Binary search

Goldsmiths Computing

November 24, 2018

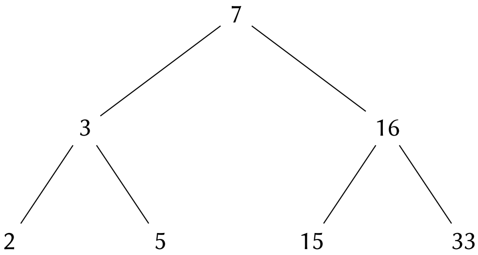
Motivation

- simple, efficient search algorithm
- one or two intersting practical lessons

Definition

Given a suitable data structure, binary search is a search algorithm for an item within that structure that can exclude half of the search space with a single comparison.

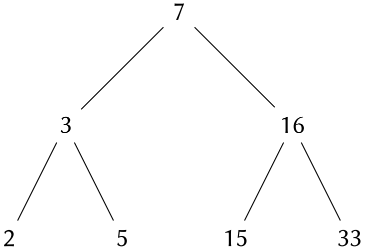
Tree representation



Binary search on trees

function BINARY-SEARCH(tree,k)
    if tree = NIL **then**
        return false
    else if tree.key = k **then**
        return true
    else if k < tree.key **then**
        return BINARY-SEARCH(tree.left,k)
    else
        return BINARY-SEARCH(tree.right,k)
    end if
end function

Sorted array (implicit tree) representation



2	3	5	7	15	16	33

Binary search on sorted arrays

function BINARY-SEARCH(A,lo,hi,k)
    mid $\leftarrow \left\lfloor \frac{lo+hi-1}{2} \right\rfloor$
    if lo = hi **then**
        return false
    else if A[mid] = k **then**
        return true
    else if k < A[mid] **then**
        return BINARY-SEARCH(A,lo,mid,k)
    else
        return BINARY-SEARCH(A,mid+1,hi,k)
    end if
end function

Complexity analysis

Recurrence relationship

$$T(N) = T\left(\frac{N}{2}\right) + 1$$

Recursion tree

$$T(N)$$

Complexity analysis

Recurrence relationship

$$T(N) = T\left(\frac{N}{2}\right) + 1$$

Recursion tree

$$\begin{array}{c} 1 \\ | \\ T\left(\frac{N}{2}\right) \end{array}$$

Complexity analysis

Recurrence relationship

$$T(N) = T\left(\frac{N}{2}\right) + 1$$

Recursion tree

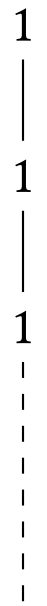
$$\begin{array}{c} 1 \\ | \\ 1 \\ | \\ T\left(\frac{N}{4}\right) \end{array}$$

Complexity analysis

Recurrence relationship

$$T(N) = T\left(\frac{N}{2}\right) + 1$$

Recursion tree



Complexity analysis

Recurrence relationship

$$T(N) = T\left(\frac{N}{2}\right) + 1$$

Recursion tree

$$\left.\begin{array}{c}1\\ |\\ 1\\ |\\ 1\\ \vdots\end{array}\right\}\log_2 N$$

Complexity analysis

Recurrence relationship

$$T(N) = T\left(\frac{N}{2}\right) + 1$$

Master theorem

$$T(N) = aT\left(\frac{N}{b}\right) + f(n)$$

- a = 1; b = 2; $f(n) \in \Theta(1) = \Theta(n^0)$ so c = 0
- $\log_b a = 0 =$ c so case 2

$$\implies \Theta(\log N)$$

Work

1. as written in these slides, the algorithm binary search on sorted arrays contains a trap for the unwary: it is mathematically correct, but if translated directly into Java or C++ it would cause problems.
   - Reading: Jon Bentley, *Programming Pearls*, Column 4: Writing Correct Programs
   - Bentley's implementation of binary search in the above column has (at least) one serious bug
2. (week of 21st January) implement binary search (correctly!)