

**Key Terms**

Incident

Adjacent/Neighbors

Degree of a vertex

Path, walk, cycle, tour

Eulerian tour

Tree

Hypercube

**I. Graph Theory**

Let  $G = (V, E)$  be an undirected graph. Match the term with the definition.

Word Bank.

Walk

Cycle

Tour

Path

tour

Walk that starts and ends at the same node

path

Sequence of edges.

walk

Sequences of edges with possibly repeated vertex or edge.

cycle

Sequence of edges that starts and ends on the same vertex and does not repeat vertices (except the first and last)

What is a simple path?

Sequence of edges where the vertices are distinct

### Exercises

1. Given a graph  $G$  with  $n$  vertices, where  $n$  is even, prove by induction that if every vertex has degree  $n/2+1$ , then  $G$  must contain a 3-cycle.

Let  $G$  be a graph with  $n$  vertices, where  $n$  is even, and every vertex has degree  $\frac{n}{2} + 1$ . Select any two vertices  $u$  and  $v$ , with an edge between them. There are  $n - 2$  remaining vertices, and both  $u$  and  $v$  are connected to  $\frac{n}{2}$  of these (because they have degree  $\frac{n}{2} + 1$  and are connected to each other). Therefore, there must be some vertex  $w$  such that both  $u$  and  $v$  are connected to  $w$  (otherwise, the set of  $\frac{n}{2}$  vertices connected to  $u$  and the set of  $\frac{n}{2}$  vertices connected to  $v$  would be disjoint, which contradicts the fact that there are only  $n - 2$  of these vertices). Thus we have edges  $(u, v)$ ,  $(v, w)$ , and  $(w, u)$ , so the graph contains a 3-cycle.

2. Every tournament has a Hamiltonian path.

Base Case: For  $n = 1$  nodes, there is a trivial Hamiltonian path.

Inductive Hypothesis: Assume that for a tournament with  $n$  nodes, there is a Hamiltonian path.

Inductive Step: Consider a tournament  $T$  with  $n + 1$  nodes. Take an arbitrary node  $x$ , and remove it along with its incident edges. The resulting subgraph  $T'$  is also a tournament (each node in  $T'$  still shares some edge with every other node in  $T'$ ). By the Inductive Hypothesis, there is some Hamiltonian path in  $T'$ . Let this Hamiltonian Path be  $v_1, v_2, v_3, \dots, v_n$ . Now we consider  $T$ . Note that since  $T$  is a tournament,  $x$  shares an edge with every other node in  $T$ . There are three possible cases:

Case 1: Everybody beat  $x$  (there is no edge from  $x$  to any node in  $T'$ ). Then there is an edge  $(v_n, x)$ . Thus, there is a Hamiltonian Path in  $T$ , namely  $v_1, v_2, v_3, \dots, v_n, x$ .

Case 2:  $x$  beat everybody (there is no edge from any node in  $T'$  to  $x$ ). Then there is an edge  $(x, v_1)$ . Thus, there is a Hamiltonian Path in  $T$ , namely  $x, v_1, v_2, v_3, \dots, v_n$ .

Case 3: There is some  $v_i$  that is the last person who beat  $x$ , in the ordering  $v_1, \dots, v_n$ . Note that  $v_i$  must exist because we are not in Case 2, and  $i \neq n$  because we are not in Case 1. Then since  $v_i$  is the last person who beat  $x$ , there is an edge  $(v_i, x)$ , and an edge  $(x, v_{i+1})$ . Thus, there is a Hamiltonian path in  $T$ , namely  $v_1, v_2, v_3, \dots, v_i, x, v_{i+1}, \dots, v_n$ .

These are the only possible cases, so it must be that  $T$  has a Hamiltonian Path.

Therefore by induction, any tournament has a Hamiltonian Path.

## II. Eulerian Tour (SECTION 2 REMOVED FROM WORKSHEET)

What is an Eulerian tour?

A walk that uses each edge exactly once and starts and ends at the same vertex

Euler's Theorem: An undirected graph  $G=(V, E)$  has an Eulerian tour if and only if  $G$  is even degree and connected (except possibly for isolated vertices).

$\Rightarrow$  : Write the "only if" portion of Euler's Theorem:

If  $G$  has an Eulerian tour then  $G$  is even degree and connected

Assume  $G$  has an Eulerian tour.

Part 1:  $G$  is connected

Every vertex that is not isolated must be on the tour

This implies that every vertex is connected with all other vertices that are on the tour.

Therefore  $G$  is connected ■

Part 2:  $G$  has even degree. (hint: **pair** up all edges)

Every time tour enters a vertex along an edge it must exit along a different edge

So, we can pair these edges up.

What vertex can we not do this for? the starting vertex

A tour must start and end at the same vertex

Pair the first edge with the last edge at the start vertex.

All edges adjacent to any vertex of the tour can be paired up.

Therefore each vertex has an even degree ■

⇐: Write the “if” portion of Euler’s Theorem:

If  $G$  is connected and has even degree, then  $G$  has an Eulerian tour.

Algorithms	
<b>FindTour(<math>G, s</math>) :</b> Finds a tour in $G$ . It starts at vertex $s$ and keeps going until it gets stuck at $s$	<b>Euler (<math>G, s</math>):</b> $T = \text{FindTour}(G, s)$ Let $G_1, \dots, G_k$ be the connected components when edges in $T$ are removed from $G$ . Let $s_i$ be the first vertex in $T$ that intersects $G_i$ Output $\text{Splice}(T, \text{Euler}(G_1, s_1), \dots, \text{Euler}(G_k, s_k))$
<b>Splice(<math>T, T_1, \dots, T_k</math>):</b> Input: $T, T_1, \dots, T_k$ are tours. $T$ intersects each $T_1, \dots, T_k$ Output: Tour that traverses all edges in the input	

**Claim:** **FINDTOUR**( $G, s$ ) must get stuck at  $s$ .

An easy proof by induction on the length of the walk shows that when **FINDTOUR** enters any vertex  $v \neq s$ , it will have traversed an odd / even number of edges incident to  $v$ .

When it enters  $s$  it will have traversed an odd / even number of edges incident to  $s$ .

Since every vertex in  $G$  has even degree, this means every time it enters  $v \neq s$ , there is/are untraversed edge(s) incident to  $v$ , and therefore the walk cannot get stuck.

So the only vertex it can get stuck at is  $s$ .

**Claim:** **EULER**(G,s) outputs an Eulerian Tour in G.

Proof: by induction on the size of G

*Base Case:*  $m=0$

1. The tour is

---

*Inductive Hypothesis:* **Euler**(G, s) outputs an Eulerian Tour in G for any even degree, connected graph with at most  $m \geq 0$  edges.

*Inductive Step:*

2. Suppose G has \_\_\_\_\_ edges

3. Recall that  $T = \mathbf{FINDTOUR}(G,s)$  is a tour, and therefore has \_\_\_\_\_ degree at every vertex

4. Remove edges of T from G

5. Left with even degree graph with \_\_\_\_\_ edges, but it might be disconnected.

6. Let  $G_1, G_2, \dots, G_k$  be the connected components

7. Each  $G_1, \dots, G_k$  is \_\_\_\_\_ and has \_\_\_\_\_ degree.

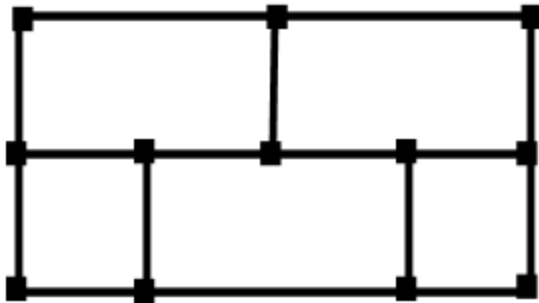
8. T intersects each  $G_i$ . Call the vertex that T shares with each  $G_i$ ,  $s_i$ .

9. By the induction hypothesis, **EULER**( $G_i, s_i$ ) outputs an Eulerian tour of  $G_i$ .

10. Splice can then \_\_\_\_\_

11. Therefore we produced an Eulerian tour.

Fun Exercise (REMOVED FROM WORKSHEET)



Is it possible to draw a continuous curve that crosses each line once?  
(each segment between the dots is considered a distinct line)

No, draw a point in the middle of each box and connect to an edge → becomes Euler problem

### III. Trees

If complete graphs are “maximally connected,” then trees are the opposite: Removing just a single edge disconnects the graph! Formally, there are a number of equivalent definitions of when a graph  $G = (V, E)$  is a tree, including:

What are four ways to describe trees?

- 1)  $G$  is **connected** and **contains no cycles**
- 2)  $G$  is **connected** and has  **$n - 1$  edges**
- 3)  $G$  is **connected** and **removing any edge disconnects  $G$**
- 4)  $G$  has no cycles and **addition of any edge creates a cycle**

Theorem:  $G$  is connected and contains no cycles if and only if  $G$  is connected and has  $n - 1$  edges.

1. We saw in the notes on page 8 that 1 and 2 above were saying the same thing- that is, stated rigorously,  $1 \Leftrightarrow 2$ . We will now prove that  $1 \Leftrightarrow 3$ :

First we prove the forward direction ( $1 \Rightarrow 3$ ). If  $G$  is connected and has no cycles, obviously the first part of 3 (connectedness) is satisfied. We now also prove that, in this case, the removal of any edge from  $G$  must also disconnect it. Consider the possibility that removing an edge from  $G$  did not disconnect it (**point out that you are doing a proof by contradiction!**). Consider the nodes on the endpoints of the removed edge, denoted  $u$  and  $v$ . If there is still some path  $P$  from  $u$  to  $v$  in  $G$ , then that path, when augmented with the removed edge, would form a cycle since we could start at  $u$ , use  $P$  to arrive at  $v$ , and finally use the removed edge to arrive back at  $u$  (from  $v$ ). But this is a cycle, which is a contradiction! Hence, we are done with the forward direction.

Now we prove the reverse direction: that is, that  $3 \Rightarrow 1$ . We are *now* given that  $G$  is connected and that removing *any* edge will disconnect it. We now wish to prove that this implies 1. Obviously the first part of 1 (connectedness) is satisfied. Consider the possibility that  $G$  *did* contain a cycle (**again, point out that you are doing a proof by contradiction!**). If this were the case, we could remove an edge in that cycle, and  $G$  would still be connected. Why? Consider any node on the cycle. If we remove one of the edges from the cycle, then it means that the other nodes on that cycle are still reachable from one another (**try showing this pictorially**). But this contradicts the given that removing *any* edge disconnects  $G$ ! Now we have proven the reverse direction, and we are done.



## **DO NOT GO OVER -- HOMEWORK PROBLEM**

Recall that a tree is a connected acyclic graph. Note that this is similar to but slightly different from a rooted tree that you might have come across before in the context of data structures. In our context the tree does not have any designated root vertex, and is just a graph on  $n$  vertices and with  $m$  edges.

### Exercises

1. We start by proving two useful properties of trees:

(a) Any pair of vertices in a tree are connected by exactly one (simple) path.

Pick any pair of vertices  $x, y$ . We know there is a path between them since the graph is connected. We will prove that this path is unique by contradiction:

Suppose there are two distinct paths from  $x$  to  $y$ . At some point (say at vertex  $a$ ) the paths must diverge, and at some point (say at vertex  $b$ ) they must reconnect. So by following the first path from  $a$  to  $b$  and the second path in reverse from  $b$  to  $a$  we get a cycle. This gives the necessary contradiction.

(b) Adding any edge to a tree creates a simple cycle.

Pick any pair of vertices  $x, y$  not connected by an edge. We prove that adding the edge  $\{x, y\}$  will create a simple cycle. From part (a), we know that there is a unique path between  $x$  and  $y$ . Therefore, adding the edge  $\{x, y\}$  creates a simple cycle obtained by following the path from  $x$  to  $y$ , then following the edge  $\{x, y\}$  from  $y$  back to  $x$ .

Now show that if a graph satisfies either of these two properties then it must be a tree:

(c) If for every pair of vertices in a graph they are connected by exactly one simple path, then the graph must be a tree.

Assume we have a graph with the property that there is a unique simple path between every pair of vertices. We will show that the graph is a tree, namely, it is connected and acyclic. First, the graph is connected because every pair of vertices is connected by a path. Moreover, the graph is acyclic because there is a unique path between every pair of vertices. More explicitly, if the graph has a cycle, then for any two vertices  $x, y$  in the cycle there are at least two simple paths between them (obtained by going from  $x$  to  $y$  through the right or left half of the cycle), contradicting the uniqueness of the path.

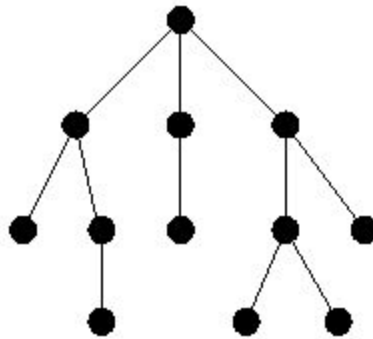
Therefore, we conclude the graph is a tree.

(d) If the graph has no simple cycles but has the property that the addition of any single edge (not already in the graph) will create a simple cycle, then the graph is a tree.

Assume we have a graph with no simple cycles, but adding any edge will create a simple cycle. We will show that the graph is a tree. We know the graph is acyclic because it has no simple cycles. To show the graph is connected, we prove that any pair of vertices  $x, y$  are connected by a path. We consider two cases: If  $\{x, y\}$  is an edge, then clearly there is a path from  $x$  to  $y$ . Otherwise, if  $\{x, y\}$  is not an edge, then by assumption, adding the edge  $\{x, y\}$  will create a simple cycle. This means there is a simple path from  $x$  to  $y$  obtained by removing the edge  $\{x, y\}$  from this cycle. Therefore, we conclude the graph is a tree.

Recall from the notes that a **rooted tree** is a tree with a particular node designated as the root, and the other nodes arranged in levels, “growing down” from the root. An alternative, recursive, definition of rooted tree is the following:

A rooted tree consists of a single node, the root, together with zero or more “branches,” each of which is itself a rooted tree. The root of the larger tree is connected to the root of each branch



*A rooted tree.*

2. Prove that given any tree, selecting any node to be the root produces a rooted tree according to the definition above.




Use induction! (on number of vertices)

Base case: one-vertex tree; have to select that to be the root; trivially a rooted tree (with zero branches)

Inductive hypothesis: tree with  $k$  or fewer vertices can be made into rooted tree by selecting any vertex as root

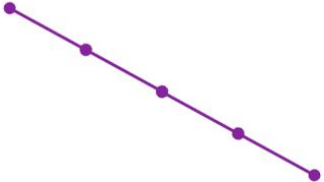

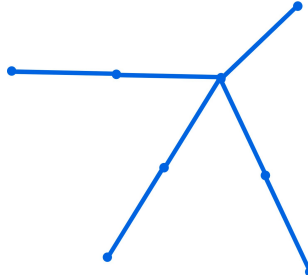
Inductive step: given a tree with  $k + 1$  vertices, let an arbitrary vertex  $v$  be selected as the root. This vertex  $v$  has, let us say,  $m$  neighbors. Disconnecting each neighbor from  $v$  would produce  $m$  subtrees (which must be disjoint, or else we would have a cycle). By the inductive hypothesis, because each of these has at most  $k$  vertices, they each form a rooted tree when we select  $v$ 's neighbor as the root. Overall, then, we have a root node,  $v$ , connected to a number of disjoint rooted trees, which are its branches.

3. Now we consider some example trees: **(REMOVED FROM WORKSHEET)**


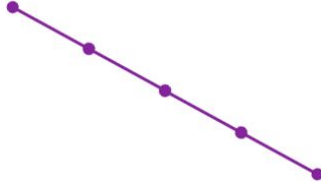
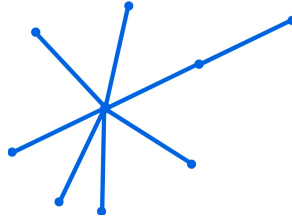
			
# Leaves	2	5	4
Max depth	4	2	8
Min depth	2	1	4

4. Draw trees with the following properties **(REMOVED FROM WORKSHEET)**

(a) Where the longest path between two vertices is:

		
$ V  - 1$	2	4 AND graph has 7 vertices

(b) Where the number of leaves is:

		
$ V  - 1$	2	7 AND graph has 9 vertices

A **spanning tree** of a graph  $G$  is a subgraph of  $G$  that contains all the vertices of  $G$  and is a tree.

5. Prove that a graph  $G = (V, E)$  is connected if and only if it contains a spanning tree.

First the “if” direction. If a graph contains a spanning tree, which is a connected graph that contains all the vertices, there is a path between any two vertices, so the graph is connected.

Now the “only if.” Let  $G$  be a connected graph. Either  $G$  is already a tree, in which case it is its own spanning tree, or else there is an edge that can be removed from  $G$  while it remains connected. Because there are only a finite number of edges, we can continue this process until no more edges can be removed, at which point we have found our spanning tree.

6. How many distinct spanning trees does  $K_3$  have? How many does  $K_4$  have?

$K_3$ : 3

$K_4$ : 16

#### IV. Hypercubes

What is an n dimensional hypercube?

The bit definition: Two vertices  $x$  and  $y$  are adjacent and only if  $x$  and  $y$  differ in exactly one bit position.

Recursive definition: Define the 0-subcube as the  $(n-1)$  dimensional hypercube with vertices labeled  $0x$  ( $x$  is an element of  $\{0, 1\}^{n-1}$  (hint: how many remaining bits are there?)). Do the same for the 1-subcube with vertices labeled  $1x$ . Then an  $n$  dimensional hypercube is created by placing an edge between  $0x$  and  $1x$  in the 0-subcube and 1-subcube respectively.

#### Exercises

1. How many vertices does an  $n$  dimensional hypercube have?

$$V_{x,y} : 2^n$$

2. How many edges does an  $n$  dimensional hypercube have?

$$E_{x,y} : n(2^{n-1})$$

3. How many edges do you need to cut from a hypercube to isolate one vertex in an  $n$ -dimensional hypercube?

$n$  because each node has  $n$  edges

**(REMOVED FROM WORKSHEET)**

A *Hamiltonian path* in an undirected graph  $G = (V, E)$  is a path that goes through every vertex exactly once. A *Hamiltonian cycle* (or *Hamiltonian tour*) is a cycle that goes through every vertex exactly once. Note that, in a graph with  $n$  vertices, a Hamiltonian path consists of  $n-1$  edges, and a Hamiltonian cycle consists of  $n$  edges.

4. *Hamiltonian path*

Prove that for every  $n \geq 2$ , the  $n$ -dimensional hypercube has a Hamiltonian cycle.

**Proof:** By induction on  $n$ .

In the base case,  $n = 2$ , the 2-dimensional hypercube, the cycle starts at 00, goes through 01, 11, and 10, and returns to 00.

Suppose now that every  $(n - 1)$ -dimensional hypercube has a Hamiltonian cycle. Let  $v \in \{0, 1\}^{n-1}$  be a vertex adjacent to  $0_{n-1}$  (the notation  $0_{n-1}$  means a sequence of  $n-1$  zeroes) in the Hamiltonian cycle in a  $(n-1)$ -dimensional hypercube. The following is a Hamiltonian cycle in an  $n$ -dimensional hypercube: have a path that goes from  $0_n$  to  $0_v$  by passing through all vertices of the form  $0x$  (this is simply a copy of the Hamiltonian path in dimension  $(n-1)$ , minus the edge from  $v$  to  $0_{n-1}$ ), then an edge from  $0_v$  to  $1_v$ , then a path from  $1_v$  to  $1_{0_{n-1}}$  that passes through all vertices of the form  $1x$ , and finally an edge from  $1_{0_{n-1}}$  to  $0_n$ . This completes the proof of the Theorem. \* When we start from  $0_n$  and we follow the Hamiltonian cycle described in the above proof, we find an ordering of all the  $n$ -bit binary strings such that each string in the sequence differs from the previous string in only one bit. Such an ordering is called a *Gray code* (from the name of the inventor) and it has various applications in error correction schemes and other fields. You can read about some of these in the wikipedia entry ([http://en.wikipedia.org/wiki/Gray\\_code](http://en.wikipedia.org/wiki/Gray_code)).

Proof by induction.

Base case  $n = 1$  is trivial.

Assume there exists a Hamiltonian cycle on a  $k$ -cube. To prove that a cycle for a  $k + 1$ -cube exists:, Note that a  $k + 1$ -cube is constructed by taking two copies of a  $k$ -cube and connecting the corresponding vertices. Take a Hamiltonian cycle for a  $k$ -cube and delete the last edge from the cycle. Call this path  $P$ , and say it starts at  $a$  and ends at  $b$ . Let the reverse of this path be  $P_0$ , starting at  $b$  and ending at  $a$ . Follow  $P$  on one copy of the  $k$ -cube and  $P_0$  on the other copy (going from  $b_0$  to  $a_0$ ). Since there is an edge between  $a$  and  $a_0$  and one between

en  $b$  and  $b_0$ , we can start at  $a$ , follow  $P$  to  $b$  through every point in cube 1, go to  $b_0$ , follow  $P_0$  to  $a_0$  through every point in cube 2, then back to  $a$ .

5. The hypercube is a popular architecture for parallel computation. Let each vertex of the hypercube represent a processor and each edge represent a communication link. Suppose we want to send a packet for vertex  $x$  to vertex  $y$ . Consider the following “bit-fixing” algorithm:

In each step, the current processor compares its address to the destination address of the packet. Let’s say that the two addresses match up to the first  $k$  positions. The processor then forwards the packet and the destination address on to its neighboring processor whose address matches the destination address in at least the first  $k+1$  positions. This process continues until the packet arrives at its destination.

Consider the following example where  $n = 4$ : Suppose that the source vertex is (1001) and the destination vertex is (0100). Give the sequence of processors that the packet is forwarded to using the bit-fixing algorithm.

1001  $\rightarrow$  0001  $\rightarrow$  0101  $\rightarrow$  0100

Emphasis on strings that differ by one bit are neighbors

### Bonus Exercises:

Let  $v$  be an odd degree node. Consider the longest walk starting at  $v$  that does not repeat any edges (though it may omit some). Let  $w$  be the final node of the walk. Show that  $v \neq w$ .

Proof by contradiction.

Assume that the longest walk does get stuck at  $v$ .

This means that at some point in the walk we entered vertex  $v$  and there were no more untraversed edges.

Now pair up the incoming and departing edges as in the notes.

Every time we entered vertex  $v$  before we got stuck, we also left it.

Pair up the first edge we traverse when we leave with the last edge we traverse when we got stuck.

This must imply that we used up an even amount of edges incident to  $v$ .

But  $v$  is of odd degree (assumption)

Therefore there must be at least one untraversed edge when we entered  $v$  the last time.

But now we have that we got stuck and there is an untraversed edge. Contradiction. We must not get stuck at  $v$ .

Prove that undirected connected graph with  $|V| \geq 2$ , 2 nodes have same degree

Pigeonhole principle. Suppose graph  $G$  has  $n$  vertices.

What is the minimum degree of any vertex? 1

What is the maximum degree of any vertex?  $n - 1$

So all degree values must be :  $\{1, \dots, n - 1\}$

But there are  $n$  vertices.



Prove that every undirected finite graph where every vertex has degree of at least 2 has a cycle.

Assume that it does not have a cycle.

So we have a graph  $G$  that is connected and does not have a cycle. It must be a tree.

Every tree has at least one leaf. Leaves have degree 1. But we assumed that every vertex has degree at least 2. Contradiction, there must be a cycle.

Prove that every undirected finite graph where every vertex has degree of at least 3 has a cycle of even length.

Let  $P$  be a maximal path and  $v$  be an endpoint of  $P$ .

$v$  has at least 3 neighbors that are on  $P$  (why?)

Let the path  $P$  be:  $\{ \dots x, y, z, v \}$

Now consider the following three paths:  $v \rightarrow y$ ,  $y \rightarrow x \rightarrow v$ , and  $y \rightarrow z \rightarrow v$

The union of two of these paths is a cycle.

At least two of these paths must have lengths that are both even or both odd. Adding either of those cases together creates an even length cycle.