CHAPTER 5

Section 5.1

Graphs

- Informal definition of a **graph**
  - A nonempty set of **nodes** (**vertices**) and a set of **arcs** (**edges**) such that each arc connects two nodes.
  - Our graphs will always have a finite number of nodes and arcs.
  - Example: Airline route map

Undirected Graphs

- Formal Definition of a Graph
  - An ordered triple (N, A, g) where
    - N = a nonempty set of **nodes** (**vertices**)
    - A = a set of **arcs** (**edges**)
    - g = a function associating with each arc a an *unordered* pair x–y of nodes called the **endpoints** of a
  - Note this refers to an **undirected** graph

Undirected Graphs

- Example 3
  - Consider the graph of Figure 5.3 (Example 2)
  - Describe the function g which associates arcs with endpoints.

Directed Graphs

- In a **directed graph**, each arc has a direction associated with it
  - Also called a **digraph**.
- Formal Definition of a **Directed Graph**
  - An ordered triple (N, A, g) where
    - N = a nonempty set of nodes
    - A = a set of arcs
    - g = a function associating with each arc a an *ordered* pair (x,y) of nodes where x is the initial point and y is the terminal point of a

Directed Graphs

- Example 4
  - Consider the directed graph in Figure 5.4
  - How many nodes?
  - How many arcs?
  - How does the function g map endpoints to arcs?

Graphs

- **Labeled graph**
  - A graph which carries identifying information on the nodes and/or arcs
  - Example: City names on airline route map
- **Weighted graph**
  - Each arc has a weight, which is a numerical value, associated with it.
  - Example: Distance from city to city
- Book uses "graph" to mean undirected. They will explicitly state directed graph.

Applications of Graphs

- Hasse Diagrams
- PERT charts
- E-R diagrams (Entity-Relationship)
- Logic networks
- Finite state machines
- Data Flow diagrams
- Conceptual Graphs
- Chemical structure of a molecule

Graph Terminology

- **Adjacent**: 2 nodes are adjacent if they are the endpoints associated with an arc.
  - A node can be adjacent to itself
- **Loop**: an arc with endpoints n–n for some node n
  - A graph with no loops is called **loop-free**.
- **Parallel arcs**: 2 arcs with the same endpoints
- **Simple graph**: one with no loops or parallel nodes

Graph Terminology

- **Isolated node**: a node which is adjacent to no other node
- **Degree** of a node: the number of arc ends at that node
  - In a directed graph, ends with arrows count toward the **in-degree** and ends without arrows count toward the **out-degree**.

Graph Terminology

- **Complete graph**: one in which any two distinct nodes are adjacent
- **Subgraph** of a graph: consists of a set of nodes and a set of arcs that are subsets of the original node set and arc set, respectively, in which the endpoints of an arc must be the same nodes as in the original graph.
  - A graph obtained by erasing part of the original graph and leaving the rest unchanged.

Graph Terminology

- **Path** from node n_0 to node n_k: a sequence n_0, a_0, n_1, a_1, … $n_{\mathrm{k-1}}$, $a_{\mathrm{k-1}}$, n_{k},
  of nodes and arcs where for each i, the endpoints of arc a_i are n_i–n_{i+1}.
- **Length** of a path: the number of arcs it contains
  - An arc is counted each time it is used in the path.

Graph Terminology

- **Connected graph**: one in which there is a path from any node to any other node.
- **Cycle**: a path in a graph from some node n_0 back to n_0 where
  - No arc appears more than once in the path sequence
  - n_0 is the only node appearing more than once
  - n_0 occurs only at the ends
- **Acyclic graph**: a graph with no cycles

Practice 1

- Sketch a graph having
  - nodes $\{1, 2, 3, 4, 5\}$,
  - arcs $\{a_1, a_2, a_3, a_4, a_5, a_6\}$, and
  - function

    $g(a_1)=1\text{–}2,\quad g(a_2)=1\text{–}3,\; g(a_3)=3\text{–}4,$
    $g(a_4) = 3\text{–}4,\; g(a_5)=4\text{–}5,\; g(a_6)=5\text{–}5.$

Practice 3

- In the graph of practice 1:

1. Find 2 nodes that are not adjacent.
2. Find a node adjacent to itself.
3. Find a loop.
4. Find two parallel arcs.
5. Find the degree of node 3.
6. Find a path of length 5.
7. Find a cycle.
8. Is this graph complete?
9. Is this graph connected?

Bipartite Complete Graph

- Definition of **Bipartite Complete Graph**
  - A graph in whose nodes can be partitioned into 2 disjoint nonempty sets N_1 and N_2, such that
    - 2 nodes x and y are adjacent if and only if $x \in N_1$ and $y \in N_2$.
    - If $|N_1| = m$ and $|N_2| = n$, such a graph is denoted by $K_{m,n}$.
- Graph can be divided into two distinct sets such that
  - The nodes in the first set are not adjacent to one another.
  - The nodes in the second set are not adjacent to one another.
  - If you chose any node from the first set and any node from the second set, those nodes are adjacent.

Bipartite Complete Graph

- Practice 5
  - Draw $K_{3,3}$

Directed Graph Terminology

- **Path** from node n_0 to node n_k in a directed graph: a sequence n_0, a_0, n_1, a_1, … n_{k-1}, a_{k-1}, n_{k}, where for each i, n_i is the initial point and n_{i+1} is the terminal point of a_i.
- If a path exists from node n_0 to node n_k, then n_k is **reachable** from n_0.
- **Cycle** is the same for directed graphs.

Example 9

- Consider the digraph of Figure 5.13
  - List some paths from node 1 to node 3
  - Is node 4 reachable from node 2?
  - Is node 1 reachable from node 3?
  - List any cycles in the graph.

Isomorphic Graphs

- Consider Figures 5.14, 5.15 and 5.16.
- Ignoring labeling, are these graphs structurally the same?
  - Called **isomorphic graphs**
  - Have to be able to produce a relabeling that preserves all of the essential structure information (the nodes and the arcs between the nodes)
    - The relabeling must be one-to-one, onto mappings between the elements of the 2 structures.
    - What is a relabeling to map graph in 5.14 to 5.16?

Isomorphic Graphs

- Formal Definition of Isomorphic Graphs
  - Two graphs (N_1, A_1, g_1) and (N_2, A_2, g_2) are isomorphic if there are bijections
    - $f_1: N_1 \to N_2$ and
    - $f_2: A_1 \to A_2$
  - such that for each arc $a \in A$,
    - $g_1(a) = x\text{–}y$ if and only if $g_2[f_2(a)] = f_1(x)\text{–}f_1(y)$.

Practice 7

- Complete the definition of the function f_2 in Example 11.
- The bijections that establish the isomorphism are partially given on the right:

$$
\begin{array}{llll}
f_1: & 1 \to c & f_2: & a_1 \to e_1 \\
& 2 \to e & & a_2 \to e_4 \\
& 3 \to d & & a_3 \to e_2 \\
& 4 \to b & & \\
& 5 \to a & &
\end{array}
$$

Simple Graph Isomorphism

- Theorem on Simple Graph Isomorphism
  - Two simple graphs (N_1, A_1, g_1) and (N_2, A_2, g_2) are isomorphic is there is a bijection
    - $f: N_1 \rightarrow N_2$
  - such that for any nodes n_i and n_j of N_1, n_i and n_j are adjacent if and only if $f(n_i)$ and $f(n_j)$ are adjacent.
  - The function f is called an **isomorphism** from graph 1 to graph 2.

Practice 8

- Find an isomophism from graph of Figure 5.18a to that of Figure 5.18b.

Planar Graphs

- A planar graph is one that can be represented in the plane so that its arcs intersect only at nodes.
  - Don't want connections to cross (ex., integrated circuits)
  - Practice 10: Prove that K_4 is a planar graph

Euler's Formula

- Consider a simple, connected, planar graph (when drawn in planar representation, with no arcs crossing)
  - It divides the plane into a number of regions, including totally enclosed regions and one infinite exterior region.
- Euler observed a relationship between the number n of nodes, the number a of arcs and the number r of regions in such a graph.
- This relationship is know as Euler's Formula

$$n - a + r = 2$$

Practice 13

- Verify Euler's formula for the simple, connected, planar graph in Figure 5.18b.

Representation of Graphs

- Computer representation usually involves one of the following data structures:
  - Adjacency matrix
  - Adjacency list

Adjacency Matrix

- Adjacency matrix
  - Given a graph with n nodes $(n_1, n_2, \ldots, n_n)$.
  - Form an $n \times n$ matrix where entry i,j is the number of arcs between nodes n_i and n_j.
    - Called adjacency matrix of the graph with respect to this ordering (of the nodes above)
  - $a_{ij} = p$ where there are p arcs between n_i and n_j

Adjacency Matrix

- Practice 16
  - Complete the adjacency matrix for Figure 5.25.

Adjacency Matrix

- Example 18
  - What is the adjacency matrix for the graph in Figure 5.26?
  - For a simple *weighted* graph, the weight can be used instead of a 1 in the matrix.

Adjacency Matrix

- Disadvantages of adjacency matrices
  - Often adjacency matrices are **sparse**
    - The matrix has a lots of zeros, because the graph does not contain many arcs
  - Still need n^2 data items to represent the matrix of a graph with n nodes, even if it contains mostly 0s.
  - Any algorithm that needs to examine every arc has to scan all n^2 matrix values.
  - Can be inefficient.
- Use an adjacency list instead.

Adjacency List

- For each node, keep a list of all the nodes adjacent to it.
  - Each list is stored as a linked list.
  - To find all the nodes adjacent to some node, just traverse the linked list for that node.
  - Tradeoff: what if we want to know if node n_j is adjacent to node n_i?

Adjacency List

- Practice 17
  - Draw the adjacency list representation for the graph shown in Figure 5.28.