Week 9 (Notes)

Searching

Search is a common task in computing.

- Sequential Search
- Binary Search
- Fibonacci Search
- Tree Search
- Hashing

Sequential Search

Assume k is an array of *n* records. We want to find the first i such that k[i] =KEY. If search is successful return i else -1.

```
for ( i = 0; i <= n, i++)
        if (KEY == k(i))
                return (i);
return (-1);
```

Complexity:

Binary Search

- Given a sorted array of numbers, a.
  Range: low = 1b, high = ub
- Want to find out if x is in the array, a, in the index range low to high.

Binary Search Process:

look at middle [= (low + high) div 2] element
if x = a[middle] then return middle

else if x < a [middle] then
look for x in the range
low = low
high = middle - 1

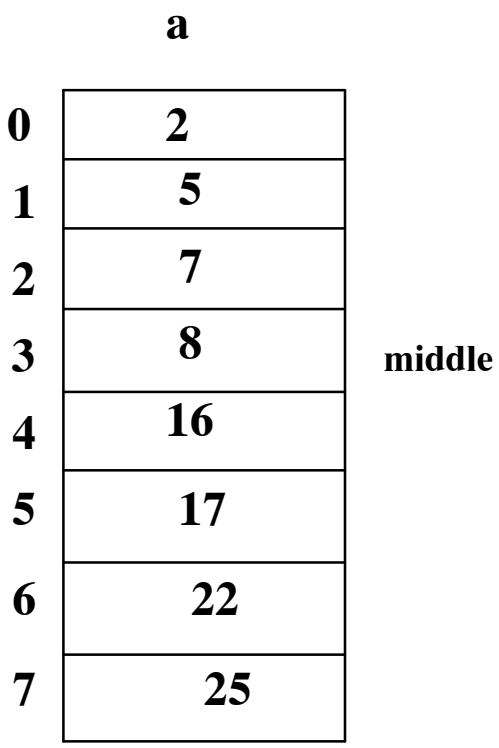
else x > a[middle], therefore,
look for x in the
range: low = middle + 1
high = high.

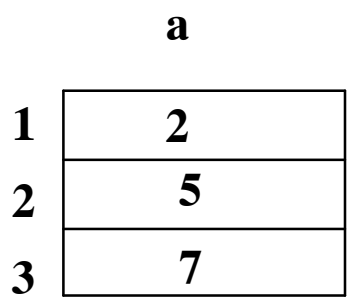
if not there -- return NIL

termination condition?

Some Analysis

How many calls are necessary to find out if *x* is present in an array of *n* elements ?





Fibonacci Search
(Binary Decision Tree)

Fibonacci Sequence: 0, 1, 1, 2, 3, 5, 8, 13, ...

Recursive Definition:

$$F(k) = F(k-1) + F(k-2) \qquad k > 1$$

Fibonacci Search

Binary Search Requires Division by 2

In Fibonacci Search, split a search interval of length F(k+1) into two sub-intervals of length F(k) and F(k-1). No division is needed, only subtraction. Use a Fibonacci Tree.

Fibonacci Tree

Given an array of N elements, assuming
N = F(k+1) – 1, we need a tree:
- The values of nodes represent index in the array.
- It is a Fibonacci tree of order k.

The tree is generated as follows:

Select the root node value F(k), $k \geq 2$.

It has left subtree which is Fibonacci tree of order F(k–1), and right subtree which is Fibonacci of order F(k–2), with all the numbers increased by F(k). This property is true for every subtree.

Values of the both children differ from the parent by the same amount.

Example

An important observation: The left most path represents the Fibonacci sequence, that starts as 1, 2, 3 .... and ends at the root having value F(k).

Fibonacci Search Algorithm
(Binary Decision Tree)

Similar to the one used for searching a binary tree, except use the node values as index of the arrray. Search terminates if the value of the child is one less than the value of the parent.

Fibonacci Search Algorithm
(Binary Decision Tree)

Assume number of records N = F(k+1) - 1, Sorted in an ascending order. Search for K.

1. Set i = F(k), p = F(k-1), q = F(k-2)
i = 21, p =13, q = 8

2. If K < K(i) Then (If K =K(i) ---> successful)

    If q = 0 ----> unsuccessful,
    otherwise i = i - q,
    (p , q) = (q , p-q)
    Go to 2

3. Else

    If p = 1 ----> unsuccessful,
    otherwise i = i + q,
    p = p-q and then q = q-p
    Go to 2

Binary Search Tree

Organization of Input Data or Array as a Binary Tree

Definition: A binary tree with left descendents having values less than their ancestors and right descendents have values greater than ancestors. Values may be distinct.

Used for Searching

Inorder traversal gives an ascending order listing of values. (Binary Tree Sort)

Generation of Binary Search Tree

Compare the new element to be inserted with the root node. If its value is greater than or equal to root, move towards right subtree, otherwise move towards left subtree.

Repeat the process with root of the subtree, until a leaf node is encountered. Insert the data element as one of the child, depending upon the comparison.

If no duplication is required, then search before insertion.

Example

14, 15, 4, 9, 7, 18, 3, 5, 16, 4, 20, 17, 9, 14, 5

< to left

>= to right

Note: Inorder traversal

Example

Input Data: 1, 2, 3, 4, 5, 6...

Note: A nicer binary tree is generated if data is not sorted.

This is like a list!

Binary Tree Search

To search for a key, traverse tree by comparing keys until a key matched is found or there is no match possible (fall off tree!)

Deletion from a Binary Search Tree

Find the node with search-key. Three cases to consider:

(1) node has no children - no adjustment needed

(2) if node has only 1 child, move that child to take its place.

(3) if node has 2 children then its inorder successor, S, must take its place.
(Note: S is either the largest element in its left subtree or the smallest one in its right subtree).
Proceed to perform the deletion algorithm to replace S.

Efficiency of Binary Search Tree

- Time to search a binary tree is:

- If records are inserted in random order, a balanced tree results more often. So average search time is:

Balanced Binary Trees (AVL)

One way to get around the worst case behavior of binary tree search is to ensure that the tree is nearly balanced.

Balanced trees yield the most efficient search given that the probability for any key is the same for all keys.

Note: Insert & Deletion routines do note ensure a balanced tree.

Definition: An AVL tree is a binary tree in which the heights of the two subtrees of every node never differs by more than 1. (Balanced binary tree)