Lecture 12: More Data Structures

BT 3051 – Data Structures and Algorithms for Biology

Karthik Raman
Department of Biotechnology
Indian Institute of Technology Madras

September 8, 2014

Queue

- Queues can be implemented using an array or linked list

Operations

- `create()`
- `delete()`
- `isEmpty()`
- `length()`
- `enqueue()`
- `dequeue()`
- `front()`

Array-based Queue implementation

- How will you implement a queue using an array?
- Can the same stack idea work?
- Self-assessment Exercise: Implement `ArrayQueue` class

Linked Lists

- A singly linked list, in its simplest form, is a collection of nodes that collectively form a linear sequence
- Each node stores a reference to an object that is an element of the sequence, as well as a reference to the next node of the list
- Sedgewick's definition: *A linked list is a recursive data structure that is either empty (null) or a reference to a node having a generic item and a reference to a linked list.*

Linked Lists

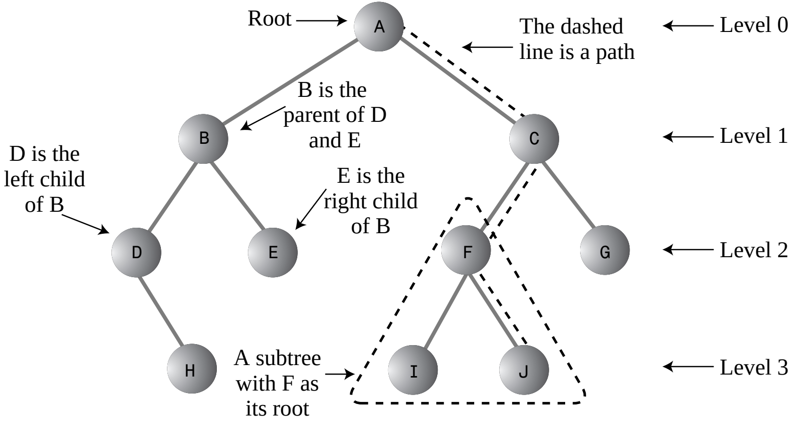
Efficiency

- Insertion and deletion at the beginning of a linked list are very fast
  - involve changing only one or two pointers, which takes $O(1)$ time
- Finding or deleting a specified item requires searching through, on the average, half the items in the list — O(N) comparisons
- An array is also $O(N)$ for these operations, but the linked list is nevertheless faster because nothing needs to be moved when an item is inserted or removed
- The increased efficiency can be significant, especially if a copy takes much longer than a comparison
- Linked list uses exactly as much memory as it needs, and can expand to fill all available memory
- Even use of memory by dynamically resizing arrays is still not as efficient as a linked list

Linked List
Design Questions

- When is a linked list preferred over an array?
- When is an array preferred over a linked list?

Trees



H, E, I, J, and G are leaf nodes

Tree Jargon

- Node
- Root
- Parent
- Child, [Left Child, Right Child]
- Leaf
- Subtree
- Visiting (a node)
- Traversing (all nodes)
- Keys
- Binary Tree
- Balanced Tree

Implementing a Binary Tree

Figure 8.1 from *Data Structures and Algorithms in Python*