GRAPHS - EXTRA PRACTICE 2

COMPUTER SCIENCE MENTORS 70

September 10 to September 14, 2018

1 Eulerian Tour

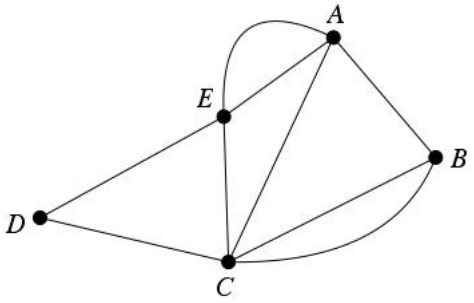
1.1 Introduction

An **Eulerian path** is a path that uses every edge exactly once.

An **Eulerian tour** is a path 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).

1.2 Questions



1. Is there an Eulerian Tour? If so, find one. Repeat for an Eulerian Path.

Solution: There is no Eulerian Tour in the graph, because not all vertices have an even degree. An Eulerian Tour must visit every edge and end up at the same vertex. So the number of times it leaves/enters the start vertex must be even (every time it leaves, it must come back). Now every other vetex, must have the same condition. Since our tour doesn't end at any of these vertices, every time the tour enters a vertex, it must leave that vertex. Therefore, every vertex must have an even degree for there to be an Eulerian Tour.
There is an Eulerian Path. An Eulerian Path is almost like an Eulerian Cycle, without the condition that the start and end vertices must be the same. Therefore there are two vertices where the path can leave, and not return or enter and not leave. So there can be 2 vertices of odd degree. This graph does have 2 vertices of odd degree.

$$B \to C \to D \to E \to A \to E \to C \to A \to B \to C$$

2. If every node has even degree except two nodes that have odd degree, prove that the graph has a Eulerian path.

Solution: First, add an edge from one odd degree vertex X to the other odd degree Y. This modified graph has an Eulerian Tour by Euler's Theorem. This tour contains something like $Z \to X \to Y \to \ldots \to Z$, where $\ldots \to Z$ contains every edge in the original graph. Deleting an edge leaves a Eulerian path in the graph.

1.3 Assorted Graph Questions

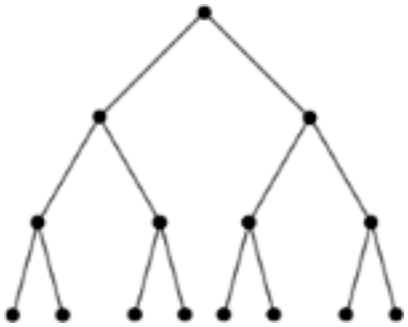
1. Prove that every undirected finite graph where every vertex has degree of at least 2 has a cycle.

Solution: 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.

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

Solution: 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: $v \to y$, $y \to x \to v$, and $y \to z \to v$ 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.

3. 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.



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

Solution: 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 k 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 vs neighbor as the root. Overall, then, we have a root node, v, connected to a number of disjoint rooted trees, which are its branches.

4. Show that the edges of a complete graph on n vertices for even n can be partitioned

into $\frac{n}{2}$ edge disjoint spanning trees.

Hint: Recall that a complete graph is an undirected graph with an edge between every pair of vertices. The complete graph has $\frac{n*(n-1)}{2}$ edges. A spanning tree is a tree on all n vertices – so it has $n-1$ edges. So the complete graph has enough edges (for even n) to create exactly $\frac{n}{2}$ edge disjoint spanning trees (i.e. each edge participates in exactly one spanning tree). You have to show that this is always possible.

Solution: We proceed by induction.

Base Case: Consider a complete graph on 2 vertices. This can clearly be partitioned into 2/2 = 1 edge disjoint spanning tree, because the graph is already a tree.

Inductive Hypothesis: Assume that the edges of a complete graph on k vertices (for k even) can be partitioned into $\frac{k}{2}$ edge disjoint spanning trees.

Inductive Step: We need to partition the edges of a complete graph G_{k+2} on $k+2$ vertices into $\frac{k}{2}+1$ edge disjoint spanning trees.

To do this, label the vertices of G_{k+2} as $v_1, v_2, \ldots, v_{k+2}$. Remove the vertices $v_k + 1$ and v_{k+2} (and associated edges) to form a complete graph G_k with k vertices $v_1, \ldots, v_k$. By the inductive hypothesis, G_k has $\frac{k}{2}$ edge disjoint spanning trees; call these trees $T_1, \ldots, T_k = 2$. Add the vertices v_{k+1} and v_{k+2} back into G_k to once again form the graph G_{k+2}. These vertices come with $2k+1$ extra edges, connecting (v_i, v_{k+1}) and (v_i, v_{k+2}) for each $i = 1, 2, \ldots, k$, and also (v_{k+1}, v_{k+2}). These edges must be included into spanning trees. We wish to extend the trees $T_1, \ldots, T_{k+2}$ to include the new vertices v_{k+1} and v_{k+2}. To do this, for each tree T_i, attach two new edges (v_i, v_{k+1}) and $(v_{i+\frac{k}{2}}, v_{k+2})$. This extends each tree T_i to be a spanning tree. The remaining edges form one additional spanning tree. These edges are $(v_{i+\frac{k}{2}}, v_{k+1})$ and (v_i, v_{k+2}) for $i = 1$ to $\frac{k}{2}$, along with the connecting edge (v_{k+1}, v_{k+2}). These edges connect each of the vertices v_{k+1} and v_{k+2} to half the remaining vertices, and together with the edge between v_{k+1} and v_{k+2} this gives the desired spanning tree. Therefore, we have covered the graph in $\frac{k}{2}+1$ edge disjoint spanning trees. This completes the induction.

Remark: The key idea here is the following:

Take a graph with k vertices that is partitioned into $\frac{k}{2}$ spanning trees. In the inductive step, we want to add two vertices (with associated edges). To maintain a partitioning into spanning trees, we must expand the preexisting $\frac{k}{2}$ trees to the new vertices, but this is a bit subtle! We need to add the two new vertices to each of the preexisting $\frac{k}{2}$ trees, which takes $2\,\frac{k}{2} = k$ edges connecting the preexisting k vertices to the two new vertices. Its really important that we use only one edge out of each of the original vertices! This is because otherwise, we would use up both new edges out of one of the vertices v_j, but then our final new spanning tree wouldnt be able to reach v_j, so the remaining $k+1$ edges wouldnt be able to form

a spanning tree!
So to do this, we need to split the original k vertices into two equal subsets of $\frac{k}{2}$ vertices each, and connect each half to one of the two new vertices. Once we do that, we can then justify forming a new spanning tree from the remaining edges, which allows us to complete the argument.

5. **Coloring Hypercubes**
Let $G = (V, E)$ be an undirected graph. G is said to be k-vertex-colorable if it is possible to assign one of k colors to each vertex of G so that no two adjacent vertices receive the same color. G is k-edge-colorable if it is possible to assign one of k colors to each edge of G so that no two edges incident on the same vertex receive the same color.
Show that the n-dimensional hypercube is 2-vertex-colorable for every n.

Solution:

Direct proof: Notice that 2-vertex colorable is another way to say bipartite. Thus we are looking for a way to split the vertices into two sets U and V such that every edge in the graph connects some vertex in U to some vertex in V. Let U be the set of vertices with odd parity and let V be the set of vertices with even parity. An edge will always connect a vertices of different parities since an edge corresponds with a bit flip in the bit-definition. Thus every edge crosses U and V.

Inductive proof: Base case: For $n = 1$, the hypercube is a single edge. If we color one vertex red and the other blue we have a 2-vertex coloring, since the adjacent vertices are colored differently.

Inductive Step: Assume weve shown this to hold for n-dimensional hypercubes, we will show this holds for $n + 1$-dimensional hypercubes. Recall that we can define an $n + 1$ dimensional hypercube as two n-dimensional hypercubes where every vertex i in the first hypercube is connected to vertex i in the second hypercube. Considering this definition, for a given $n + 1$ dimensional hypercube, let H_0, H_1 denote the first and second n-dimensional hypercubes, respectively. By the inductive hypothesis, we assume that H_0 is 2-vertex colorable, and therefore there exists some coloring scheme which is a legal 2-vertex coloring of H_0. Given this coloring, we will color H_1 in the opposite coloring scheme which, given a color of vertex i in H_0 assigns the opposite color to the vertex i in H_1. Since we colored the vertices in both H_0 and H_1, we have colored all the vertices in the hypercube.

It remains to show that this coloring scheme is legal. Assume, for purpose of contradiction that there is a given vertex i in H_1 which has an adjacent neighbor

colored with the same color. If that neighbor is in H_1, then this means that the coloring of H_1 is not legal. Observe that if a coloring scheme is a legal 2-vertex coloring on some graph G, then the opposite coloring scheme is also a legal 2-vertex coloring on G. Since we colored H_1 with the opposite scheme of H_0, and H_0 is identical to H_1, this implies that the coloring of H_1 is a legal 2-vertex coloring, which contradicts having two adjacent vertices in H_1 sharing the same color. If the neighbor is in H_0, then we know, by definition of the $n+1$-dimensional hypercube, that the neighbor must be i in H_0. In our coloring however, we colored i in H_0 and i in H_1 in opposite colors, which again contradicts our assumption. Similarly, we can show for the case where i is in H_0