UNIT-2

 U2. 1

Learning Objective

- Sorting and Searching
- Hashing
- BST

 U2. 2

Insertion Sort

Insertion Sort

- Insertion sort keeps making the left side of the array sorted until the whole array is sorted.
- Basic Algorithm:
  - Repeat the following steps until value in proper position
    - ✓Step 1. Pull out current value into a temp variable
    - ✓Step 2. Check index's value - if less than temp then move it left.
- It is the simplest of all sorting algorithms.

 U2. 3

Insertion Sort Cont…

Sort: 34 8 64 51 32 21

- 34 8 64 51 32 21
  - Pull out 8 into Temp
- 34 8 64 51 32 21
  - Compare 34 and 8 - move 34 up a spot
- 34 34 64 51 32 21
  - Spot is found for 8 - place it where it belongs
- 8 34 64 51 32 21

Insertion Sort Cont…

Sort: 34 8 64 51 32 21

- 8 34 64 51 32 21
  - Pull out 64 into Temp
- 8 34 64 51 32 21
  - Compare 64 and 34 - place 64 back into slot 2
- 8 34 64 51 32 21

Insertion Sort Cont…

Sort: 34 8 64 51 32 21

- 8 34 64 51 32 21
  - Pull out 51 into Temp
- 8 34 64 51 32 21
  - Compare 51 and 64 - move 64 to the right
- 8 34 64 64 32 21
  - Compare 51 and 34 - place 51 into slot 2
- 8 34 51 64 32 21

Insertion Sort Cont…

Sort: 34 8 64 51 32 21

- 8 34 51 64 32 21
  - Pull out 32 into Temp
- 8 34 51 64 32 21
  - Compare 32 and 64 - move 64 to the right
- 8 34 51 64 64 21
  - Compare 32 and 51 - move 51 to the right
- 8 34 51 51 64 21
  - Compare 32 and 34 - move 34 to the right
- 8 34 34 51 64 21
  - Compare 32 and 8 - place 32 in slot 1
- 8 32 34 51 64 21

What comes next?

Sequential Search

- (Unordered)

```
Get the search criterion (key)
Get the first element
While ( (element != key) and (still more elements) )
   Get the next element
End_while
```


Sequential Search Cont..

- Sequential Search(ordered)

Basic algorithm:

```
Get the search criterion (key)
Get the first element
While ( (element < key) and (still more elements) )
   Get the next element
End_while
If ( element = key )
   Then success
   Else there is no match in the file
End_else
```


Binary Search

- Binary Search
  - Binary search algorithm assumes that the items in the array being searched are **sorted**
  - The algorithm **begins at the middle** of the array in a binary search
  - If the item for which we are searching **is less than the item in the middle**, we know that the item won't be in the second half of the array
  - **Once again** we examine the "middle" element
  - The process continues with each comparison cutting in half the portion of the array where the item might be

Balanced Search Trees

Balanced Search trees

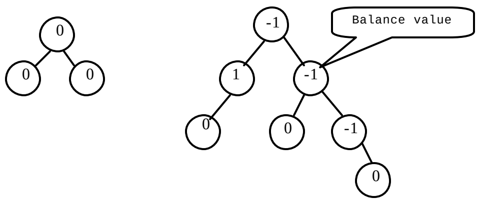
- AVL trees and Red-black trees balance a binary search tree so it more nearly resembles a complete tree.

Balanced Search Trees Cont…

AVL Trees.

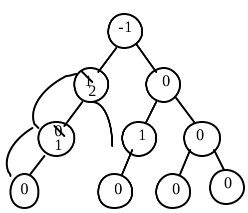
- a BST where each node has a **balance factor**
  - balance factor of a leaf node is 0
  - balance factor of a node:
    - ✓height of left subtree - height of right subtree
- insertions or deletions change the balance factor of one or more nodes
- if a balance factor becomes 2 or -2 the AVL tree must be rebalanced
  - done by rotating nodes

Balanced Search Trees Cont…

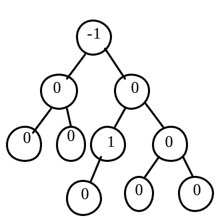


balance is height(left subtree) - height(right subtree)

Balanced Search Trees Cont…



Balanced Search Trees Cont…



Balanced Search Trees Cont…

Red Black Tree(RB Tree):

A red black tree is an extended binary search tree which meets the following red black properties.

1. Every node is either red or black.
2. No, red node can have a red child.
3. A simple path from any node to a descendent leaf always contain the same number of black nodes (the number of black nodes which appear in this simple path from the root node is termed as the black height of the tree).
4. The root node is always black.

Balanced Search Trees Cont…

If uncle is red
{
Set uncle to black
Set parent to black
Set grand parent to red
Set x to grand parent
}

Hash Table(Hashing)

- Hashing is used when the universe of keys is very large but at any given point of time very small number of keys is to be stored.
- Under this technique a Data structure known as the **Hash Table** is taken. An arithmetic function termed as a hash function is used to translate a given key to its index in the hash table.

A hash function is chosen keeping in mind the following three factors:

1. Size of the hash table.
2. It must ensure a uniform distribution
3. It should ensure minimal collisions.

Hash Table(Hashing) Cont…

Hash Functions:

1. Division method:

$h(k)=k \bmod m$

Where m is the size of hash table.

2. Mid square method.

$h(k)=$ Any sequence of digits from the mid k^2.

The length of the sequence is determined by hash table

3. Folding method.

The index is obtain by breaking the given key into equal fragments, adding the fragment and ignoring the last carry(the size of each fragment is determined by the size of the hash table).

Hash Table(Hashing) Cont…

4.Multiplication Method:

$h(k)=(kA \bmod 1) \bmod m$

Here

A- is a constant

$0<A<1$ and is taken to ensure a uniform distribution of the keys across the hash table.

Data structures for Disjoint Sets

Disjoint Set:

- A disjoint set data structure maintains a collection S={s1,s2,…..sk} of disjoint dynamic sets. Each set is identified by a representative, which is some members of the set.
- In dynamic implementations each element of a set is represented by an object.

Data structures for Disjoint Sets Cont..

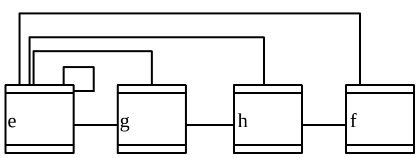
- **Linked-List representation of Disjoint Sets**

  We can give linked list implementation graphically by pointing one node representing one vertex to another and each node pointing back to first node which is representation.

Data structures for Disjoint Sets Cont..

b={e,g,h,f}

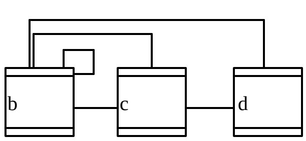
(b)



Data structures for Disjoint Sets Cont..

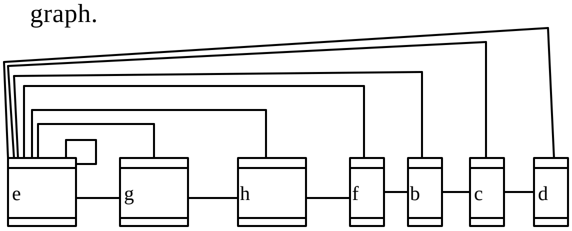
a={b,c,d}

(a)



Data structures for Disjoint Sets Cont..

When Union (b,a) is performed all nodes points to the first node of the representation of the first graph.



Divide and Conquer Paradigm of Problem Solving

- A divide-and-conquer algorithm divides the problem instance into a number of subinstances (in most cases 2), recursively solves each subinstance separately, and then combines the solutions to the subinstances to obtain the solution to the original problem instance.

Divide and Conquer Paradigm Cont…

In terms or algorithms, this method has three distinct steps:

- **Divide**: If the input size is too large to deal with in a straightforward manner, divide the data into two or more disjoint subsets.
- **Recursion**: Use divide and conquer to solve the sub-problems associated with the data subsets.
- **Conquer**: Take the solutions to the sub-problems and "merge" these solutions into a solution for the original problem.

Complexity Analysis

- The subject of the analysis of algorithms consists of the study of their efficiency.

  Two aspects of the algorithm efficiency are:

  1. The amount of time required to execute the algorithm
  2. The memory space it consumes.

Complexity Analysis Cont...

- The following main operations are used in the algorithms
  Assignment(?)
  Comparison(=, ?, <, >, =, =)
  Arithmetic operation(+, -, ×, ÷)
  Logical operations(and, or, not)

- How long does each of these operations take to execute?
  In general, assignments are very fast ,while other operations are slower. Multiplication and division are often slower than addition and subtraction.

Complexity Analysis Cont...

Time Complexity

•The running time of an algorithm is defined to be an estimate of the number of operations performed by it given a particular number of input values.

•Let *T(n) be a measure* of the time required to execute an algorithm of problem size *n. We call T(n) the time complexity* function of the algorithm.

•If *n is sufficiently* small then the algorithm will not have a long running time.

Complexity Analysis Cont...

- **Note:**

In our time analysis we will restrict ourselves to the worst case behavior of an algorithm; that is, the longest running time for any input of size n.

Complexity Analysis Cont...

Example

What is the run-time complexity based on *n for the* following algorithm segment:

1. **for** i = 1 to n **do**
   1.1 **for** j = 1 to n **do**
      1.1.1 A(i,j) ? x

Complexity Analysis Cont...

Solution The inner loop 1.1 is executed n times and the outer loop 1. is also executed n times. Hence, $T(n) = n^2$ *so that the growth is of order* n^2.

Complexity Analysis Cont...

Estimate the time complexity of the following algorithm:

1. i ? 1
2. p ? 1
3. **for** j = 1 to n **do**
   - 3.1 p ? p × i
   - 3.2 i ? i + 1

Complexity Analysis Cont...

- **Solution** It takes two assignment statements 1. and 2. to initialize the variables i *and* p.
- The loop 3. is executed n times, and each time it executes two assignment statements and two arithmetic operations 3.1 and 3.2.
- Thus, the time complexity of the algorithm is given by
  T(n) = 4n + 2 so the growth is of order *n*.

Binary Search Tree

- A binary search tree (BST), which may sometimes also be called an ordered or sorted binary tree.
- It is a node-based binary tree data structure which has the following properties:

1. The left subtree of a node contains only nodes with keys less than the node's key.
2. The right subtree of a node contains only nodes with keys greater than the node's key.
3. Both the left and right subtrees must also be binary search trees.

Binary Search Tree Cont…

1. Full binary Tree or Strictly Binary Tree:
   A Full binary tree is a binary tree where each parent node has a degree of two.
2. Complete binary tree:
   It is a full binary tree in which all the leaf nodes appear at the same level.
3. Almost complete binary tree:
   It is a binary tree in which if the height of the tree is 'h' than till 'h-1' height it is a complete binary tree.
4. Extended binary tree or 2-tree:
   An Extended binary tree is a binary tree in which special square nodes are drawn to make each nodes having a degree 2. Such square nodes are termed as the External nodes while the original nodes of the binary tree are termed as the internal nodes.

Statistics

Medians and Order statistics

Medians and Order Statistics

Order Statistics:

- ith order statistics of a set of n elements is ith smallest element.
- the minimum of a set of elements is the first order statistic(i=1)
- And the maximum is the nth order statistic (i=n).

Medians and Order Statistics Cont..

For example

Let Y is a set of 6 element.

n=6

Y={3,4,6,2,1,8}

ith order statistics of set Y is 5th

Because at the fifth place the element is smallest i.e.1

Medians and Order Statistics Cont..

Medians

- It is the middle element in the set of 'n' elements such that if 'n' is odd median is
- [(n+1)/2]th element, other wise if n is even then middle element lies at [n/2]th and [n/2+1]th elements.

Medians and Order Statistics Cont..

Q. Write an algorithm MEDIAN(S) to get the median element from the sequences of n elements Using ceiling and floor?

Conclusion

- The greedy method suggests that one can devise an algorithm that works in stages, considering one input at a time. At each stage, a decision is made regarding whether a particular input is in an optimal solution.
- Dynamic Programming is an algorithm design method that can be used when the solution to a problem can be viewed as the result of a sequence of decisions.

 U2.43

Review Questions

1. Optimal BST is the one which has ________ search cost.
2. Elements of Dynamic Programming are ________, _____________, _________.
3. Huffman Codes is an example of __________ codes.
4. The tasks whose deadline is over and they are not been executed yet are know as_________.
5. The main goal of Matrix Chain multiplication problem is only to determine an order for multiplying matrices that has the _____________.

 U2.44

Review Questions Cont...

6.The complexity of linear search algorithm is

a. O(n)
b. O(log n)
c. O(n2)
d. O(n log n)

7.The complexity of Binary search algorithm is

a. O(n)
b. O(log)
c. O(n2)
d. O(n log n)

 U2.45

Review Questions Cont...

8. The complexity of Bubble sort algorithm is
a. O(n)
b. O(log n)
c. O(n2)
d.O(n log n)

9.The complexity of merge sort algorithm is
a. O(n)
b. O(log n)
c. O(n2)
d. O(n log n)

Review Questions Cont...

10.The complexity of the average case of an algorithm is
a. Much more complicated to analyze than that of worst case
b. Much more simpler to analyze than that of worst case
c. Sometimes more complicated and some other times simpler than that of worst case
d. None or above

Review Questions Cont...

1. Explain Huffman codes and fixed length codes
2. What do you mean by activity selection problem explain with the help of an example?
3. Write short notes on Memoization
4. Explain Optimal Binary Search tree with the help of an example.
5. What the elements of Dynamic Programming?

Review Questions Cont...

6. Prove that time complexity of binary search is logn
7. Explain various hashing functions with the help of an example. Also explain what you can do in the event of collision in hash table.
8. Explain Divide and conquer Paradigm of Problem solving with the help of an example.
9. w.r.t BST Prove that E=I+2N.
10. what is Quick sort? Explain with the help of Example?

Review Questions Cont..

1. Explain Matrix Chain Multiplication with the help of an example.
2. What do you mean by a task scheduling algorithm? Also define late and early tasks.
3. Find the longest common subsequence of

   X=<A, B, C, D, A, B>

   Y=<B, D, C, A, B, A>
4. Let n=5, (P1,P2,…P5)=(20,15,10,5,1) and (d1-d5)=(2,2,1,3,3). Find the optimal Schedule.
5. Suppose the dimensions of matrixes A,B,C,D are 20*2,2*15, 15*40 and 40*4 respectively. What will be the optimal number of scalar multiplications.

Review Questions Cont..

6. Explain ASP with activites? explain with the help of an example?
7. Explain Memoization concept?
8. Explain OBST tree with the help of an example.
9. What the elements of DP?
10. Create the RB Tree with the following nodes.

    12,9,23,17,29,76,65

Suggested Reading/References

1. T. H. Cormen, C. E. Leiserson, R. L. Rivest, Clifford Stein, "Introduction to Algorithms", 2nd Ed., PHI, 2004.
2. A. V. Aho, J. E. Hopcroft, J. D. Ullman, "The Design and Analysis of Computer Algorithms", Addition Wesley, 1998.
3. Ellis Horowitz and Sartaz Sahani, "Computer Algorithms", Galgotia Publications,1999.
4. D. E. Knuth, "The Art of Computer Programming", 2nd Ed., Addison Wesley, 1998