computer science

illuminated

Trees & Graphs

Nell Dale & John Lewis

(adaptation by Michael Goldwasser)

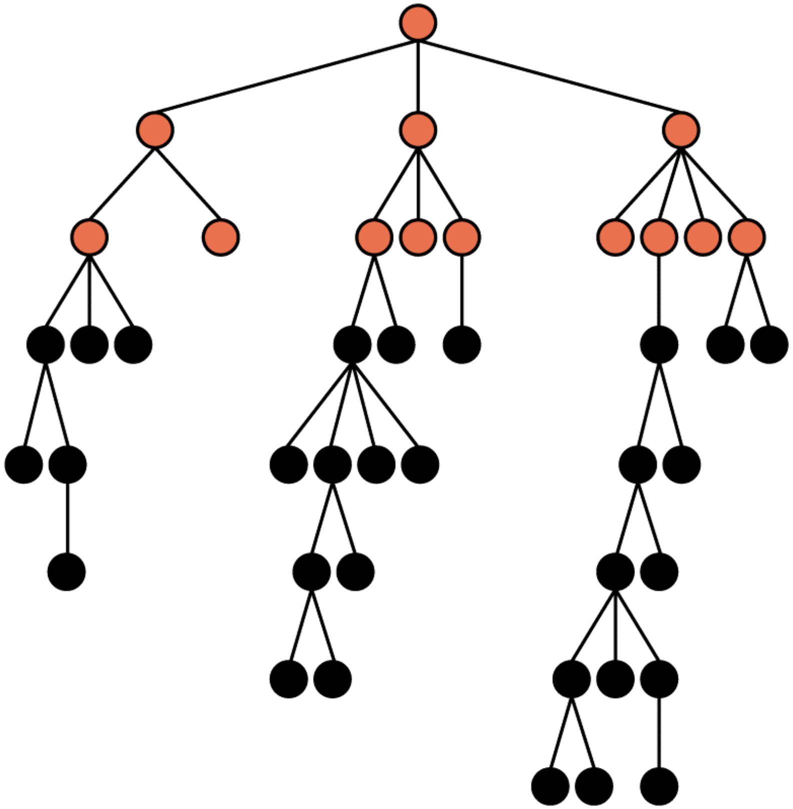
Trees

- Arrays and Linked Lists represent data which is inherently linear.
- More complex relationships require more complex structures.
- A common set of relationships is a “hierarchy”

Hierarchies

- Company Organization
  (President, VPs, Managers, …)
- Biology Taxonomy
  (Kindom, Phylum, Class, …)
- Genealogy (Abraham, Isaac, Jacob, …)
- Table of Contents for a text book
- File Systems (Folders, Subfolders, …)
- Web Portals (e.g., Yahoo’s catagories)

Terminology



- This is a **tree**
- The positions are **nodes**
- The topmost node is the **root**
- Nodes at the other extreme are called **leaves**
- A node may have a **parent**, **ancestors**, **children**, **siblings** or **descendants**
- A natural recursive view leads us to discussing **subtrees** of the tree

Binary Trees

- Binary trees
  - A tree in which each node has at most two children
  - The node to the left of a node, if it exists, is called its **left child**
  - The node to the right of a node, if it exists, is its **right child**

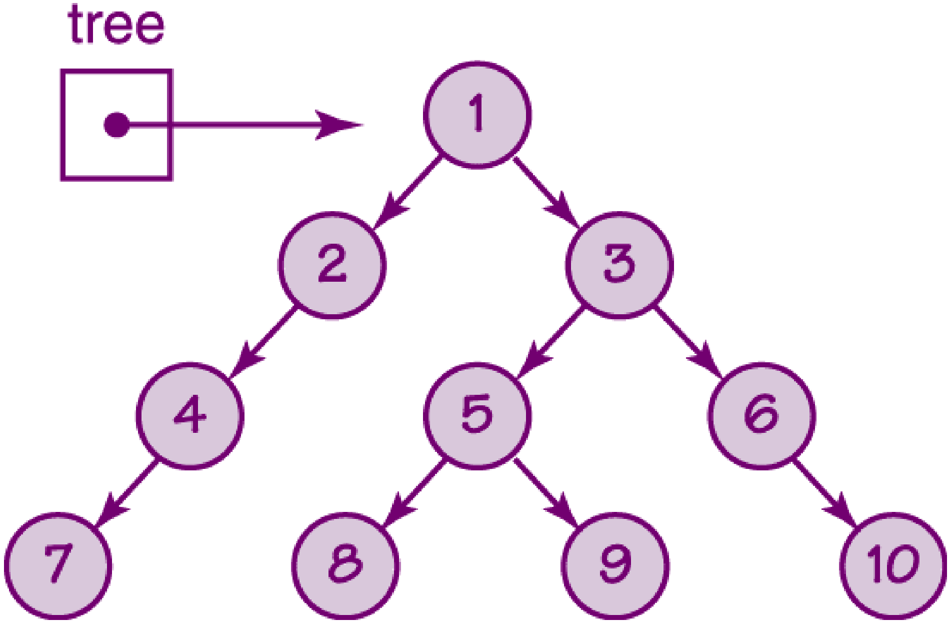


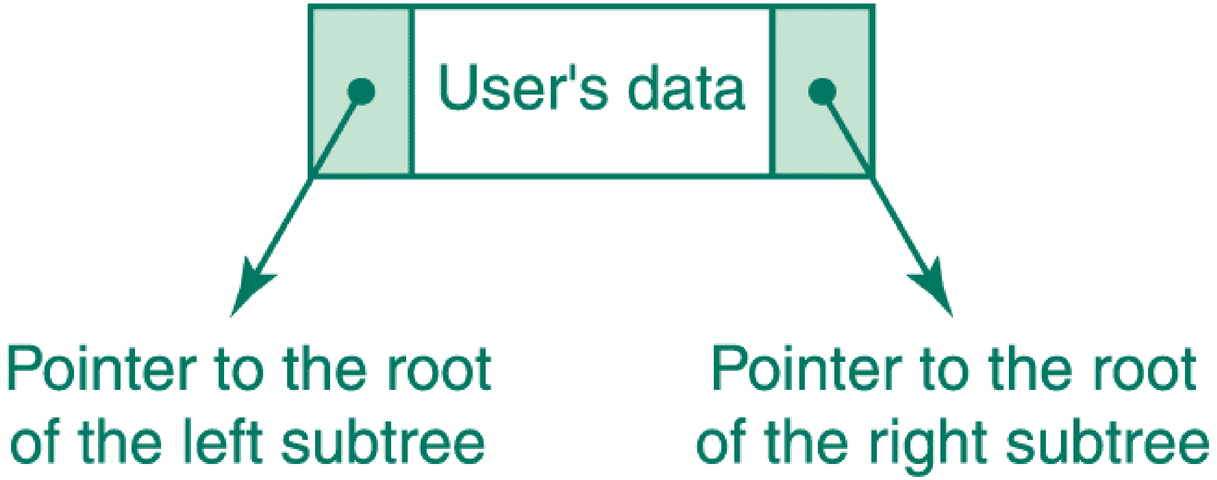
Figure 9.16 **A binary tree**

Representation

- We can represent a binary tree as a linked structure (similar to linked lists)
- A node of the tree might be represented by three consecutive cells of memory
  - User’s Data
  - Explicit Pointer to Left Child
  - Explicit Pointer to Right Child

    (we will use “null” pointer if no such child)

Representation (cont)



User's data

Pointer to the root of the left subtree

Pointer to the root of the right subtree

Page 305

A Simple Database

Let’s revisit idea of maintaining a list of names, while supporting the following operations:

- **search** for the presence of an entry
- **print** names in alphabetical order
- **insert** new names

How should we accomplish this?

A Simple Database

- Use an (alphabetized) array ?
  - can do binary search
  - straightforward to print alphabetically
  - but inserting new item can be costly
- Use a (sorted) linked list?
  - easy to insert item, if we know the location
  - straightforward to print alphabetically
  - but cannot search efficiently
    (can’t binary search; no way to jump to middle)

Binary Search Trees

- A *binary search tree* is a special kind of binary tree.
- A binary search tree has a semantic property among the values in the nodes in the tree:
  - The value in any node is greater than the value in any node in its left subtree and less than the value in any node in its right subtree

Binary Search Tree

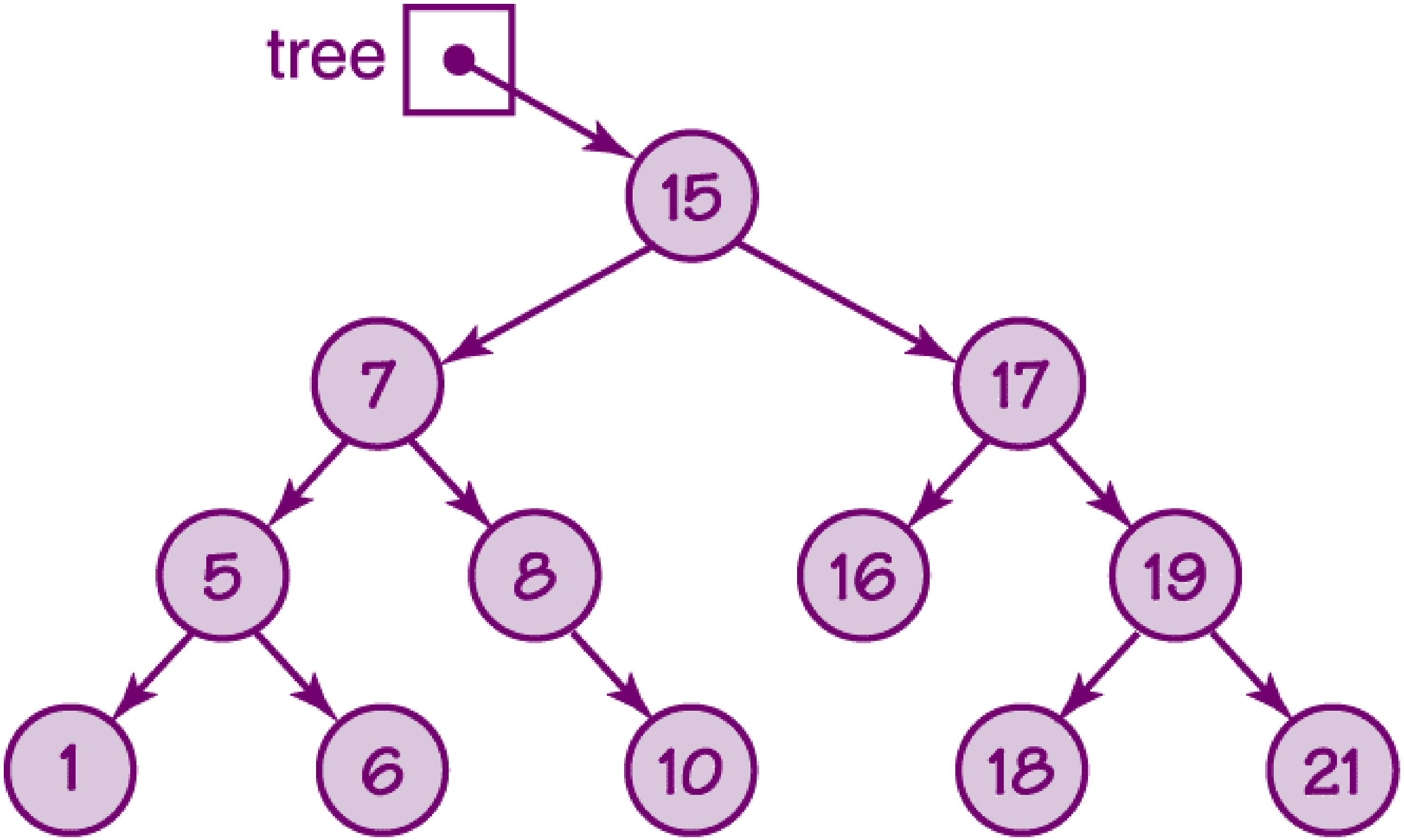


Figure 9.18 **A binary search tree**

Searching

```
Boolean isThere(current, item)
If (current is null)
        return false
Else
        If (item = info(current))  return true
        If (item < info(current))
                return IsThere(left(current), item)
        If (item > info(current))
                return IsThere(right(current), item)
```

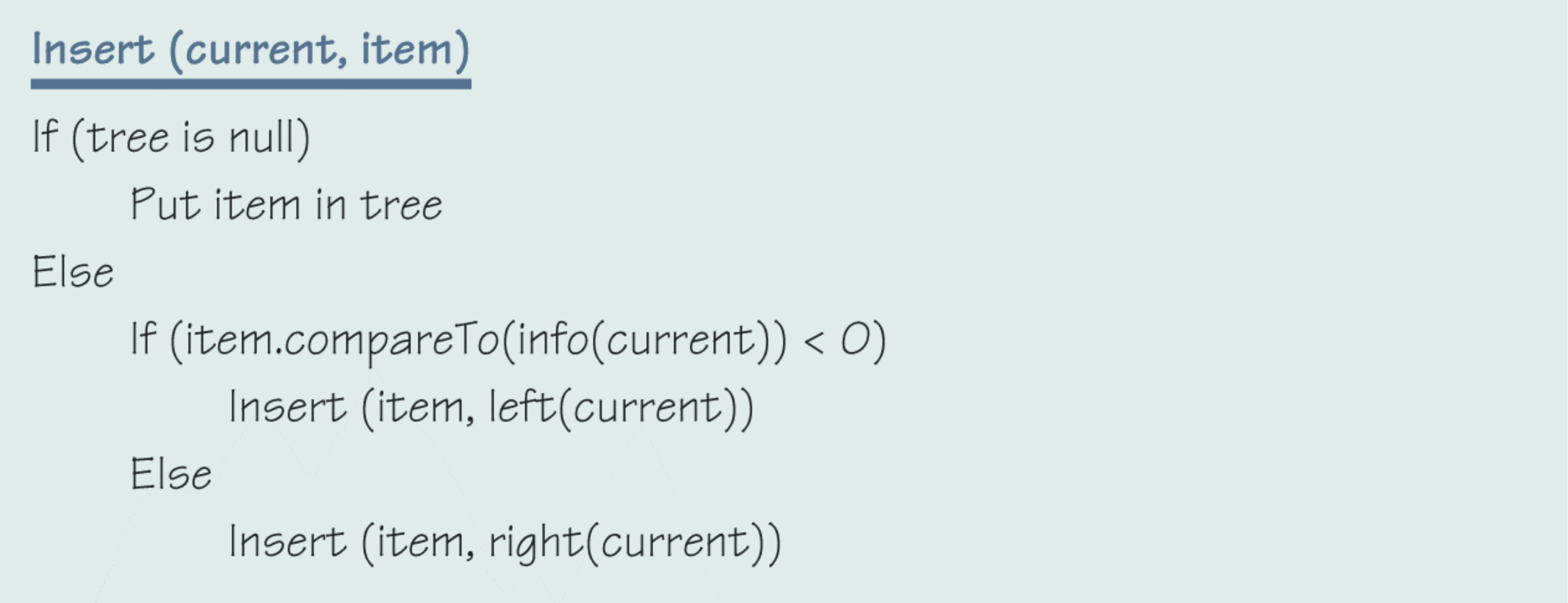
Alphabetical Printing

<u>Print(tree)</u>

```
If (tree is NOT null)
    Print(left(tree))
    Write info(tree) to output
    Print(right(tree))
```

Why does this work?

Insertion



Insert (current, item)

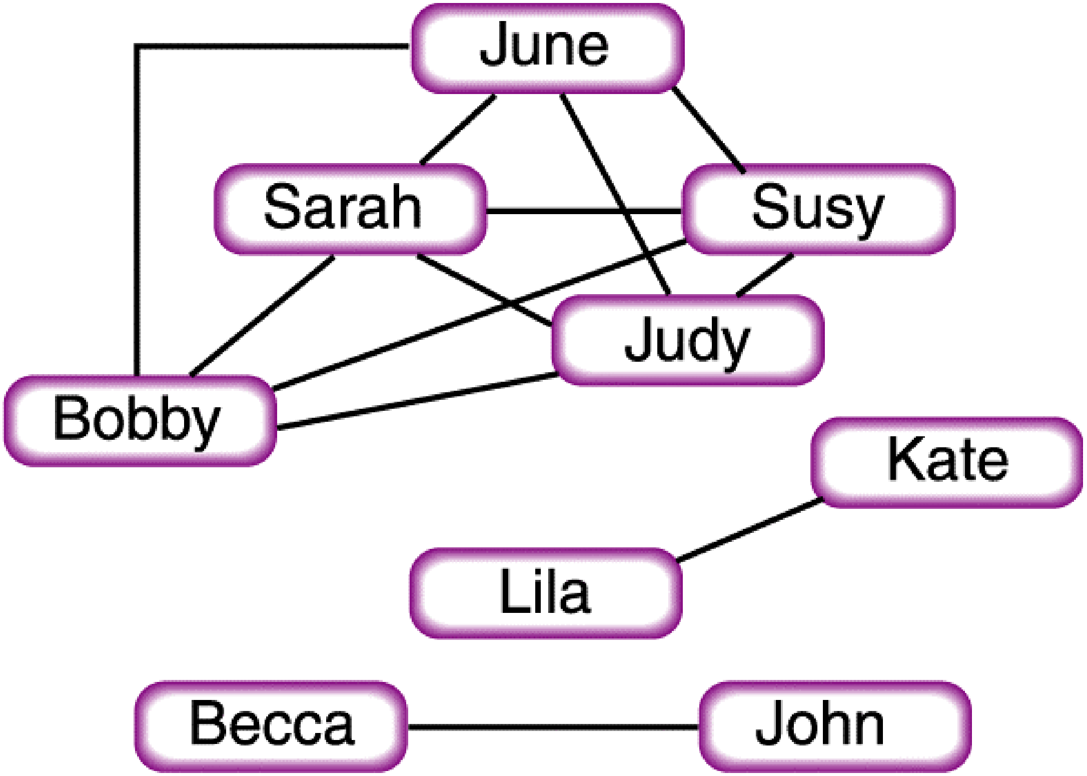
```
If (tree is null)
    Put item in tree
Else
    If (item.compareTo(info(current)) < 0)
        Insert (item, left(current))
    Else
        Insert (item, right(current))
```

Page 306

Graphs

- **Graph:** a data structure that consists of a set of nodes and a set of edges that relate the nodes to each other
- **Undirected graph:** a graph in which the edges have no direction
- **Directed graph (Digraph):** a graph in which each edge is directed from one vertex to another (or the same) vertex

Graphs

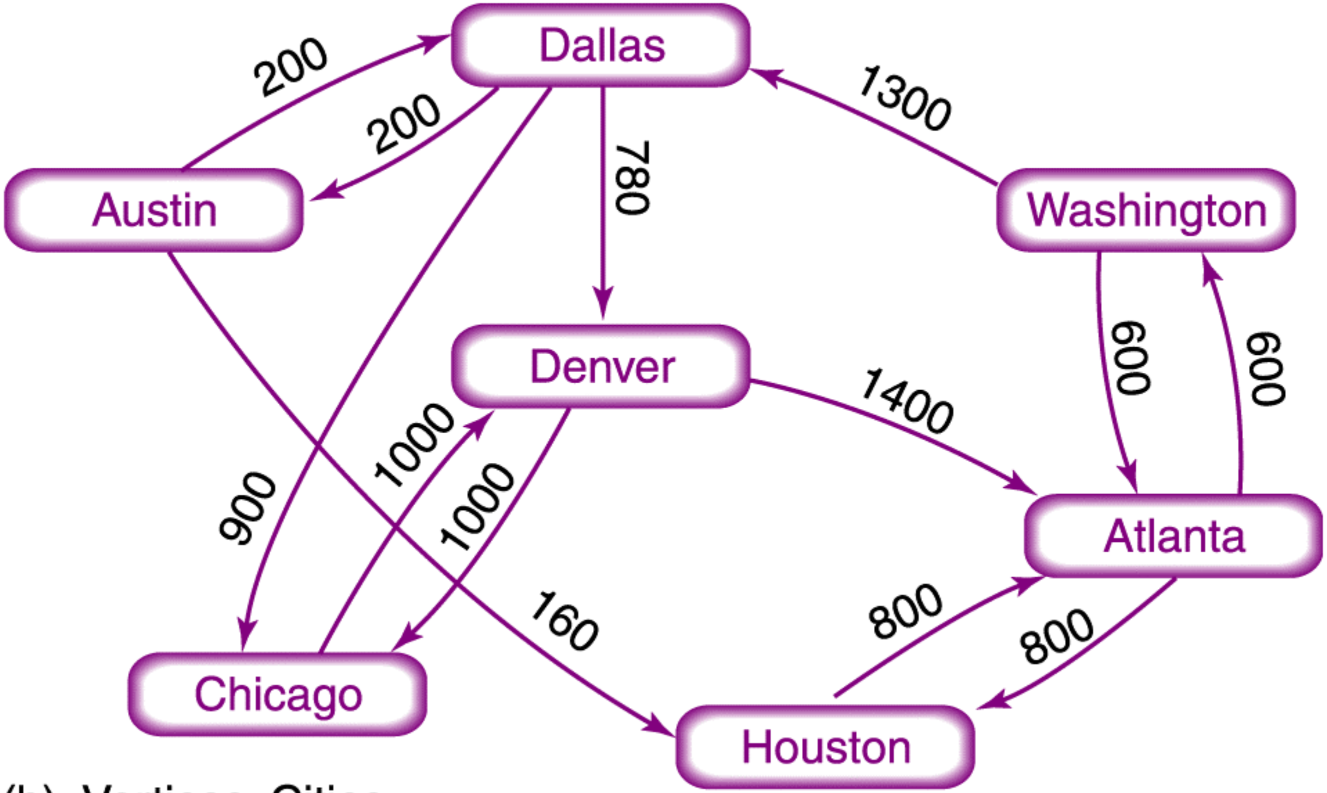


(a) Vertices: People

Edges: Siblings

Figure 9.21
Examples of graphs

Graphs

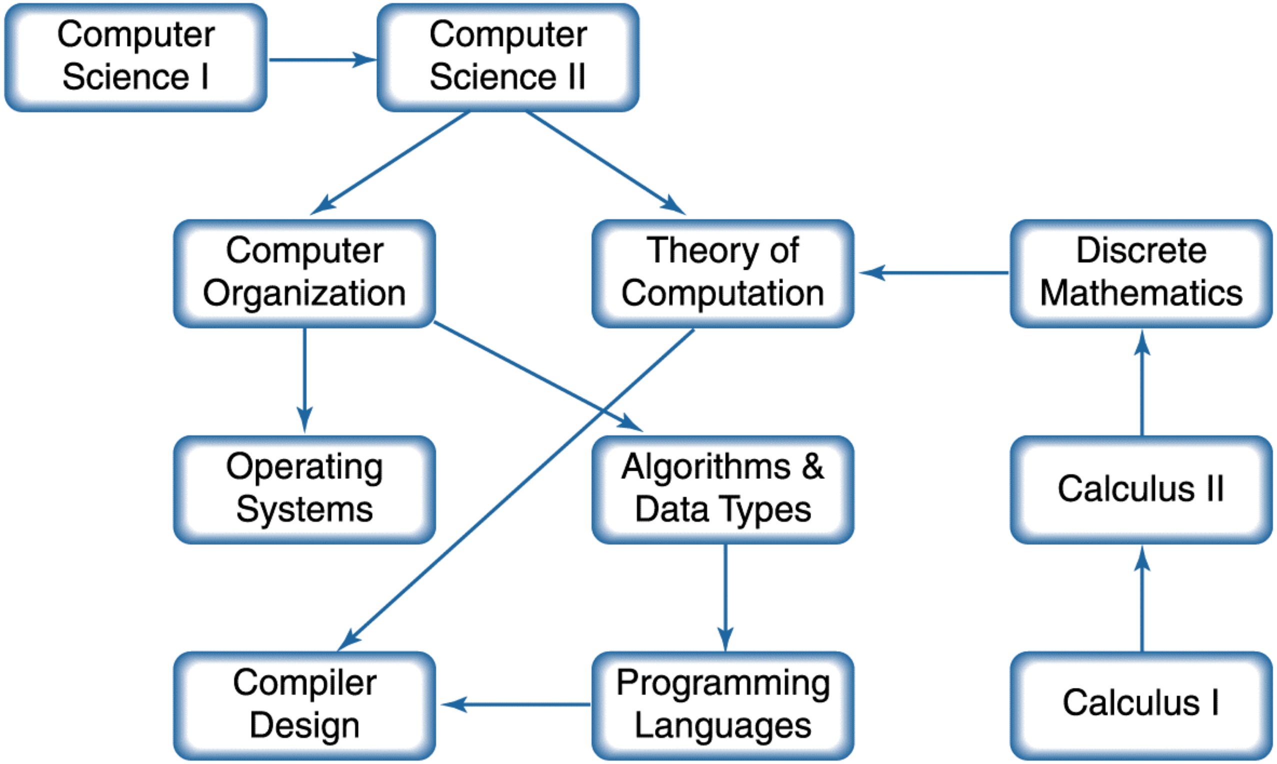


(b) Vertices: Cities

Edges: Direct Flights

Figure 9.21
Examples of graphs

Graphs



Computer Science I

Computer Science II

Computer Organization

Theory of Computation

Discrete Mathematics

Operating Systems

Algorithms & Data Types

Calculus II

Compiler Design

Programming Languages

Calculus I

(c) Vertices: Courses
Edges: Prerequisites

Figure 9.21
Examples of graphs