Lecture 5: Binary Search, Binary Search Trees (BST)

Instructor: Saravanan Thirumuruganathan

Outline

1. Search problem - Linear and Binary Search
2. Data Structures for representing Dynamic Sets
    - Unsorted list
    - Sorted list
    - Binary Search Trees (BSTs)

In-Class Quizzes

- **URL:** `http://m.socrative.com/`
- **Room Name: 4f2bb99e**

Search Problem

- **Input:** Set A with n numbers and an element e
- **Output:** First index of e in A
- **Example:** $A = \langle 3, 2, 4, 1, 6, 8, 11 \rangle$, $e = 4$. Output=3

Linear Search

- Also called as Sequential search
- Idea: Examine each element in A one by one from start to finish
- Always works - whether A is sorted or not
- Analysis:

Linear Search

- Also called as Sequential search
- Idea: Examine each element in A one by one from start to finish
- Always works - whether A is sorted or not
- Analysis:
  - Complexity Measure: Number of comparisons
  - Time Complexity: $O(n)$

Searching in a Sorted Set

- If the application is search intensive, then sorting is a good idea
- If you do linear Search of A for n times then it requires $O(n^2)$ time
- We can do better!

Binary Search Intuition

- Intuition 1: Searching for a word in a dictionary

Binary Search Intuition

- Intuition 1: Searching for a word in a dictionary
  - Check the middle of the book
  - Proceed in one direction based on the middle page
  - Recurse
- Intuition 2: Guessing game

Binary Search Intuition

- Intuition 1: Searching for a word in a dictionary
  - Check the middle of the book
  - Proceed in one direction based on the middle page
  - Recurse
- Intuition 2: Guessing game
  - Your friend wants thinks of a number between 1 and n
  - You have to find it in least number of guesses
  - When you make a guess, your friend tells *Yes*, *Lower* or *Higher*
  - Use the information to cut the search space

Binary Search

- *Very* popular searching technique
- Based on D&C technique
- At each step, cut your search space by half
- **High Level Idea:**
  - Get midpoint of range (aka search space)
  - Determine which half contains the data
  - Search that half recursively using Binary search

Binary Search

```
BinarySearch(A, e, low, high):
    if low > high
        return Not found
    else
        mid = (low + high) / 2
        if e == A[mid]
            return mid
        else if e < A[mid]
            return BinarySearch(A, e, low, mid - 1)
        else
            return BinarySearch(A, e, mid+1, high)
```

Binary Search Analysis

- **Analysis:** Given a set with n elements - at each iteration,
  - You do one comparison
  - Recursively call Binary Search with $n/2$ elements
- $n \rightarrow \frac{n}{2} \rightarrow \frac{n}{4} \rightarrow \ldots \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$

Binary Search Analysis

- **Analysis:** Given a set with n elements - at each iteration,
  - You do one comparison
  - Recursively call Binary Search with $n/2$ elements
- $n \rightarrow \frac{n}{2} \rightarrow \frac{n}{4} \rightarrow \ldots \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$
- Requires at the most $\lceil \lg n \rceil$ comparisons
- Time Complexity: $O(\lg n)$

Practical Issues

- Despite simplicity, very hard to get the implementation right!
- Bentley
  - Bell Labs story - only 10% of engineers got it right after 2 hours!
  - Java story - If interested, read `http://googleresearch.blogspot.com/2006/06/extra-extra-read-all-about-it-nearly.html`
  - Issues found in C, C++, Java etc
  - If interested, read (from UTA network) `http://comjnl.oxfordjournals.org/content/26/2/154.full.pdf`
- Moral of the story: Don't implement it yourself - Always use from language library!

Data Structures

Dynamic Sets

- **Set:** A set is a collection of distinct objects
- **Dynamic Set:** A set that changes over time (grow or shrink)
- **Objective:** Design an efficient data structure to represent a dynamic set

Operations on Dynamic Sets

- Search(S, k)
- Insert(S, x)
- Delete(S, x)
- Minimum(S)
- Maximum(S)
- Successor(S, x)
- Predecessor(S, x)

Dictionary:

- Insert
- Delete
- Test membership (Search)

Elements of Dynamic Set

- **Key:** A set of attributes that identify an object
  - Only Key is used by set maintenance algorithms
- **Satellite Information:** Auxiliary information about the object not used by the algorithms
  - Not used by set maintenance algorithms

Linear Data Structures:

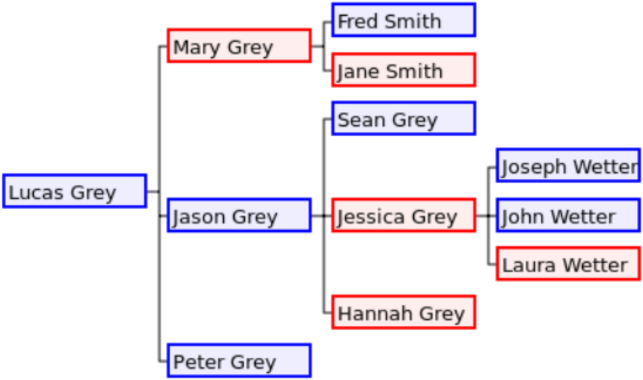
- Stacks
- Queues
- Linked Lists

Non-Linear Data Structures:

- Very common and useful category of data structures
- Most popular one is **hierarchical**

Trees - Applications[1]

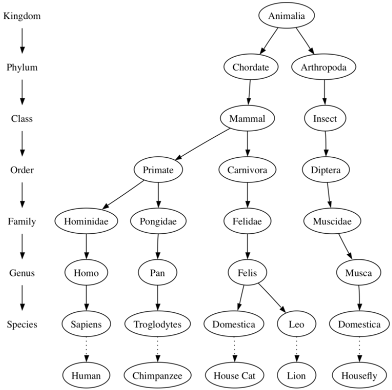
Family Tree:



[1] `http://interactivepython.org/runestone/static/pythonds/Trees/trees.html`

Trees - Applications[2]

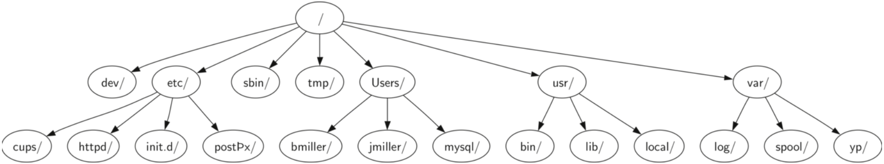
Taxonomy Tree:



[2] `http://interactivepython.org/runestone/static/pythonds/Trees/trees.html`

Trees - Applications[3]

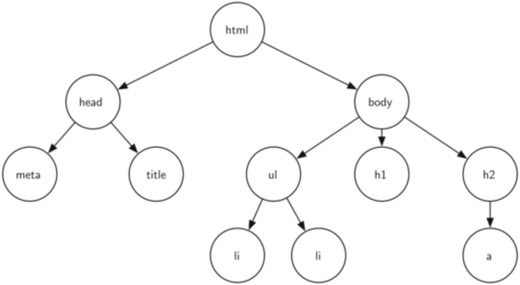
Directory Tree:



[3] http://interactivepython.org/runestone/static/pythonds/Trees/trees.html

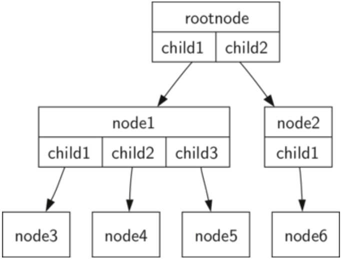
Trees - Applications[4]

HTML DOM (Parse) Tree:



[4] http://interactivepython.org/runestone/static/pythonds/Trees/trees.html

Tree - Abstract Representation[5]



[5] http://interactivepython.org/runestone/static/pythonds/Trees/trees.html

Tree - Terminology

- Node
- Edge
- Root
- Children
- Parent
- Sibling
- Subtree
- Leaf node (or external node)
- Internal node
- Level of a node (level(root) $= 0$)
- Height of tree

Summary

Major Concepts:

- Search Problem
- Linear and Binary Search algorithms
- Data Structures for Dynamic Sets
- BSTs