Final Review

cse2011

Algorithm Analysis

- Given an algorithm, compute its running time in terms of O, Ω, and Θ (if any).
  - Usually the big-Oh running time is enough.
- Given f(n) = 5n + 10, show that f(n) is O(n).
  - Find c and n_0
- Compare the growth rates of 2 functions.
- Order the growth rates of several functions.
  - Use L' Hôpital' s rule.

Running Times of Loops

Nested `for` loops:

- If the <u>exact</u> number of iterations of each loop is known, multiply the numbers of iterations of the loops.

- If the exact number of iterations of some loop is not known, “open” the loops and count the total number of iterations.

Running Time of Recursive Methods

- Could be just a hidden “for" or “while” loop.
  - See “Tail Recursion” slide.
  - “Unravel” the hidden loop to count the number of iterations.
- Logarithmic
  - Examples: binary search, exponentiation, GCD
- Solving a recurrence
  - Example: merge sort, quick sort

Recursion

Know how to write recursive functions/methods:

- Recursive call
  - Adjusting recursive parameter(s) for each call
- Base case(s)

1. Use the definition (factorial, tree depth, height)
2. Cut the problem size by half (binary search), or by *k* elements at a time (sum, reversing arrays).
3. Divide and conquer (merge sort, quick sort)

Sorting Algorithms

- Insertion sort
- Merge sort
- Quick sort
- Lower bound of sorting algorithms
  - O(NlogN)
- When to use which sorting algorithm?

Arrays and Linked Lists

Arrays

- A = B
- Cloning arrays
- Extendable arrays
- Strategies for extending arrays:
  - doubling the size
  - increment by *k* cells

Linked lists

- Singly linked
- Doubly linked
- Implementation
- Running times for insertion and deletion at the two ends.

Running Times of Array and Linked List Operations

Operation	Array unsorted	Array sorted	DL list unsorted	DL list sorted
insert	O( )	O( )	O( )	O( )
delete	O( )	O( )	O( )	O( )
search	O( )	O( )	O( )	O( )

Stacks, Queues and Deques

- Operations
- Array implementation
- Linked list implementation
- Running times for each implementation
- Assignment 2 (deques)

Trees

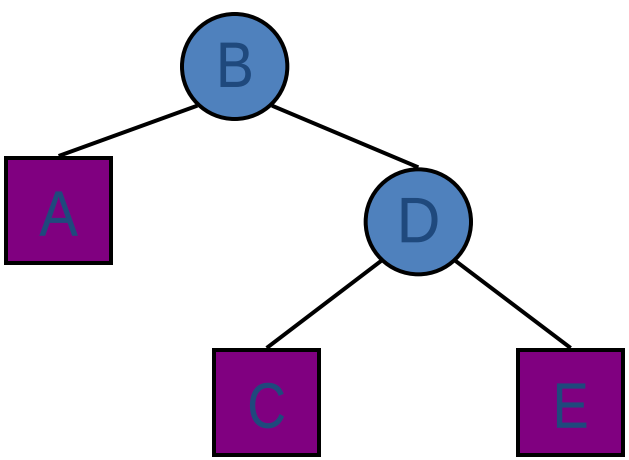
- Definitions, terminologies
- Traversal algorithms and applications
  - Preorder
  - Postorder
- Computing depth and height of a tree or node.

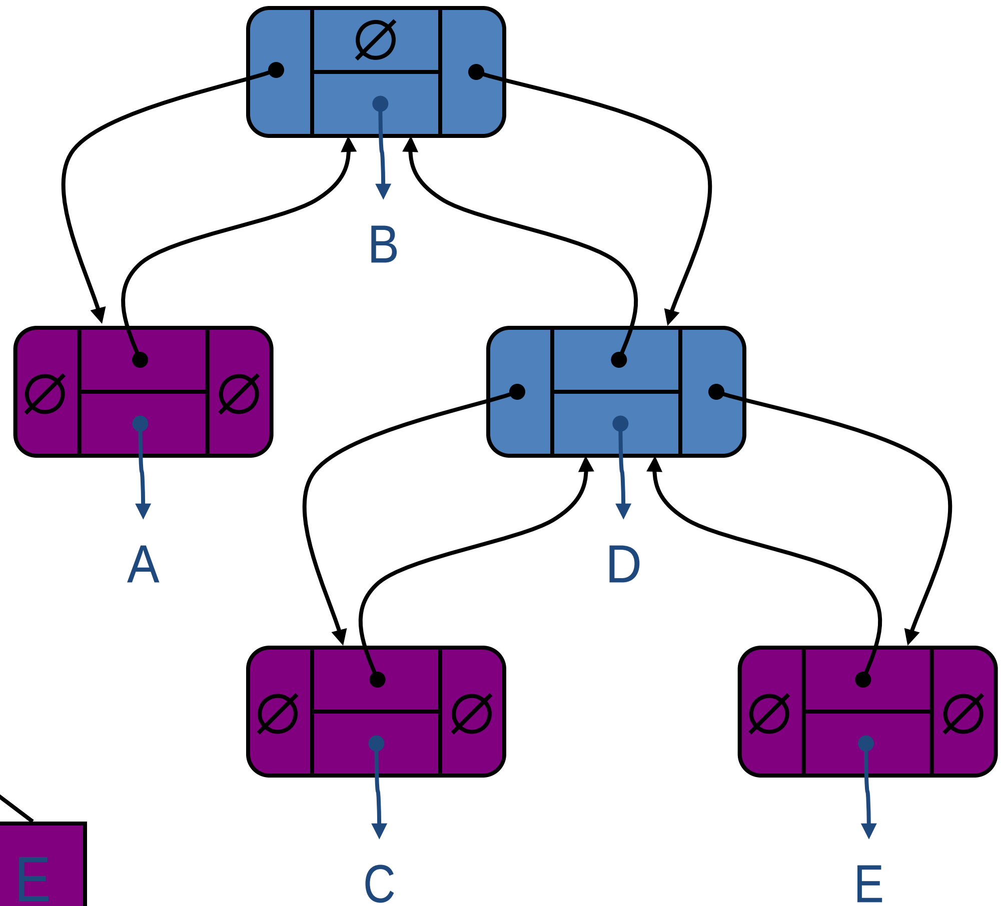
Binary Trees

- Linked structure implementation
- Array implementation
- Traversal algorithms
  - Preorder
  - Postorder
  - Inorder
- Properties: relationships between n, i, e, h.
- Definitions:
  - complete binary tree
  - full binary tree

Linked Structure of Binary Trees

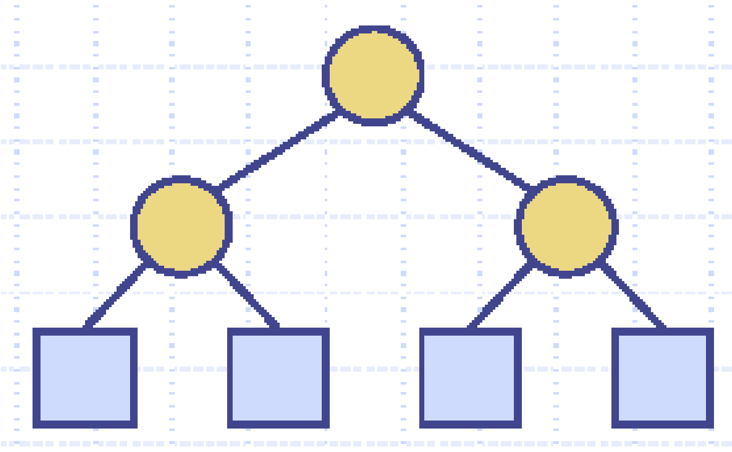
- A node is represented by an object storing
  - Element
  - Parent node
  - Left child node
  - Right child node

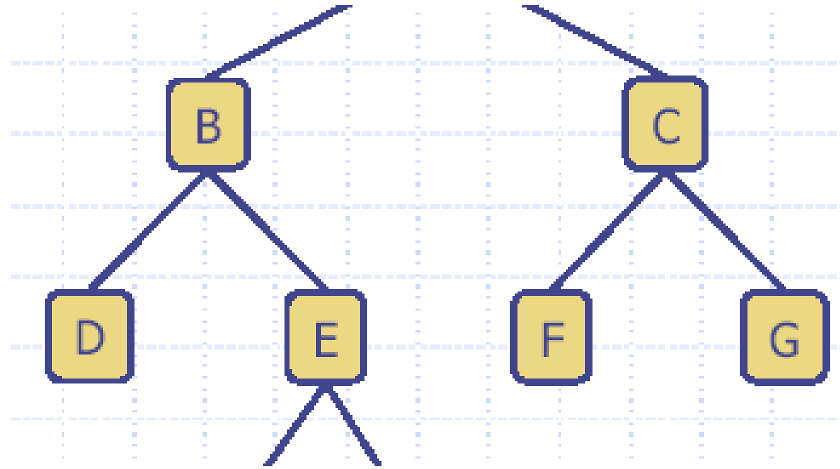




12

Array Implementation of Binary Trees





Each node *v* is stored at index *i* defined as follows:

- If *v* is the root, $i = 1$
- The left child of *v* is in position $2i$
- The right child of *v* is in position $2i + 1$
- The parent of *v* is in position ???

Space Analysis of Array Implementation

- n: number of nodes of binary tree T
- p_M: index of the rightmost leaf of the corresponding ***full*** binary tree (or size of the full tree)
- N: size of the array needed for storing T; $N = p_M + 1$

Best-case scenario: balanced, full binary tree $p_M = n$

Worst case scenario: unbalanced tree

- Height $h = n – 1$
- Size of the corresponding full tree:
  $p_M = 2^{\boldsymbol{h+1}} - 1 = 2^{\boldsymbol{n}} - 1$
- $N = 2^{\boldsymbol{n}}$

Space usage: $O(2^{\boldsymbol{n}})$

Arrays versus Linked Structure

Linked lists

- Slower operations due to pointer manipulations
- Use less space if the tree is unbalanced
- Rotation (restructuring) code is simple

Arrays

- Faster operations
- Use less space if the tree is balanced (no pointers)
- Rotation (restructuring) code is complex

Binary Search Trees and AVL Trees

BST

- Properties
- Searching
- Insertion
  - Distinct keys
  - Duplicate keys
- Deletion (3 cases)
- Running times

AVL trees

- Properties
- Searching
- Insertion: as BST plus
  - restructuring (once)
- Deletion: as BST plus
  - restructuring (maybe more than once)
- Running times

BSTs versus AVL Trees

Operation	BSTs	AVL Trees
search		
insert		
delete		
findMin		
findMax		

Implementations of Priority Queues

- Unsorted linked list
  - insertion O( )
  - deleteMin O( )
- Sorted linked list
  - insertion O( )
  - deleteMin O( )
- AVL trees
  - insertion O( )
  - deleteMin O( )
- Unsorted array
  - insertion O( )
  - deleteMin O( )
- Sorted array
  - insertion O( )
  - deleteMin O( )
- Heaps
  - insertion O( )
  - deleteMin O( )

Heaps

- Properties
- Array implementation
- Insert
  - upheap percolation
- Delete
  - downheap percolation
- Running time

- Other operations:
  - decreaseKey(i, k)
  - increaseKey(i, k)
  - delete(i)

Heap Sort

<u>Using a temp heap T</u>

```
for (i = 0; i++; i < n)
   T.insert(A[i]);
for (i = 0; i++; i < n)
   A[i] = T.deleteMin();
```

<u>In-place sorting</u>

```
run buildHeap on A;
repeat
   deleteMax;
   copyMax;
until the heap is empty;
```

Hashing

- Table size (a prime number)
- Hash functions
  - For integer keys
    - division (modular)
    - Multiple, Add and Divide (MAD)
  - For strings: polynomial accumulation
    - z = 33, 37, 39, 41

Collision handling

•Separate chaining

•Probing (open addressing)

- Linear probing
- Quadratic probing
- Double hashing

•Probing: 3 types of cells (null, in use, available)

Comparing Collision Handling Schemes

- Separate chaining:
  – simple implementation
  – faster than open addressing in general
  – using more memory

- Open addressing:
  – using less memory
  – slower than chaining in general
  – more complex removals

- Linear probing: items are clustered into contiguous runs (primary clustering).

- Quadratic probing: secondary clustering.

- Double hashing: distributes keys more uniformly than linear probing does.

Graphs

- Definitions (terminology)
- Properties (with respect to V and E)
- Data structures
  - Adjacency matrix
  - Adjacency lists
- Running time of graph methods
- Graph traversal algorithms:
  - BFS
  - DFS

Properties of Undirected Graphs

Property 1

$\Sigma_v \deg(\boldsymbol{v}) = 2\boldsymbol{E}$

Proof: each edge is counted twice

Property 2

In an undirected graph with no loops

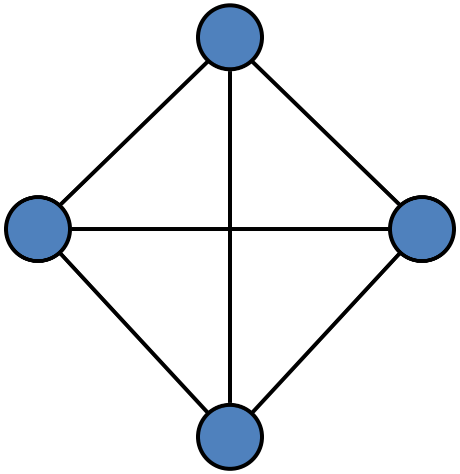
$\mathbf{E} \leq \mathbf{V}\,(\mathbf{V} - 1)/2$

Proof: each vertex has degree at most $(\mathbf{V} - 1)$

What is the bound for a directed graph?

Notation

$\boldsymbol{V}$	number of vertices
$\boldsymbol{E}$	number of edges
$\deg(\boldsymbol{v})$	degree of vertex $\boldsymbol{v}$

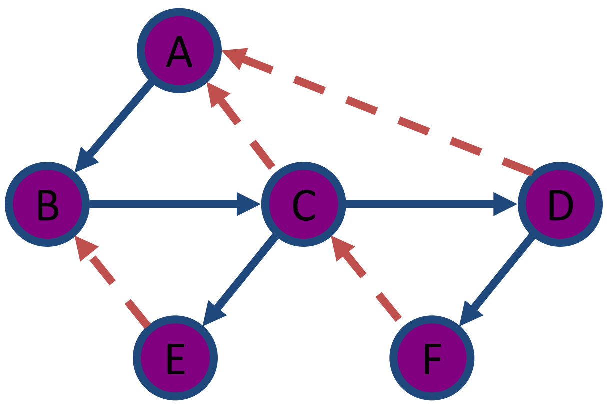


Example

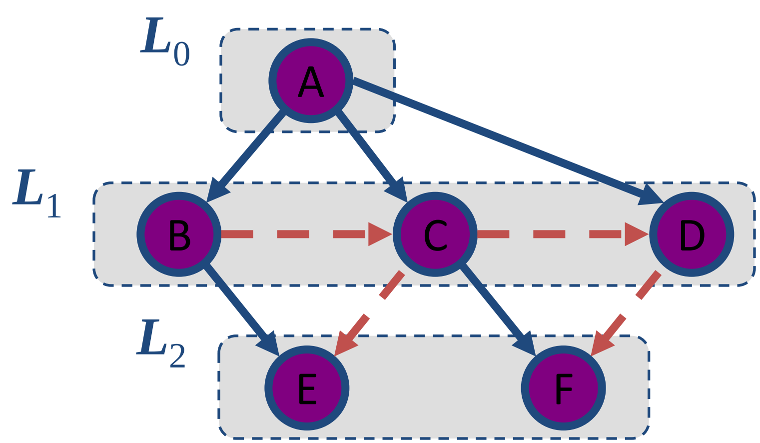
- $\boldsymbol{V} = 4$
- $\boldsymbol{E} = 6$
- $\deg(\boldsymbol{v}) = 3$

Applications of DFS and BFS

Applications	DFS	BFS
Spanning tree/forest, connected components, paths, cycles	√	√
Shortest paths		√



DFS



BFS

Final Exam

- Date: Thursday August 8th
  Time: 14:00-17:00 (3 hours)

- Materials
  – Lectures notes from the beginning to the end.
  – Corresponding sections in the textbook.

Exam Rules

- This is a closed-book exam. No books, notes or calculators are allowed. You will be given blank paper for scrap work.

- Bring a photo ID, pens, and pencils. You may use pencils with darkness of at least HB; 2B is preferred.

- You may leave the classroom if you hand in your exam booklet 15 minutes or more before the exam ends.

Exam Rules (2)

- Programming problems will be marked based on both correctness and efficiency.

- You may use either Java or pseudo-codes if you are asked to write the code.

Office Hours before Exam

- Wednesday Aug 7th, 14:00-16:00.
- Email, anytime!