Discrete Mathematical Models

Lecture 24

Kane Townsend

Semester 2, 2024

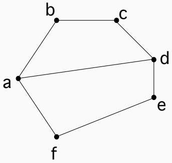
Euler paths and circuits

Euler paths and circuits

- An **Euler path** for a graph is a path that is **not closed** and passes through every edge.

Euler paths and circuits

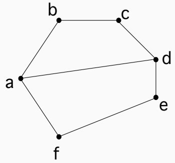
- An **Euler path** for a graph is a path that is **not closed** and passes through every edge. Example:



This graph has an Euler path: *abcdafed*.

Euler paths and circuits

- An **Euler path** for a graph is a path that is **not closed** and passes through every edge. Example:

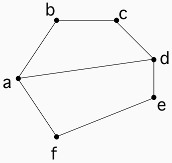


This graph has an Euler path: *abcdafed*.

- An **Euler circuit** for a graph is a circuit passing through every edge.

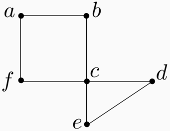
Euler paths and circuits

- An **Euler path** for a graph is a path that is **not closed** and passes through every edge. Example:



This graph has an Euler path: *abcdafed*.

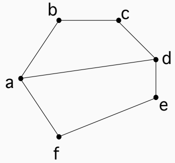
- An **Euler circuit** for a graph is a circuit passing through every edge. Example:



This graph has an Euler circuit: $abcedcfa$.

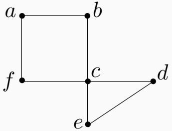
Euler paths and circuits

- An **Euler path** for a graph is a path that is **not closed** and passes through every edge. Example:



This graph has an Euler path: $abcdafed$.

- An **Euler circuit** for a graph is a circuit passing through every edge. Example:



This graph has an Euler circuit: $abcedcfa$.

When do Euler Paths and Circuits exist?

- **Theorem:** A connected graph has an Euler circuit if and only if each of its vertices has even degree.

When do Euler Paths and Circuits exist?

- **Theorem:** A connected graph has an Euler circuit if and only if each of its vertices has even degree.

  In this case we will give an algorithm for finding an Euler circuit.

When do Euler Paths and Circuits exist?

- **Theorem:** A connected graph has an Euler circuit if and only if each of its vertices has even degree.

  In this case we will give an algorithm for finding an Euler circuit.

- **Corollary:** A connected graph has an Euler path if and only if it has exactly two vertices of odd degree.

When do Euler Paths and Circuits exist?

- **Theorem:** A connected graph has an Euler circuit if and only if each of its vertices has even degree.

  In this case we will give an algorithm for finding an Euler circuit.

- **Corollary:** A connected graph has an Euler path if and only if it has exactly two vertices of odd degree.

  The algorithm easily adapts to this case.

Issues regarding finding an Euler Circuit in a graph G

- By the theorem, G must be connected and have all its vertices of even degree. Given this, an Euler circuit must exist.

Issues regarding finding an Euler Circuit in a graph G

- By the theorem, G must be connected and have all its vertices of even degree. Given this, an Euler circuit must exist.
- On a circuit, you can start anywhere and return after completing the circuit. So start the search at any vertex in G.

Issues regarding finding an Euler Circuit in a graph G

- By the theorem, G must be connected and have all its vertices of even degree. Given this, an Euler circuit must exist.
- On a circuit, you can start anywhere and return after completing the circuit. So start the search at any vertex in G.
- As you walk through G mark each edge you use; it cannot be used again. Imagine the edge has been erased, so that graph is 'reduced'.

Issues regarding finding an Euler Circuit in a graph G

- By the theorem, G must be connected and have all its vertices of even degree. Given this, an Euler circuit must exist.
- On a circuit, you can start anywhere and return after completing the circuit. So start the search at any vertex in G.
- As you walk through G mark each edge you use; it cannot be used again. Imagine the edge has been erased, so that graph is 'reduced'.
- After 'erasing' an edge, the start vertex and the 'current' vertex have odd degree in the reduced graph, while all other vertices remain with even degree. You then seek an Euler **path** (in the reduced graph) from the current vertex to the start vertex. By the corollary, such a path exists provided the reduced graph is connected.

Issues regarding finding an Euler Circuit in a graph G

- By the theorem, G must be connected and have all its vertices of even degree. Given this, an Euler circuit must exist.
- On a circuit, you can start anywhere and return after completing the circuit. So start the search at any vertex in G.
- As you walk through G mark each edge you use; it cannot be used again. Imagine the edge has been erased, so that graph is 'reduced'.
- After 'erasing' an edge, the start vertex and the 'current' vertex have odd degree in the reduced graph, while all other vertices remain with even degree. You then seek an Euler **path** (in the reduced graph) from the current vertex to the start vertex. By the corollary, such a path exists provided the reduced graph is connected.
- So choose each edge so that the reduced graph is still connected.

Issues regarding finding an Euler Circuit in a graph G

- By the theorem, G must be connected and have all its vertices of even degree. Given this, an Euler circuit must exist.
- On a circuit, you can start anywhere and return after completing the circuit. So start the search at any vertex in G.
- As you walk through G mark each edge you use; it cannot be used again. Imagine the edge has been erased, so that graph is 'reduced'.
- After 'erasing' an edge, the start vertex and the 'current' vertex have odd degree in the reduced graph, while all other vertices remain with even degree. You then seek an Euler **path** (in the reduced graph) from the current vertex to the start vertex. By the corollary, such a path exists provided the reduced graph is connected.
- So choose each edge so that the reduced graph is still connected.
- Always leave an edge to return to the start vertex as the last step.

Fleury's Algorithm for finding Euler Circuits

Let G be a connected graph with all vertices of even degree.

Fleury's Algorithm for finding Euler Circuits

Let G be a connected graph with all vertices of even degree.

The Algorithm

1. Pick any vertex of G as a starting point.

Fleury's Algorithm for finding Euler Circuits

Let G be a connected graph with all vertices of even degree.

The Algorithm

1. Pick any vertex of G as a starting point.
2. From that vertex **choose any edge to traverse that does not cross a bridge of the current reduced graph, unless there is no other choice**. ("No choice" will only happen when the vertex has degree 1 in the current reduced graph. The new reduced graph will then omit that vertex and so remain connected.)

Fleury's Algorithm for finding Euler Circuits

Let G be a connected graph with all vertices of even degree.

The Algorithm

1. Pick any vertex of G as a starting point.
2. From that vertex **choose any edge to traverse that does not cross a bridge of the current reduced graph, unless there is no other choice**. ("No choice" will only happen when the vertex has degree 1 in the current reduced graph. The new reduced graph will then omit that vertex and so remain connected.)
3. Mark the edge (*e.g.* darken it) as a reminder not to traverse it again. Treat the edge as erased, so reducing the graph.

Fleury's Algorithm for finding Euler Circuits

Let G be a connected graph with all vertices of even degree.

The Algorithm

1. Pick any vertex of G as a starting point.
2. From that vertex **choose any edge to traverse that does not cross a bridge of the current reduced graph, unless there is no other choice**. ("No choice" will only happen when the vertex has degree 1 in the current reduced graph. The new reduced graph will then omit that vertex and so remain connected.)
3. Mark the edge (*e.g.* darken it) as a reminder not to traverse it again. Treat the edge as erased, so reducing the graph.
4. Travel that edge, coming to the next vertex.

Fleury's Algorithm for finding Euler Circuits

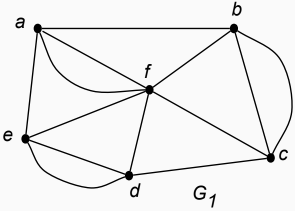
Let G be a connected graph with all vertices of even degree.

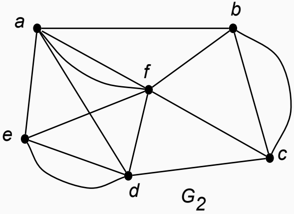
The Algorithm

1. Pick any vertex of G as a starting point.
2. From that vertex **choose any edge to traverse that does not cross a bridge of the current reduced graph, unless there is no other choice**. ("No choice" will only happen when the vertex has degree 1 in the current reduced graph. The new reduced graph will then omit that vertex and so remain connected.)
3. Mark the edge (*e.g.* darken it) as a reminder not to traverse it again. Treat the edge as erased, so reducing the graph.
4. Travel that edge, coming to the next vertex.
5. Repeat steps 2 - 4 until all edges have been traversed, and you are back to the starting vertex.

Candidate Graphs for Fleury's Algorithm

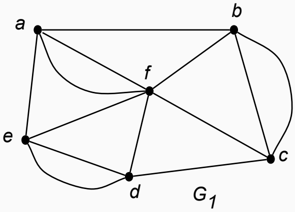
The graph G_1 below satisfies the criterion that all vertices have even degree, so it contains an Euler circuit and Fleury's algorithm can be used to find that circuit.

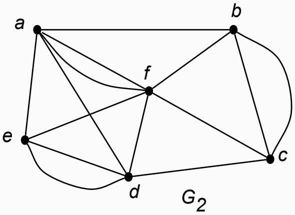




Candidate Graphs for Fleury's Algorithm

The graph G_1 below satisfies the criterion that all vertices have even degree, so it contains an Euler circuit and Fleury's algorithm can be used to find that circuit.

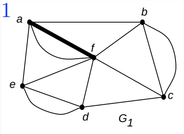


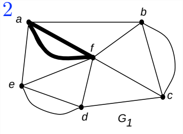


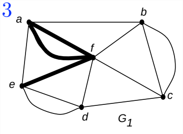
The graph G_2 has two vertices of odd degree. Fleury's algorithm can be modified to find an Euler path in this graph. The only modification needed is that the first vertex must be one of the vertices of odd degree.

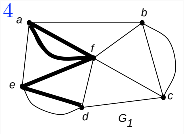
Fleury's Algorithm example in pictures

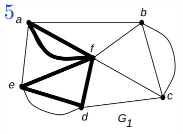
Start at f (just because we feel like it!)

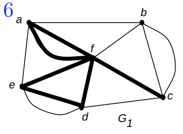


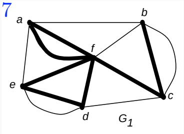








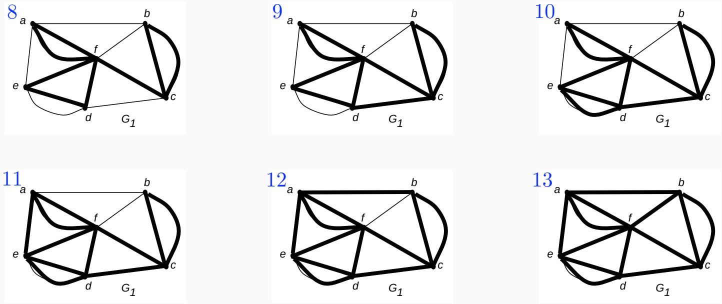




Notice that from here
we MAY NOT step to f
but either a or c is allowed

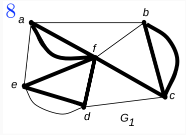
Fleury's Algorithm example in pictures

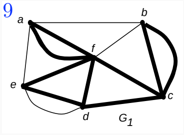
At step 9 we're forced to go to *d*; and then all steps are forced.

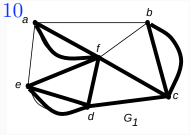


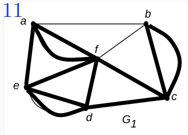
Fleury's Algorithm example in pictures

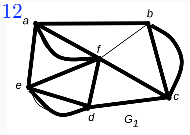
At step 9 we're forced to go to d; and then all steps are forced.

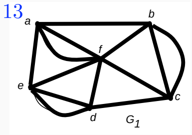








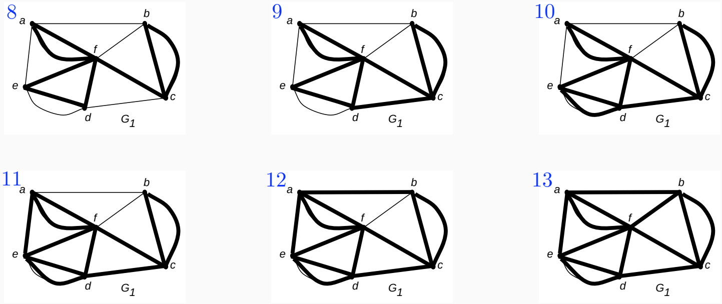




The final path is $fafedfcbcdeabf$.

Fleury's Algorithm example in pictures

At step 9 we're forced to go to d; and then all steps are forced.

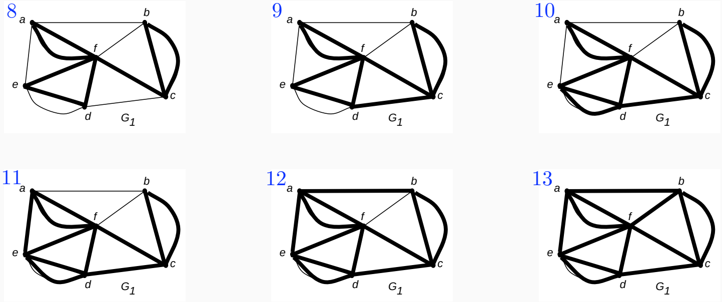


The final path is *fafedfcbcdeabf*.

The path is now closed, providing the Euler circuit we sought.

Fleury's Algorithm example in pictures

At step 9 we're forced to go to d; and then all steps are forced.



The final path is $\qquad$ *fafedfcbcdeabf* .

The path is now closed, providing the Euler circuit we sought.

(As mentioned earlier, we do not call this path an Euler *path*, because, by convention, Euler paths are not closed.)

Fleury's Algorithm example discussion

- Note that if we had started at some other vertex, or made different choices along the way, Fleury's algorithm would have given a different Euler circuit.

Fleury's Algorithm example discussion

- Note that if we had started at some other vertex, or made different choices along the way, Fleury's algorithm would have given a different Euler circuit.

- In any implementation, we never have to back-track, so the algorithm is quite fast; as are some other algorithms to solve this problem.

Fleury's Algorithm example discussion

- Note that if we had started at some other vertex, or made different choices along the way, Fleury's algorithm would have given a different Euler circuit.
- In any implementation, we never have to back-track, so the algorithm is quite fast; as are some other algorithms to solve this problem.
- By contrast, the following problem – that of finding a *Hamilton* path or circuit – has no known 'fast' algorithm.

Applications of Euler Paths and Circuits: Optimal logic gate ordering, reconstructing DNA sequencing, optimising delivery services.

Hamilton paths and circuits

Hamilton Paths and Circuits

- A **Hamilton path** for a graph is a *simple* path which passes through every vertex.

Hamilton Paths and Circuits

- A **Hamilton path** for a graph is a *simple* path which passes through every vertex.
- A **Hamilton circuit** for a graph is a *simple* circuit which passes through every vertex.

Hamilton Paths and Circuits

- A **Hamilton path** for a graph is a *simple* path which passes through every vertex.
- A **Hamilton circuit** for a graph is a *simple* circuit which passes through every vertex.
- Note that a Hamilton path (respectively circuit)
  - does not have to use every edge,

Hamilton Paths and Circuits

- A **Hamilton path** for a graph is a *simple* path which passes through every vertex.
- A **Hamilton circuit** for a graph is a *simple* circuit which passes through every vertex.
- Note that a Hamilton path (respectively circuit)
  - does not have to use every edge,
  - may use an edge at most once by the definition of path (respectively circuit),

Hamilton Paths and Circuits

- A **Hamilton path** for a graph is a *simple* path which passes through every vertex.
- A **Hamilton circuit** for a graph is a *simple* circuit which passes through every vertex.
- Note that a Hamilton path (respectively circuit)
  - does not have to use every edge,
  - may use an edge at most once by the definition of path (respectively circuit),
  - must use each vertex exactly once by the definition of simple.

Hamilton Paths and Circuits

- A **Hamilton path** for a graph is a *simple* path which passes through every vertex.
- A **Hamilton circuit** for a graph is a *simple* circuit which passes through every vertex.
- Note that a Hamilton path (respectively circuit)
  - does not have to use every edge,
  - may use an edge at most once by the definition of path (respectively circuit),
  - must use each vertex exactly once by the definition of simple.
- A graph is called **Hamiltonian** if and only if it possesses a Hamilton circuit.

Hamilton Paths and Circuits

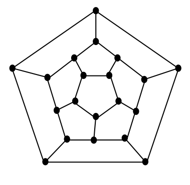
- A **Hamilton path** for a graph is a *simple* path which passes through every vertex.

- A **Hamilton circuit** for a graph is a *simple* circuit which passes through every vertex.

- Note that a Hamilton path (respectively circuit)
  - does not have to use every edge,
  - may use an edge at most once by the definition of path (respectively circuit),
  - must use each vertex exactly once by the definition of simple.
- A graph is called **Hamiltonian** if and only if it possesses a Hamilton circuit.

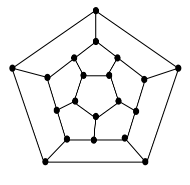
Note: By convention, a Hamilton path must be open, *i.e* not a circuit.

- This graph has a Hamilton Circuit. *Can you find one?*

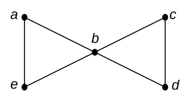


Graphs and Hamilton Paths / Circuits

- This graph has a Hamilton Circuit. *Can you find one?*

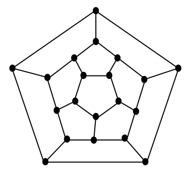


- This graph has no Hamilton circuit. *Why not?*

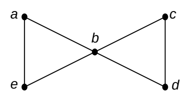


Graphs and Hamilton Paths / Circuits

- This graph has a Hamilton Circuit. *Can you find one?*



- This graph has no Hamilton circuit. *Why not?*



Applications of Hamilton Circuit and Paths: Updating servers and you only want one server down at a time, scheduling problems, route optimisation.

Types of Walks on Graphs – Summary

For a walk $v_0,\ e_1,\ v_1,\ e_2,\ \ldots,\ e_n,\ v_n$ on a graph G:

	Properties				
Name:	**closed**	—	**Euler**	**simple**	**Hamilton**
Description:	—	no repeated edges	uses all edges	no repeated vertices	uses all vertices
Requirement:	$v_0 = v_n$	$i \neq j \implies e_i \neq e_j$	$\forall e \in E(G)\ \exists i\ e_i = e$	$i \neq j \implies v_i \neq v_j$ ($v_0 = v_n$ OK)	$\forall v \in V(G)\ \exists i\ v_i = v$
path		✓			
simple path		✓		✓	
Euler path		✓	✓		
Hamilton path		✓		✓	✓
closed walk	✓				
circuit	✓	✓			
simple circuit	✓	✓		✓	
Euler circuit	✓	✓	✓		
Hamilton circuit	✓	✓		✓	✓

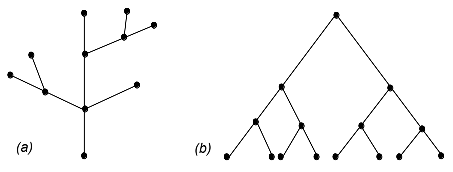
Trees

Trees

- A **tree** is a connected graph with no circuits other than the trivial ones.

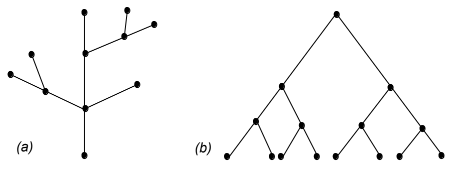
Trees

- A **tree** is a connected graph with no circuits other than the trivial ones. Examples:

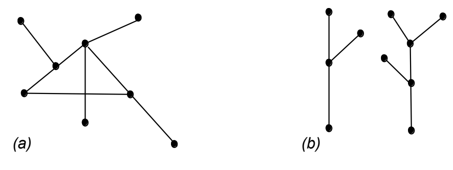


Trees

- A **tree** is a connected graph with no circuits other than the trivial ones. Examples:

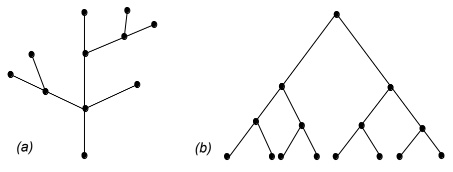


- Examples of **non-trees**

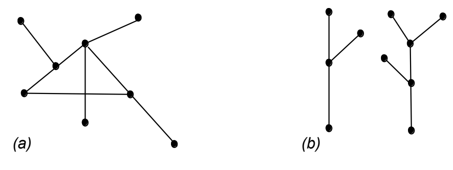


Trees

- A **tree** is a connected graph with no circuits other than the trivial ones. Examples:



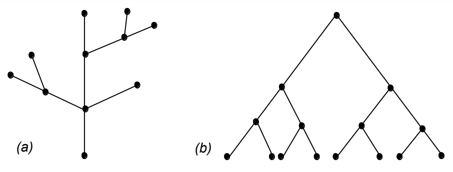
- Examples of **non-trees**



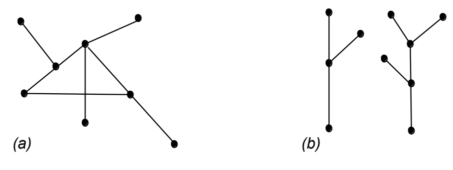
  - Example (a) contains a circuit, so is not a tree.

Trees

- A **tree** is a connected graph with no circuits other than the trivial ones. Examples:



- Examples of **non-trees**



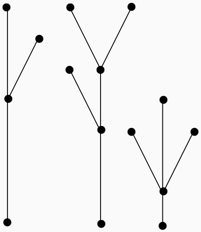
  - Example (a) contains a circuit, so is not a tree.
  - Example (b) is not a tree. *Why not?*

Forests and leaves

- A **forest** is a disjoint collection of trees.

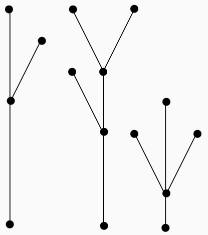
Forests and leaves

- A **forest** is a disjoint collection of trees. Example:



Forests and leaves

- A **forest** is a disjoint collection of trees. Example:



A **leaf** of a tree or forest is a vertex v with $\deg(v) = 1$.

Trees as Models

- Trees are used as models in such diverse areas as computer science, chemistry, geology, botany, psychology, management, linguistics and many others. (See Epp,10.5[4ed], 10.4[5ed]).

Trees as Models

- Trees are used as models in such diverse areas as computer science, chemistry, geology, botany, psychology, management, linguistics and many others. (See Epp,10.5[4ed], 10.4[5ed]).
- On your computer, the folder and file directory has a tree structure.

Trees as Models

- Trees are used as models in such diverse areas as computer science, chemistry, geology, botany, psychology, management, linguistics and many others. (See Epp,10.5[4ed], 10.4[5ed]).
- On your computer, the folder and file directory has a tree structure.
- Trees are used in linguistics to describe grammatical relationships.

Trees as Models

- Trees are used as models in such diverse areas as computer science, chemistry, geology, botany, psychology, management, linguistics and many others. (See Epp,10.5[4ed], 10.4[5ed]).
- On your computer, the folder and file directory has a tree structure.
- Trees are used in linguistics to describe grammatical relationships.
- In botany and zoology, taxonomy of species is entirely based on a tree structure.

Trees as Models

- Trees are used as models in such diverse areas as computer science, chemistry, geology, botany, psychology, management, linguistics and many others. (See Epp,10.5[4ed], 10.4[5ed]).
- On your computer, the folder and file directory has a tree structure.
- Trees are used in linguistics to describe grammatical relationships.
- In botany and zoology, taxonomy of species is entirely based on a tree structure.
- In evolutionary science we have 'the tree of life'.

Trees as Models

- Trees are used as models in such diverse areas as computer science, chemistry, geology, botany, psychology, management, linguistics and many others. (See Epp,10.5[4ed], 10.4[5ed]).
- On your computer, the folder and file directory has a tree structure.
- Trees are used in linguistics to describe grammatical relationships.
- In botany and zoology, taxonomy of species is entirely based on a tree structure.
- In evolutionary science we have 'the tree of life'.
- Organic chemistry uses graphs and trees for models of molecules.

Trees as Models

- Trees are used as models in such diverse areas as computer science, chemistry, geology, botany, psychology, management, linguistics and many others. (See Epp,10.5[4ed], 10.4[5ed]).
- On your computer, the folder and file directory has a tree structure.
- Trees are used in linguistics to describe grammatical relationships.
- In botany and zoology, taxonomy of species is entirely based on a tree structure.
- In evolutionary science we have ‘the tree of life’.
- Organic chemistry uses graphs and trees for models of molecules. We look at this next.

Structure of Hydrocarbon Molecules

- Graphs can be used to represent molecules, where atoms are represented by vertices and the bonds between them by edges.

Structure of Hydrocarbon Molecules

- Graphs can be used to represent molecules, where atoms are represented by vertices and the bonds between them by edges.
- **Hydrocarbon** molecules are composed of carbon and hydrogen only.

Structure of Hydrocarbon Molecules

- Graphs can be used to represent molecules, where atoms are represented by vertices and the bonds between them by edges.

- **Hydrocarbon** molecules are composed of carbon and hydrogen only.
- Each carbon atom normally forms 4 bonds with other atoms; we shall only consider this case.

Structure of Hydrocarbon Molecules

- Graphs can be used to represent molecules, where atoms are represented by vertices and the bonds between them by edges.

- **Hydrocarbon** molecules are composed of carbon and hydrogen only.
- Each carbon atom normally forms 4 bonds with other atoms; we shall only consider this case.
- Each hydrogen atom forms just one bond with another atom.

Structure of Hydrocarbon Molecules

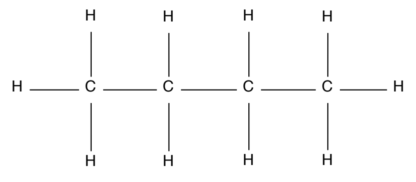
- Graphs can be used to represent molecules, where atoms are represented by vertices and the bonds between them by edges.

- **Hydrocarbon** molecules are composed of carbon and hydrogen only.
- Each carbon atom normally forms 4 bonds with other atoms; we shall only consider this case.
- Each hydrogen atom forms just one bond with another atom.

- In the representation of such molecules we use C to represent a carbon atom and H to represent a hydrogen atom. These will be used instead of dots for the vertices.

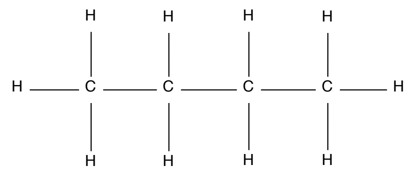
Examples of Hydrocarbon Graphs

- Butane:

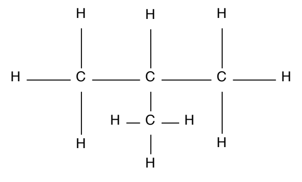


Examples of Hydrocarbon Graphs

- Butane:



- Isobutane:



Graphs and Isomers

- Butane and Isobutane graphs each have 4 carbon atoms and 10 hydrogen atoms, but the configuration of the atoms is different.

Graphs and Isomers

- Butane and Isobutane graphs each have 4 carbon atoms and 10 hydrogen atoms, but the configuration of the atoms is different.
- They have the same chemical formula, C_4H_{10}, but different chemical bonds and are called **isomers**.

Graphs and Isomers

- Butane and Isobutane graphs each have 4 carbon atoms and 10 hydrogen atoms, but the configuration of the atoms is different.
- They have the same chemical formula, C_4H_{10}, but different chemical bonds and are called **isomers**.
- **Saturated hydrocarbon** molecules contain the maximum number of hydrogen atoms for a given number of carbon atoms.

Some History of Trees

Arthur Cayley 1821 - 1895

- The English mathematician Arthur Cayley discovered trees when he was trying to enumerate the isomers of the saturated hydrocarbons of the form C_nH_{2n+2}.

Some History of Trees

Arthur Cayley 1821 - 1895

- The English mathematician Arthur Cayley discovered trees when he was trying to enumerate the isomers of the saturated hydrocarbons of the form C_nH_{2n+2}.
- Cayley showed that a saturated hydrocarbon molecule must have this formula.

Some History of Trees

Arthur Cayley 1821 - 1895

- The English mathematician Arthur Cayley discovered trees when he was trying to enumerate the isomers of the saturated hydrocarbons of the form C_nH_{2n+2}.
- Cayley showed that a saturated hydrocarbon molecule must have this formula.
- You will explore a proof of this formula in a Workshop in the coming weeks.

Characterising Trees

Theorem: Let T be a graph with n vertices. The following statements are logically equivalent:

Characterising Trees

Theorem: Let T be a graph with n vertices. The following statements are logically equivalent:

(i) T is a tree (it is connected and has no non-trivial circuits).

Characterising Trees

Theorem: Let T be a graph with n vertices. The following statements are logically equivalent:

(i) T is a tree (it is connected and has no non-trivial circuits).

(ii) T has no simple circuits and $n-1$ edges.

Characterising Trees

Theorem: Let T be a graph with n vertices. The following statements are logically equivalent:

(i) T is a tree (it is connected and has no non-trivial circuits).

(ii) T has no simple circuits and $n - 1$ edges.

(iii) T is connected and has $n - 1$ edges.

Characterising Trees

Theorem: Let T be a graph with n vertices. The following statements are logically equivalent:

(i) T is a tree (it is connected and has no non-trivial circuits).

(ii) T has no simple circuits and $n - 1$ edges.

(iii) T is connected and has $n - 1$ edges.

(iv) T is connected and every edge is a bridge.

Characterising Trees

Theorem: Let T be a graph with n vertices. The following statements are logically equivalent:

(i) T is a tree (it is connected and has no non-trivial circuits).

(ii) T has no simple circuits and $n-1$ edges.

(iii) T is connected and has $n-1$ edges.

(iv) T is connected and every edge is a bridge.

(v) Any two vertices of T are connected by exactly one simple path.

Characterising Trees

Theorem: Let T be a graph with n vertices. The following statements are logically equivalent:

(i) T is a tree (it is connected and has no non-trivial circuits).

(ii) T has no simple circuits and $n - 1$ edges.

(iii) T is connected and has $n - 1$ edges.

(iv) T is connected and every edge is a bridge.

(v) Any two vertices of T are connected by exactly one simple path.

(vi) T contains no non-trivial circuits, but the addition of any new edge (connecting an existing pair of vertices) creates a simple circuit.

Proving and Using the Theorem

- The previous Theorem collects together several different ways of characterizing a tree. In different contexts, each of the different descriptions are useful.

Proving and Using the Theorem

- The previous Theorem collects together several different ways of characterizing a tree. In different contexts, each of the different descriptions are useful.
- To prove the Theorem, we should show that any statement in the list is derivable from any other statement. One way to do this is to show the chain of implications:

$$(i) \Rightarrow (ii) \Rightarrow (iii) \Rightarrow (iv) \Rightarrow (v) \Rightarrow (vi) \Rightarrow (i).$$

- *Can you prove that* $(i) \Rightarrow (j)$ *for any of the statements in the Tree-Characterising Theorem?*

Proving the Theorem

- *Can you prove that $(i) \Rightarrow (j)$ for any of the statements in the Tree-Characterising Theorem?*
- Some results that you might find helpful are ...

  Lemma A: Any tree that has more than one vertex has at least one vertex of degree 1.

Proving the Theorem

- *Can you prove that* $(i) \Rightarrow (j)$ *for any of the statements in the Tree-Characterising Theorem?*
- Some results that you might find helpful are ...

  Lemma A: Any tree that has more than one vertex has at least one vertex of degree 1.

  Lemma B: If G is any connected graph, C is a non-trivial circuit in G, and one of the edges of C is removed, then the subgraph that remains is connected.

Applications of graphs, paths and trees

- Many problems can be modelled with paths formed by travelling along the edges of graphs.

Applications of graphs, paths and trees

- Many problems can be modelled with paths formed by travelling along the edges of graphs.
- For instance, the problem of determining whether a message can be sent between two computers using intermediate links can be studied with a graph model.

Applications of graphs, paths and trees

- Many problems can be modelled with paths formed by travelling along the edges of graphs.
- For instance, the problem of determining whether a message can be sent between two computers using intermediate links can be studied with a graph model.
- Problems of efficiently planning routes for mail delivery, garbage pickup, diagnostics in computer networks, and so on, can be solved using models that involve graphs.

Applications of graphs, paths and trees

- Many problems can be modelled with paths formed by travelling along the edges of graphs.
- For instance, the problem of determining whether a message can be sent between two computers using intermediate links can be studied with a graph model.
- Problems of efficiently planning routes for mail delivery, garbage pickup, diagnostics in computer networks, and so on, can be solved using models that involve graphs.
- A subgraph of a connected graph that provides a unique path between any two vertices is a *spanning tree*. These trees have applications in many fields, including engineering.

Spanning trees

Spanning trees

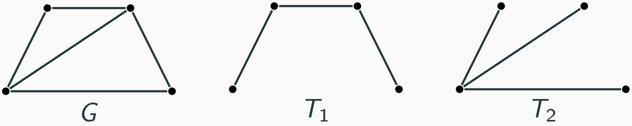
- **Definition:** A **spanning tree** for a graph G is a subgraph of G which is a tree and contains all the vertices of G.

Spanning trees

- **Definition:** A **spanning tree** for a graph G is a subgraph of G which is a tree and contains all the vertices of G.
- Spanning trees need not be unique. That is, a given graph can have a number of different spanning trees.

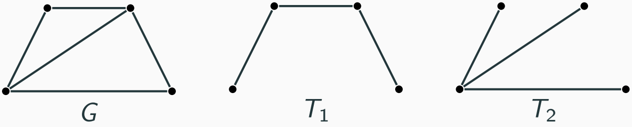
Spanning trees

- **Definition:** A **spanning tree** for a graph G is a subgraph of G which is a tree and contains all the vertices of G.
- Spanning trees need not be unique. That is, a given graph can have a number of different spanning trees.
- Example:Two distinct spanning trees, T_1 and T_2, for a graph G are shown below:



Spanning trees

- **Definition:** A **spanning tree** for a graph G is a subgraph of G which is a tree and contains all the vertices of G.
- Spanning trees need not be unique. That is, a given graph can have a number of different spanning trees.
- Example:Two distinct spanning trees, T_1 and T_2, for a graph G are shown below:



How many more spanning trees can you find for G?

Which graphs have spanning trees?

Theorem:

1. Every connected graph has a spanning tree.

Which graphs have spanning trees?

Theorem:

1. Every connected graph has a spanning tree.

   Can you prove this?

Which graphs have spanning trees?

Theorem:

1. Every connected graph has a spanning tree.

   Can you prove this?

   Hint: Use Lemma B.

Which graphs have spanning trees?

Theorem:

1. Every connected graph has a spanning tree.

   Can you prove this?

   Hint: Use Lemma B.

2. Any 2 spanning trees for a graph have the same number of edges.

Which graphs have spanning trees?

Theorem:

1. Every connected graph has a spanning tree.

   Can you prove this?

   Hint: Use Lemma B.

2. Any 2 spanning trees for a graph have the same number of edges.

   Why?

Which graphs have spanning trees?

Theorem:

1. Every connected graph has a spanning tree.

   Can you prove this?

   Hint: Use Lemma B.

2. Any 2 spanning trees for a graph have the same number of edges.

   Why?

   Hint: See (ii) or (iii) of the tree characterisation theorem.

A method to make a spanning tree

Let G be a connected graph with n vertices.

A method to make a spanning tree

Let G be a connected graph with n vertices.

The Algorithm

1. Initialize T to be the vertices of G but no edges.

A method to make a spanning tree

Let G be a connected graph with n vertices.

The Algorithm

1. Initialize T to be the vertices of G but no edges.
2. Pick an edge.

A method to make a spanning tree

Let G be a connected graph with n vertices.

The Algorithm

1. Initialize T to be the vertices of G but no edges.
2. Pick an edge.
3. Add the chosen edge to T if and only if it does not make a circuit in T.

A method to make a spanning tree

Let G be a connected graph with n vertices.

The Algorithm

1. Initialize T to be the vertices of G but no edges.
2. Pick an edge.
3. Add the chosen edge to T if and only if it does not make a circuit in T.
4. Repeat steps (2) and (3) until T has $n - 1$ edges, then stop.

A method to make a spanning tree

Let G be a connected graph with n vertices.

The Algorithm

1. Initialize T to be the vertices of G but no edges.
2. Pick an edge.
3. Add the chosen edge to T if and only if it does not make a circuit in T.
4. Repeat steps (2) and (3) until T has $n - 1$ edges, then stop.

The above description of the algorithm is intended for 'pen-and-paper' work.

A method to make a spanning tree

Let G be a connected graph with n vertices.

The Algorithm

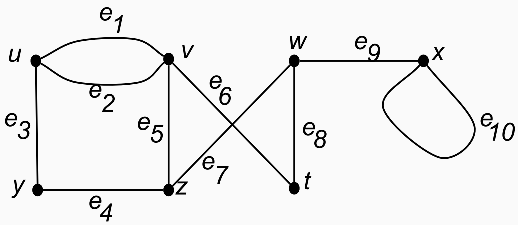
1. Initialize T to be the vertices of G but no edges.
2. Pick an edge.
3. Add the chosen edge to T if and only if it does not make a circuit in T.
4. Repeat steps (2) and (3) until T has $n-1$ edges, then stop.

The above description of the algorithm is intended for 'pen-and-paper' work.

For computer implementation it is necessary to **also**:
— at step 1 initialize a 'pool of potential edges' P to $E(G)$,
— at step 2 ensure the picked edge e comes from P and
— after step 3 remove e from P (whether it contributes to T or not).

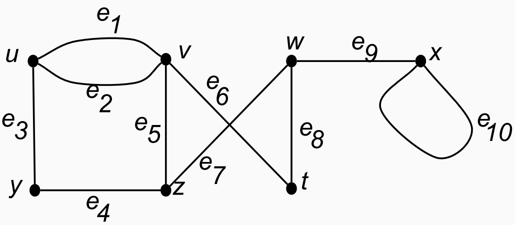
Building a spanning tree: example

To demonstrate the spanning tree algorithm we will use a graph we have seen before:



Building a spanning tree: example

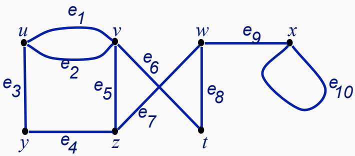
To demonstrate the spanning tree algorithm we will use a graph we have seen before:



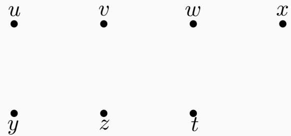
Initialize T to

- vertex set $V(T) = \{u, v, w, x, y, z, t\}$,
- edge set $E(T) = \{\}$:

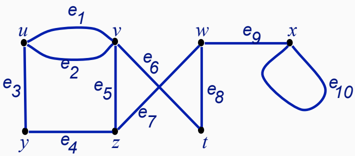
Building a spanning tree for



$E(T) = \{\}$: T :

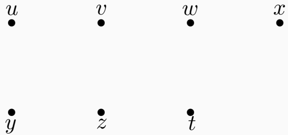


Building a spanning tree for



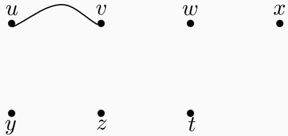
$E(T) = \{\}$:

T :

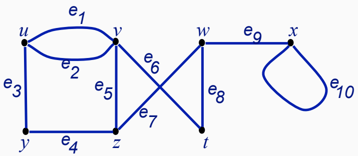


$E(T) = \{e_1\}$

T :

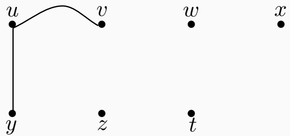


Building a spanning tree for

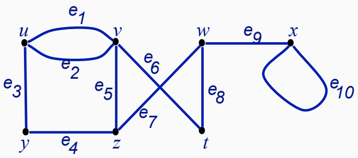


$E(T) = \{e_1, e_3\}$:

T :

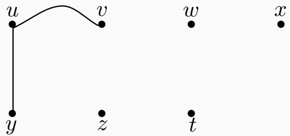


Building a spanning tree for



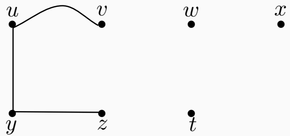
$E(T) = \{e_1, e_3\}$:

T :

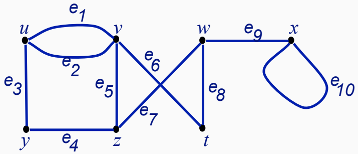


$E(T) = \{e_1, e_3, e_4\}$

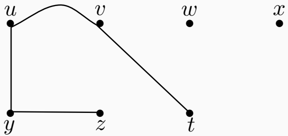
T :



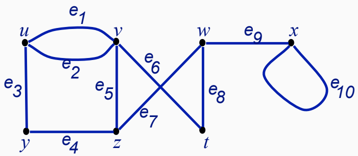
Building a spanning tree for



$E(T) = \{e_1, e_3, e_4, e_6\}$: T :

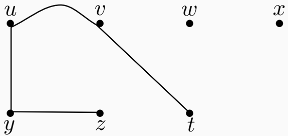


Building a spanning tree for



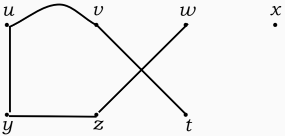
$E(T) = \{e_1, e_3, e_4, e_6\}$:

T :

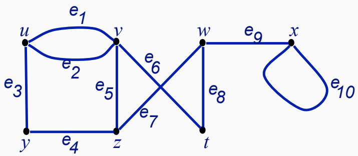


$E(T) = \{e_1, e_3, e_4, e_6, e_7\}$

T :

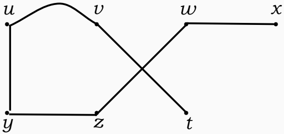


Completed spanning tree for

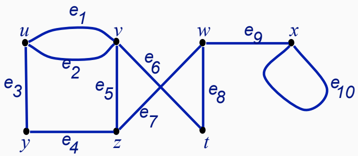


$E(T) = \{e_1, e_3, e_4, e_6, e_7, e_9\}$:

T :

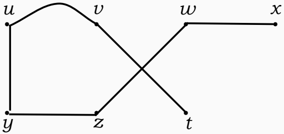


Completed spanning tree for



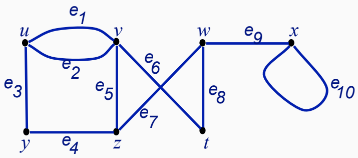
$E(T) = \{e_1, e_3, e_4, e_6, e_7, e_9\}$:

T :

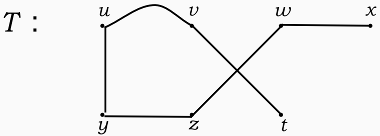


0ur tree now has $7 - 1 = 6$ edges, so we are done (as you can see).

Completed spanning tree for



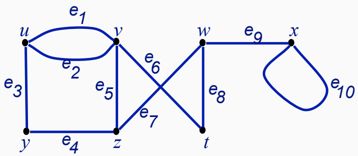
$E(T) = \{e_1, e_3, e_4, e_6, e_7, e_9\}$:



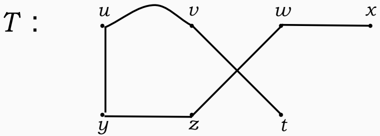
0ur tree now has $7 - 1 = 6$ edges, so we are done (as you can see).

As it happens, this tree has no branching and so contains a path of length 6. That just results from our order of choosing edges.

Completed spanning tree for



$E(T) = \{e_1, e_3, e_4, e_6, e_7, e_9\}$:

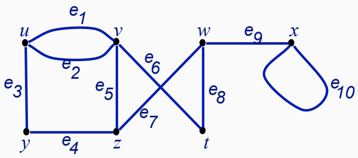


0ur tree now has $7 - 1 = 6$ edges, so we are done (as you can see).

As it happens, this tree has no branching and so contains a path of length 6. That just results from our order of choosing edges.

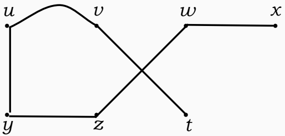
Can you make a spanning tree in which the longest path has length 4?

Completed spanning tree for



$E(T) = \{e_1, e_3, e_4, e_6, e_7, e_9\}$:

T :

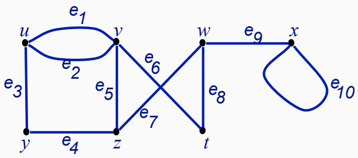


0ur tree now has $7 - 1 = 6$ edges, so we are done (as you can see).

As it happens, this tree has no branching and so contains a path of length 6. That just results from our order of choosing edges.

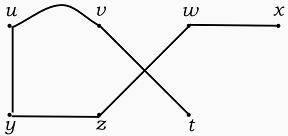
Can you make a spanning tree in which the longest path has length 4? Length 3?

Completed spanning tree for



$E(T) = \{e_1, e_3, e_4, e_6, e_7, e_9\}$:

T :



0ur tree now has $7 - 1 = 6$ edges, so we are done (as you can see).

As it happens, this tree has no branching and so contains a path of length 6. That just results from our order of choosing edges.

Can you make a spanning tree in which the longest path has length 4? Length 3?

END OF SECTION D1