x, y-path
- Adding the edge $\{x, y\}$ would create a cycle $\square$



Step 2: (d) $\Rightarrow$ (a)

Claim

If, for every pair $u, v \in V(G)$, there is exactly one u, v-path in G, then G is connected and acyclic.

Proof

- G is connected:
  - There is a path between every pair of vertices
- G is acyclic:
  - Suppose there was a cycle C in G
  - Any pair of vertices on the cycle would have two paths along the cycle $\square$

$\square$

Consequences

Definition

Given a connected graph G, a spanning tree T is a subgraph of G that is a tree containing every vertex of G.

Corollary

(a) *Every n-vertex connected graph has at least $n - 1$ edges and contains a spanning tree.*

(b) *Every edge of a tree is a cut-edge.*

(c) *Adding an edge to a tree creates exactly one cycle.*

Proving the consequences: (a)

Claim

Every n-vertex connected graph has at least $n-1$ edges and contains a spanning tree.

Proof

- Iteratively remove an edge from cycles in G
  - These are not cut-edges

  $\Rightarrow$ final graph G' is still connected
- G' is connected and acyclic $\Rightarrow$ it is a spanning tree
- Lemma $\Rightarrow e(G) \geq e(G') = n-1$ $\square$

Proving the consequences: (b) & (c)

Claim

Every edge of a tree is a cut-edge.

Proof

- Deleting an edge $\to$ graph with $n-2$ edges

$\Rightarrow$ Cannot be connected $\Rightarrow$ edge was a cut-edge $\square$

Claim

Adding an edge $\{u, v\}$ to a tree T creates exactly one cycle.

Proof

- Every cycle C must involve the edge $\{u, v\}$

$\Rightarrow$ Gives a u, v-path $C \setminus \{\{u, v\}\}$

- Unique such path $\Rightarrow$ unique cycle created $\square$

§2.3: Cayley's formula — Prüfer code

1. Definitions
2. Equivalent characterisations
3. Cayley's formula — Prüfer code
   - Counting trees
   - The Prüfer code
   - The inverse map

Counting trees

Question

How many labelled trees on n vertices are there?
Equivalently, how many spanning trees does K_n have?

Initial bounds

- At most the number of n-vertex graphs

  $\#\text{ trees} \leq 2^{\binom{n}{2}} = 2^{\Theta(n^2)}$

- At most the number of n-vertex graphs of size $n-1$

  $\#\text{ trees} \leq \binom{\binom{n}{2}}{n-1} \leq \left(\frac{\binom{n}{2}e}{n-1}\right)^{n-1} = \left(\frac{ne}{2}\right)^{n-1} = 2^{\Theta(n\log n)}$

Theorem (Cayley's Formula; Borchardt, 1860)

The number of labelled n-vertex trees is n^{n-2}.

Proof overview

Strategy

– Removing a leaf from an n-vertex tree $\rightarrow$ $(n-1)$-vertex tree

– Will remove one leaf at a time, until we are left with an edge

– Record enough data about process so that it is reversible

Bijection

– n-vertex tree $\leftrightarrow$ log of the process

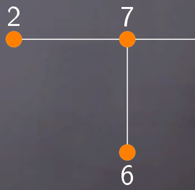
– Log will be a sequence of length $n-2$ with entries in $[n]$

$\Rightarrow$ n^{n-2} possible logs, each corresponds to a unique tree

The Prüfer code

The code

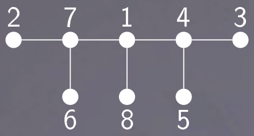
1. Remove the leaf with the lowest label
2. Record the label of its neighbour
3. Repeat until we are left with an edge



From all the leaves ...... choose the one with the lowest label.Remove it f

The final c

Identifying leaves



The code: (7,4,4,1,7,1)

Claim

The leaves of T are the vertices that do not appear in the code.

Proof

- If i is a leaf:
  - If i appears in the code, its neighbour was deleted
  - Then i would be isolated, which is impossible
- If i is not a leaf:
  - A neighbour of i must have been deleted before i was
  - At this step, i would have been added to the code $\square$

The inverse map

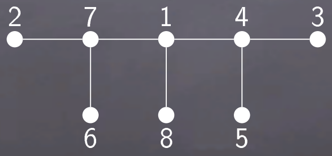
Reconstructing the tree

- We can identify the leaf with the lowest label
- Code tells us which vertex was its neighbour
- Add the edge, then repeat with the remainder of the code

The code: (7,4,4,1,7,1)

Remaining labels: {1,2,3,4,5,6,7,8}

Leaves: {1,2,3,5,4,6,7,8}



Thank you for listening!

Any questions?