CHAPTER 6

GRAPH THEORY

HOMEWORK

- Again, all homework is from the Exercises
  - No problems are from the Review Exercises
- Section 6.1, (p. 271), #5-10, 17-18, 22, 27-28, 46-48
- Section 6.2, (p. 281), #20-21, 28-38, 39, 41
- Section 6.3, (p. 296), #1-7
- Section 6.5, (p. 300), #1-3, 7-9, 13-14, 24-25
- Section 6.6, (p. 305), #1-7
- Section 6.7, (p. 311), #6-9, 18-24

VOCABULARY TO KNOW

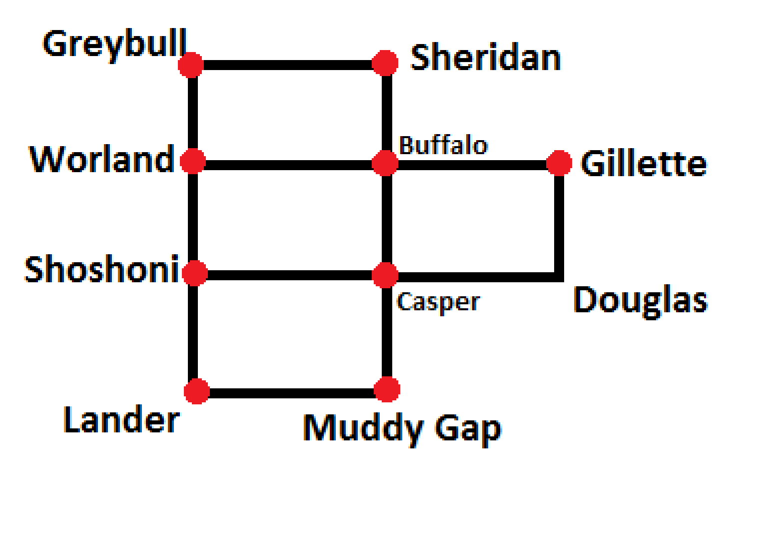
- Vertex
- Edge
- Path
  - List of edges on a "tour" from one vertex to another
- Graph
  - Set of vertices (This is V below)
  - Set of edges (This is E below)
  - Written G = (V, E)
  - V and E must be finite

TYPES OF GRAPHS

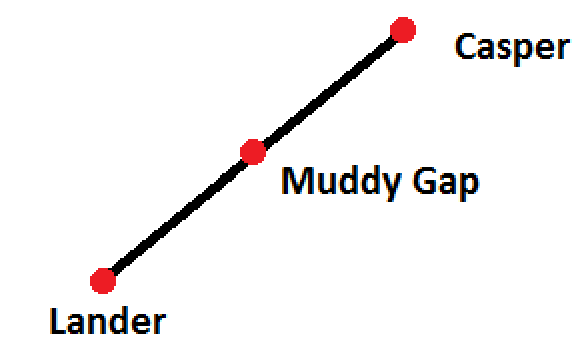
- Undirected Graph
- Directed Graph
  - Also called a digraph
  - Edges have directions
    - This is symbolized by arrows
- Weighted Graph
  - The edges have weights (numbers) on them

SHAPE DOESN'T MATTER; ONLY V AND E MATTER

- The city map from the text



- The bottom can be redrawn

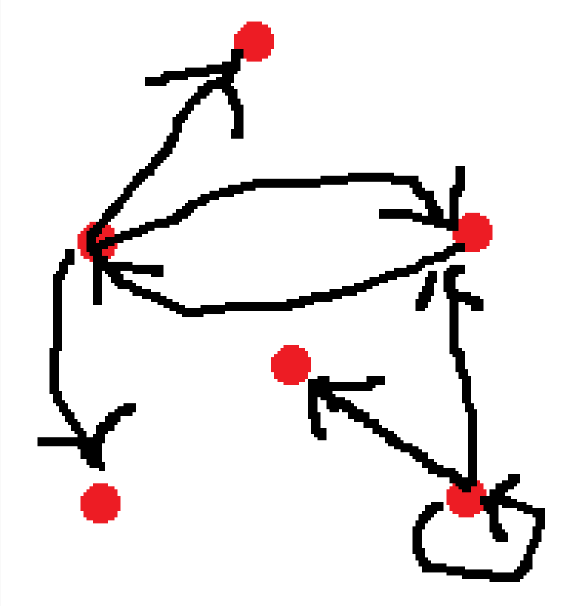


MORE VOCABULARY

- **Incident**
  - A vertex at the end (or beginning) of an edge
    - The vertex is incident on the edge
  - An edge that ends (or begins) at a vertex
    - The edge is incident on the vertex
- **Loop**
  - An edge incident on only one vertex
- **Isolated vertex**
  - Has no incident edges
- **Parallel edges**
  - Two (or more) edges that connect the same two vertices

A DIRECTED GRAPH

- This graph is directed
- It has 7 edges
- It has 6 vertices
- It has a loop



- Notice that you can only follow the arrows on a directed edge
  - You can't go backward

SIMPLE GRAPHS AND CYCLES

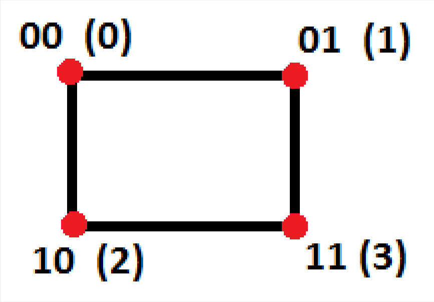
- A simple graph is a graph with
  - No loops
  - No parallel edges
- We will most frequently look at simple graphs
- A simple path is a path from one vertex to another that has no repeated edges
- A cycle is a simple path that
  - Starts and ends at the same vertex, and
  - Has at least one edge

THE N HYPERCUBE

- We have 2^n processors
- Each processor is labeled with 0, 1, 2, …, 2^n-1
  - The numbers are written in binary
- To make the graph
  - The processors are the vertices
  - The edges connect processors whose labels differ by only one bit
- A hypercube is useful for parallel computing

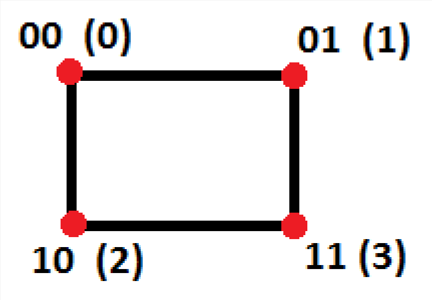
A RECURSIVE DESCRIPTION

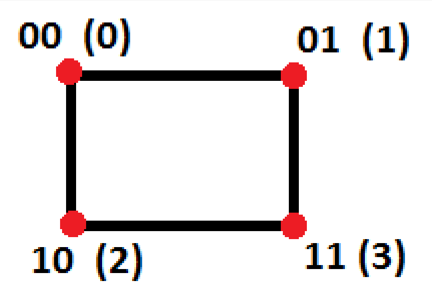
- Base cases



A RECURSIVE DESCRIPTION

- Duplicate the picture

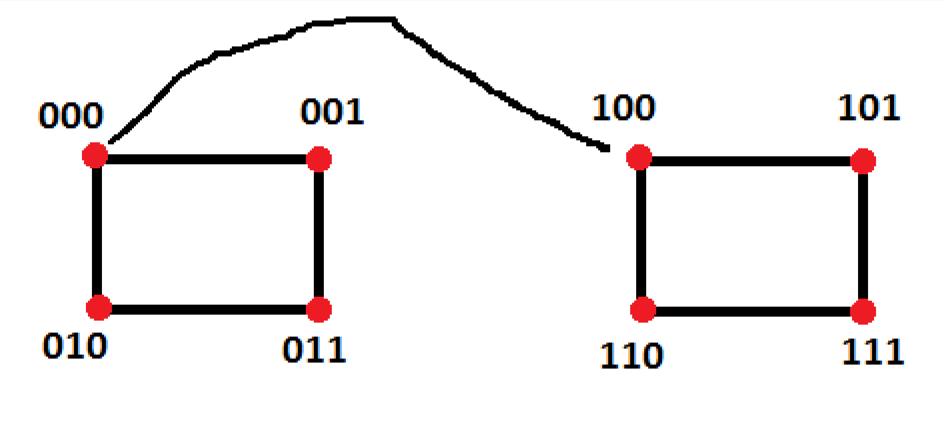




- Connect vertices with the same labels
- Add a 0 to the beginning of each vertex label on the left
- Add a 1 to the beginning of each vertex label on the right

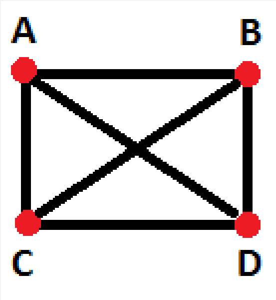
THE RESULTING PICTURE

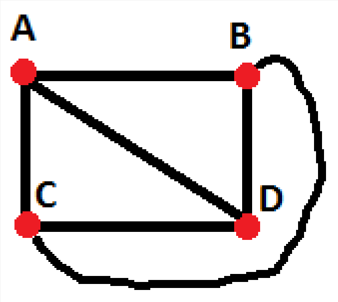
- **Only one connecting vertex is drawn**



COMPLETE GRAPHS

- A complete graph has an edge between each pair of vertices
  - It's a graph with "every" possible edge
- We call them K_n, where n is the number of vertices
- Here are two pictures of K_4

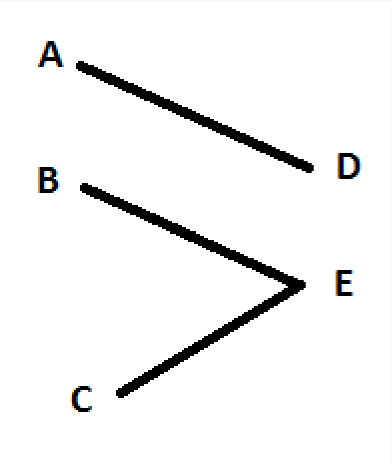




- Notice that K_n has n(n-1)/2 edges

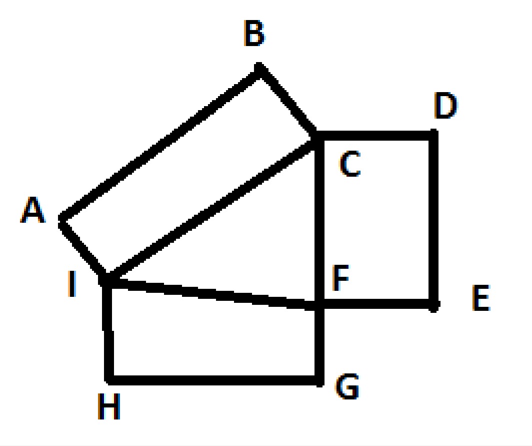
BIPARTITE GRAPHS

- A bipartite graph is one where
  - You can break the vertex set into two parts, and
  - All vertices go from one set to the other
- Here is a picture of a bipartite graph



A GRAPH THAT IS NOT BIPARTITE

- This graph is not bipartite
- If so, the vertices can be put into two sets, V and W
  - Edges only go from one set to the other
- How about C, F, and I?
- You need three sets; two won't do



A COMPLETE BIPARTITE GRAPH

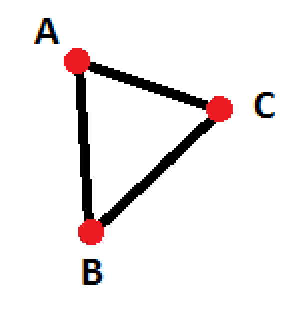
- A **complete bipartite** graph is one where
  - the graph is bipartite, and
  - every possible edge between the vertex sets is in the graph
- We write this as $K_{m,n}$
  - m is the number of vertices in one vertex set
  - n is the number of vertices in the other

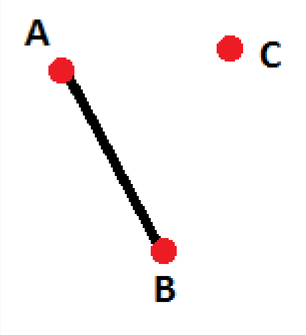
A SUBGRAPH

- Start with a graph
- Choose some edges from the original graph
- Choose <u>all</u> vertices incident on those edges
  - This restriction is so the subgraph is actually a graph
- This is a **subgraph** of the original graph

A CONNECTED GRAPH

- A graph is **connected** if there is a path between every pair of vertices
- The graph on the left is connected; the one on the right is not





- The graph on the right has two **components**

THE DEGREE OF A VERTEX

- The **degree of a vertex** is the number of edges that are incident on the edge
- Special case
  - If there is a loop, this adds 2 to the degree instead of 1

THE KÖNIGSBERG BRIDGES AGAIN

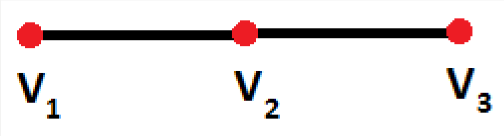
- Euler Cycles
  - An Euler cycle is a cycle that contains all edges and all vertices
- Theorem 6.2.17
  - If a graph has an Euler cycle, then every vertex has even degree
- So the Königsberg has no Euler cycle
- The converse is also true (Theorem 6.2.18)
  - If a graph is connected and every vertex has even degree, then the graph has an Euler cycle

PROOF OF THEOREM 6.2.18

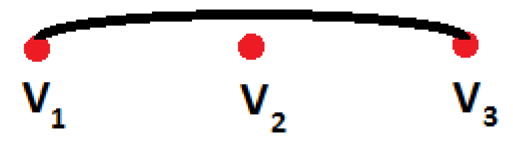
- The Theorem is
  - If a graph is connected and every vertex has even degree, then the graph has an Euler cycle
- The proof will be induction on the number of edges
- Suppose G has 1 edge and every vertex has even degree
  - What can G look like?
- Suppose G has 2 edges, is connected, and every vertex has even degree
  - What can G look like?
  - We don’t actually need this second case, but it makes us familiar with the theorem

THEOREM 6.2.18

- The inductive step is
  - Assume that any connected graph with fewer than n edges and even degree for every vertex has an Euler cycle
  - Show that any connected graph with n edges and even degree for every vertex has an Euler cycle
- So now assume that G is connected and it has at least two edges
  - The induction hypothesis is that any connected graph with fewer than n edges and ever vertex having even degree has an Euler cycle
- Since G has at least two edges it contains a picture like this



- We create a new graph by changing that picture to



- Call the new graph H

CONTINUING THE PROOF OF THE THEOREM

- How is H different from G?
  - What did we do the number of vertices?
  - What did we do to the number of edges?
  - What did we do the degrees of vertices in H?
- I show that H has 1 or 2 connected components

CONTINUING THE PROOF OF THE THEOREM

- Let v be any vertex in G except v_1
- Since G is connected, there is a path in G from v to v_1
- Let P_1 be the part of the path that has its edges in H
- P_1 must end at v_1, v_2, or v_3 in G
- There are three possibilities for the path in H
  - If it ends at v_1, then, v is in v_1's component
  - If it ends at v_2, then, v is in v_2's component
    - This is also true if v was chosen to be v_2 and the path in G was v_2-v_1
  - If it ends at v_3, then, v is in v_3's component , which is the same as v_1's component
- This shows that H every vertex in H is in either in v_1 or v_2's component
- So H has either one or two components

FINISHING THE PROOF OF THE THEOREM

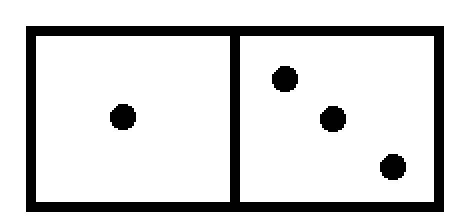
- If H has one component, then it has an Euler cycle
  - Why is that?
- So, then G has essentially that same Euler cycle
  - We change the new edge back to its original two edges

- If H has two components, each has an Euler cycle
  - We can assume the first starts and ends at v_1
  - We can also assume the second starts and ends at v_2
  - We modify it by changing (v_1, v_3) to (v_1, v_2) (v_2, v_3)
- Either way, we get an Euler cycle in G

AN INTERESTING IDEA

- **Theorem:** In a graph with m edges and n vertices, the sum of the degrees of the vertices is 2m
  - This is easy to see since each edge is counted twice
  - It's counted for each vertex it has
- **Corollary:** A connected graph has an even number of vertices of odd degree
  - Break the vertices into two sets
    - Even degree vertices, odd degree vertices
  - Sum the degrees of the odd vertices
  - The sum is even (by the theorem) so the sum of the degrees of the odd vertices must be even too

DOMINOES



- Here is a picture of a domino
- Each side can have 0-6 dots
- Suppose a set of dominoes contains all 49 dominoes
- Question: Can we arrange the dominoes in a circle so that adjacent dominoes have the same number of dots?
- To get the graph
  - The numbers 0, …, 6 are the vertices
  - The edges are "the dominoes"
    - There is an edge between every pair of vertices and a loop at each vertex

THE DOMINO CIRCLE

- What is the degree of each vertex?
- Also notice that the graph is connected
- The theorem guarantees that there is an Euler cycle
- That shows how to arrange the dominoes in a circle

REPRESENTATIONS OF GRAPHS IN SOFTWARE

- A graph can be represented by
  - An adjacency matrix
  - An incidence matrix
  - A linked list of vertices, with other linked lists showing the edges

A PROGRAM TO FIND THE SHORTEST PATH IN A GRAPH

- **Let's look at the program**

HOMEWORK

- **Again, all homework is from the Exercises**
  - **No problems are from the Review Exercises**
- **Section 6.1, (p. 271), #5-10, 17-18, 22, 27-28, 46-48**
- **Section 6.2, (p. 281), #20-21, 28-38, 39, 41**
- **Section 6.3, (p. 296), #1-7**
- **Section 6.5, (p. 300), #1-3, 7-9, 13-14, 24-25**
- **Section 6.6, (p. 305), #1-7**
- **Section 6.7, (p. 311), #6-9, 18-24**