CS 70 Discrete Mathematics and Probability Theory
Fall 2013 Vazirani Graphs Practice

1. **Decomposing a non-Eulerian graph into Walks.**

   We proved in class that a connected (undirected) graph has an Eulerian cycle if and only if every vertex has even degree.

   Prove that if a graph G on n vertices has exactly $2d$ vertices of odd degree, then there are d walks (i.e. paths that can go through the same vertex more than once) that *together* cover all the edges of G (i.e., each edge of G occurs in exactly one of the d walks; and each of the walks should not contain any particular edge more than once).

2. **Directing a graph.**

   Suppose we have an (undirected) graph in which all vertices have even degree. Briefly describe how you would give a direction to each edge such that the indegree equals the outdegree of every vertex in the resulting directed graph.

3. **Trees**

   A *tree* is a connected undirected graph that has no cycles.

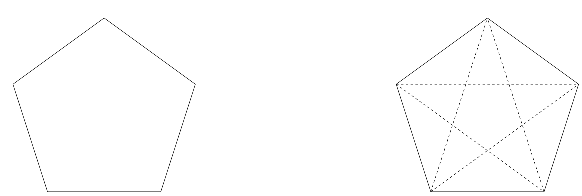
   (a) Show that every tree contains at least one vertex of degree 1.

   (b) Prove by induction on n that every tree with n vertices has exactly $n-1$ edges. [HINT: Use part (a) for the inductive step.]

4. **Polygons**

   A *diagonal* of a polygon is a line connecting two different, non-adjacent vertices. Let $d(n)$ denote the number of diagonals in a polygon with n vertices.

   For instance, shown below on the left is a polygon with five vertices.



   On the right each diagonal has been drawn as a dashed line. (The solid lines are not diagonals.) We can see there are five diagonals. Therefore, $d(5) = 5$.

   (a) Calculate $d(3)$. You do not need to justify your answer.

   $d(3) =$

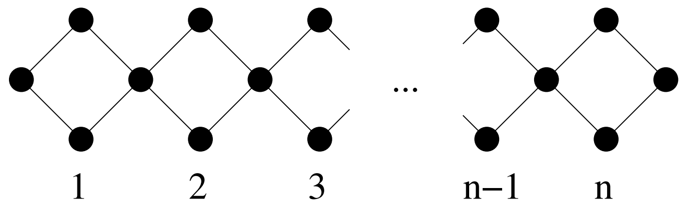
   (b) Calculate $d(4)$. You do not need to justify your answer.

   $d(4) =$

(c) Prove by induction that $d(n) = \frac{n(n-3)}{2}$ for every $n \geq 3$.

5. **Chains**

   Consider "chain" graphs, shaped like this (with n "links"):



   How many different Eulerian tours, starting and ending at the leftmost vertex, does a chain graph with n links have?

6. **Hypercube routing**

   Recall that an n-dimensional hypercube contains 2^n vertices, each labeled with a distinct n bit string, and two vertices are adjacent if and only if their bit strings differ in exactly one position.

   (a) 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.

   (b) In general, for an arbitrary source vertex and arbitrary destination vertex, how many edges must the packet traverse under this algorithm? Give an exact answer in terms of the n-bit strings labeling source and destination vertices.

   (c) Consider the following example where $n = 3$: Suppose that x is (010) and y is (100). What is the length of the shortest path between x and y? What is the set of all vertices and the set of all edges that lie on shortest paths between x and y. Do you see a pattern?

   (d) Answer the last question for an arbitrary pair of vertices x and y in the hypercube. Can you describe the set of vertices and the set of edges that lie on shortest paths between x and y?