Chapter 6.1
Graphs and their Representations

Instructor: Abhishek Santra

Email: abhishek.santra@uta.edu

1

What is the basis of a graph?

- Are you on social media?
  - Facebook, Instagram, …
- Do you use flights while traveling?
  - American, Lufthansa, …
- Do you use any navigation while driving?
  - Google Maps, Apple Maps, …

2

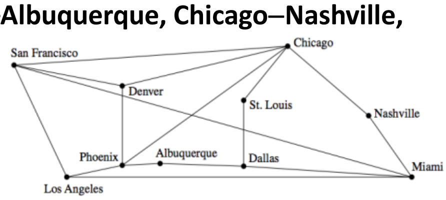
What is the basis of a graph?

- What are the ***major entities or participants***?
  - **Social Media**: People
  - **Airline Travel:** Airports or Cities
  - **Navigation:** Cities

- How are they ***related***? Can they be ***connected***?
  - People
    - – ***Friends, Caller-Receiver, …***
  - Cities/Airports
    - – ***Direct Flights, Roads, …***

Informal Definition of a Graph

- A **graph** is
  - A nonempty set of **nodes (**also called **vertices),** and
  - A set of **arcs (**also called **edges)** such that each arc *connects two nodes*
- **Example:**
  - The set of nodes in the airline map below is **{Chicago, Nashville, Miami, Dallas, St. Louis, Albuquerque, Phoenix, Denver, San Francisco, Los Angeles}**
  - There are 16 arcs; **Phoenix–Albuquerque, Chicago–Nashville, Miami–Dallas,** and so on.



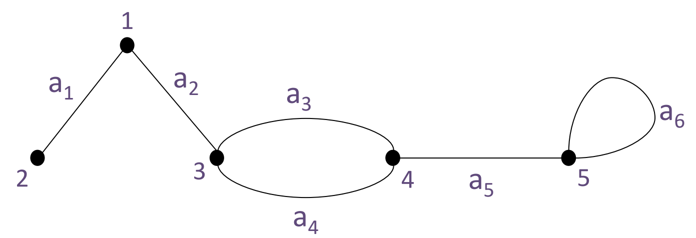
Formal Definition of a Graph

- Without the visual representation of a graph, we need a concise way to convey the same information.

- **DEFINITION (FORMAL):** A **graph** is 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*
    - – Each arc has unique endpoints

Formal Definition of a Graph: Example

- A graph having
  - A set of nodes {1, 2, 3, 4, 5},
  - A set of arcs $\{a_1, a_2, a_3, a_4, a_5, a_6\}$, and
  - function $g(a_1)$ = 1–2, $g(a_2)$ = 1–3, $g(a_3)$ = 3–4, $g(a_4)$ = 3–4, $g(a_5)$ = 4–5, and $g(a_6)$ = 5–5.

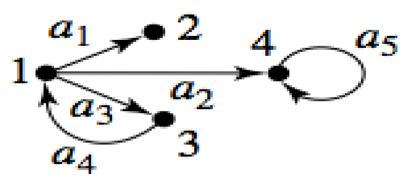


Directed Graphs

- *Requirement: Direct Flights between cities?*
  - *Directed Graph*: Arcs of a graph **begin at one node** and **end at another**.
  - Direction associated with each arc, *denoted by arrows*
- A **directed graph (digraph)** is 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 (source)** and y is the **terminal point (destination)** of a
    - Each arc has unique endpoints

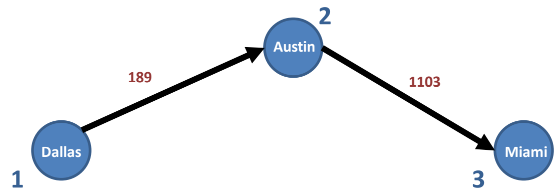
Directed Graphs: Example

- Nodes: N = {1, 2, 3, 4}
- Arcs/Edges: A = {a_1, a_2, a_3, a_4, a_5}
- The function g
  - $g(a_1) = (1, 2)$, *meaning that arc a_1 begins at node 1 and ends at node 2*
  - Also, $g(a_3) = (1, 3)$, but $g(a_4) = (3, 1)$.



Other Forms of Graphs

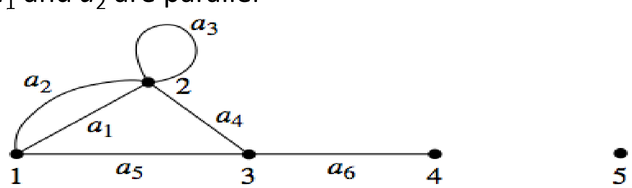
- **Labeled graph:** A graph whose nodes carry identifying information
  - **Names of the cities** in the map of airline routes



- **Weighted graph:** A graph where each arc has some numerical value, or weight, associated with it
  - **Distances (in miles)** of the various routes in the airline map
- The term "graph" is used to mean an undirected graph. To refer to a directed graph, one always says "directed graph."

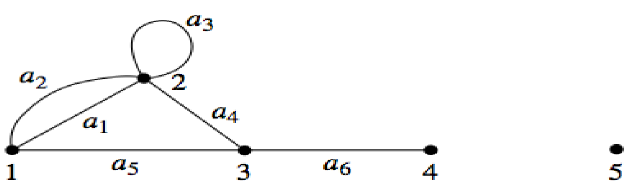
Graph Terminology

- **Adjacent Nodes**: The endpoints associated with an arc
  - 1 and 3 are adjacent nodes, but 1 and 4 are not.
- **Loop**: An arc with endpoints *n*–*n* for some node *n*
  - Arc a_3 is a loop with endpoints 2–2
  - A graph with no loops is **loop-free**
- **Parallel Arcs**: Two arcs with the same endpoints (undirected) or same start and end points (directed)
  - Arcs a_1 and a_2 are parallel



Graph Terminology

- **Simple Graph**: A **graph** with no loops or parallel arcs
- **Isolated Node**: A **node** that is adjacent to <u>no other node</u>, a node with no associated arc or edge
  - 5 is an isolated node
- **Degree of a node**: Number of *<u>arc ends</u>* at that node
  - Nodes 1 and 3 have degree 3, node 2 has degree 5, node 4 has degree 1, and node 5 has degree 0

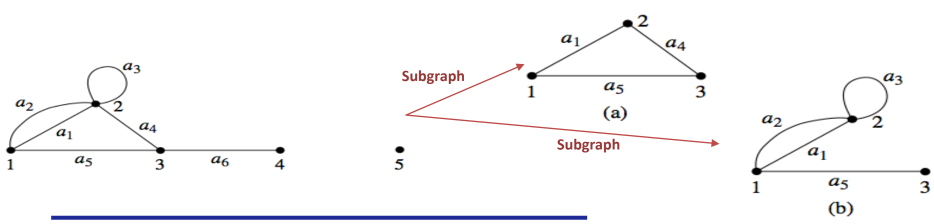


11

Graph Terminology

- **Complete Graph:** A graph in which *<u>every two distinct nodes</u> are adjacent OR all possible arcs exist.* Example (a)

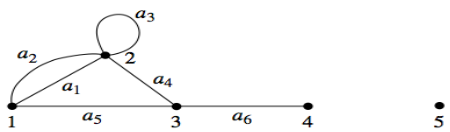
- **Subgraph** of a graph: Consists of a set of nodes and a set of arcs that are <u>subsets of the original node set and arc set</u>, respectively, in which the endpoints of an arc must be the same nodes as in the original graph.



13

Graph Terminology

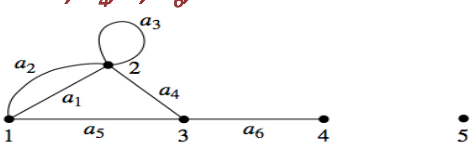
- A **path** from node n_0 to node n_k is a sequence
  $n_0, a_0, n_1, a_1, \dots, n_{k-1}, a_{k-1}, n_k$
  of nodes and arcs where, for each i, the endpoints of arc a_i are n_i–n_{i+1}. If such a path exists, then n_k is **reachable** n_0.
  - *One path from **node 2 to node 4** consists of the sequence*
    2, a_1, 1, a_2, 2, a_4, 3, a_6, 4



- **Length of a path:** Number of arcs it contains; *if an arc is used more than once, it is counted each time it is used*
  - *The length of the path described above from node 2 to node 4 is* ***4***

Graph Terminology

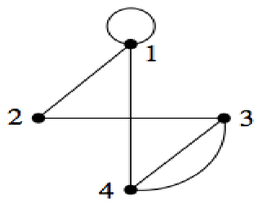
- A graph is **connected** if there is a path from any node to any other node
  - *The graph below is not connected because of node 5*
- A **cycle** in a graph is a path from some **node n_0 back to n_0**,
  - where <u>no arc appears more than once</u> in the path sequence
  - n_0 is the only node <u>appearing more than once</u>, and n_0 occurs only <u>at the ends</u>
- Example:
  - Cycle: ***2, a_1, 1, a_5, 3, a_4, 2***
  - Not a cycle: ***2, a_4, 3, a_6, 4***



- A graph with <u>no cycles</u> is **acyclic**

Graph Representation: *Adjacency Matrices*

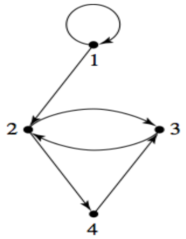
- Suppose a graph has ***n* nodes** numbered $n_1, n_2, \dots, n_n$
  - Having ordered the nodes, we can form an $n \times n$ **matrix,**
  - where **entry *i, j* is the number of arcs <u>between</u> nodes n_i and n_j**
  - This matrix is called the **adjacency matrix A** of the graph with respect to this ordering
  - Thus, $a_{ij} = p$ where there are ***p* arcs <u>between</u>** n_i **and** n_j
- For example, the following **<u>undirected graph</u>** has a corresponding adjacency matrix.



$$A = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 \\ 1 & 0 & 2 & 0 \end{bmatrix}$$

Graph Representation: *Adjacency Matrices*

- In a directed graph, the adjacency matrix A **reflects the direction of the arcs**
- For a directed matrix, $a_{ij} = p$ where there are ***p* arcs <u>from</u>** n_i **to** n_j.
- For example, the following **<u>directed graph</u>** has a corresponding adjacency matrix.



$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Adjacency Matrix Drawback

- Graph with *n* nodes requires n^2 data items to represent and store the adjacency matrix.
- Many graphs, *far from being complete graphs*, have relatively few arcs.
  - **Sparse** adjacency matrices; that is, the adjacency matrices contain many zeros
- Leads to expensive computation of any algorithm in which every arc in the graph must be examined; requires *looking at all n^2 items in the matrix*.

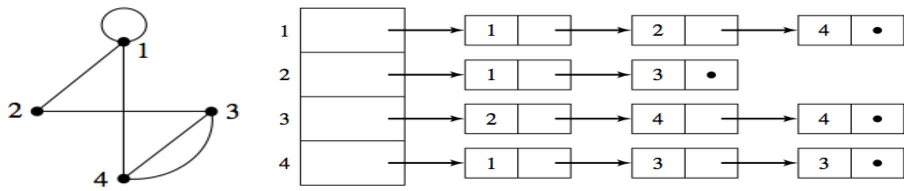
Graph Representation: *Adjacency Lists*

- **Efficient Storage Alternative**: Storing only the nonzero entries of the adjacency matrix!
- **Adjacency List:** For **each node**, consists of a list of all adjacent nodes
  - **Pointers** are used to get us from one item in the list to the next. Such an arrangement is called a **linked list.**
  - An array of pointers is maintained

    n elements in the array, one for each node

Graph Representation: *Adjacency Lists Example*

- Adjacency list for the graph contains a **four-element array of pointers, one for each node**
- The pointer for each node **points to an adjacent node**, which points to another adjacent node, and so forth.
- The dot indicates a **null pointer,** meaning that there is nothing more to be pointed to or that the **end of the list has been reached**.



Discussion