CS :206M DATA STRUCTURES AND ALGORITHMS

LECTURE NOTES:-11TH JAN 2013

WORD SIZE: - Numbers of bits that a CPU can process at a time.

- For computing Fibonacci Series
  eg. Fib(0), Fib(1), Fib(2).....Fib(n).
- If we follow the recursion algorithm, the time complexity will be $O(2^{n/2})$ as discussed in earlier classes. But if we follow a more efficient algorithm we can compute the series in O(n) time.
- But if we give the input as binary representation of the value n the no. bits required will be of order $\log_2(n)$ and then the time complexity of the linear algorithm becomes of the order exponential in terms of size of input.

Unary	Binary	Decimal Value
11111	101	5
111	11	3

If we encode our input in unary it is of O(n) then total time complexity becomes linear.

Ques.) Is a given no. prime?

If we check for all factors upto n/2 then the no. of operations involved will be O(n).If input in encoded in binary i.e input size is O(log(n)) and hence the total time complexity becomes exponential in terms of size of input.but if we represent the no in unary the time complexity become linear as no of bits required to represent no in unary is linear.

Even if we represent the input in some other base eg.3 i.e 5(in decimal) = 12 (in base 3)

Then also the order will remain the same as only the base will change.

Algorithm for addition/multiplication/division of two numbers:-

- **Addition**: the time complexity will be linear with respect to the number of bits in which the input is represented as we add bit by bit and addition of each bit is constant time and we have to add n bits as the no of bits are n.
- **Multiplication**:-
  ALGORITHM 1:-

```
      101---(n bits long)
    x 101---(n bits long)
   -----------
     101
    000
   101
------------------
11001(at most 2n bits large)
```

Time complexity will be of O(n^2) in terms of the input bits assuming we are multiplying two n bit numbers as each bit will be multiplied with n bits so it will take time of the order of n and there are n such multiplication so the order becomes n^2.

Note: The result of multiplication can be at most 2n bit long so it means we have to add 2n times and as told above addition is linear so its time complexity will be of O(n) but O(n^2)+O(n)=O(n^2). So overall time complexity of algorithm remains quadratic.

ALGORITHM 2:-recursive algorithm

- [y/2]=flooring i.e. if y is represented in binary removing its least significant bit and shifting rest bits to right.
- 2*y = if y is represented in binary then shifting all bits to lest and adding 0 as the least significant bit.
- X*Y = X+(2.(X*[Y/2])) (IF Y IS ODD)
- X*Y = 2*(X*[Y/2]) (IF Y IS EVEN)

Ex. 7*9=7+2*(7*4)
7+2*(2*(7*2))
7+2*(2*(2*(7*1)))(base case when we reach 1)
=63

- As the same algo for multiplication is implemented recursively so the time complexity remains same ie. O(n^2) as implementation don't change the time complexity.

NOTE:- when we say that addition has linear time complexity then the basic assumption made is that the input is not greater than the word size if it exceeds the word size the time complexity will change

- **DIVISION BY RECURSIVE ALGO:-**
  1. You will divide the given no by 2 each time and do the flooring and you will do it till you reach the base case ie . you get 0.
  2. Now we know that division of 0 by any no the quotient is 0 and remainder is 0.
  3. Now we will fill the consecutive upper rows of quotient table starting from the bottom by multiplying the previous quotient by 2.
  4. And we will will fill the consecutive upper rows of remainder table starting from the bottom by multiplying the previous remainder by 2 and adding the flooring made ie.1 in our case as we were dividing by 2 so the floor can only be 1.and if the remainder exceeds the divisor subtract divisor from the no

and add 1 to the corresponding quotient and the no left after subtraction becomes the remainder.

5. The top most entry in the column of quotient and remainder is the required answer.
   EX.

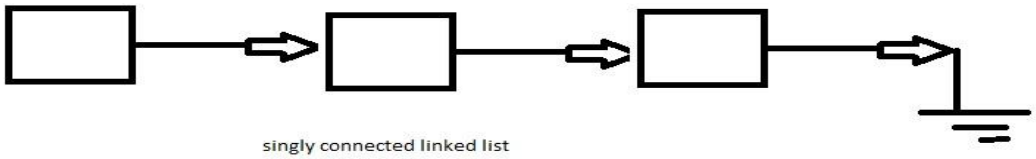
   28/9= quotient=3 and remainder=1

Divident	Divisor	Quotient	Remainder
28	9	2*1=2+1=3	1((2*5=10)=9+1)
14	9	2*0=0+1=1	5((2*7=14)=9+5)
7	9	2*0=0	7(2*3+1)
3	9	2*0=0	3(2*1+1)
1	9	2*0=0	1(2*0+1)
0	9	0	0

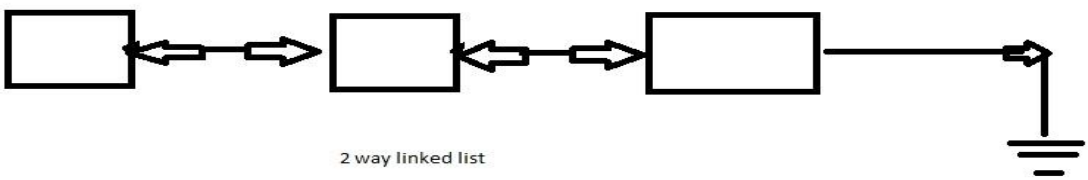
- If we had tristate computers then we had to divide at each step by 3 instead of 2.
- As Algo similar to multiplication the time complexity is of O(n^2).
- Here input size is O(n) with respect to no of bits.
- Total steps =n (as we divide by 2 we are shifting 1 bit right and there are n such bits)
- Each step = O(n) as 2 is multiplied at each step and multiplication of single bit is linear.
- Total time complexity=O(n^2).(for worst case as we are adding 1 at each step addition is of the order of O(n) and input size is also of the oreder of n so it becomes quadratic).

LINKED LIST

- It is a type of data structure.
- There can be a single way, double way or n-way linked list.



singly connected linked list



2 way linked list

-
- Operations on linked lists:-

a. Adding node to end/beginning/anywhere in the linked list
b. Searching a value
c. Deleting a node/value.
d. Inserting a value.

- Time complexities:-
    a. Searching a value:- O(n) (as we have to travel n nodes)
    b. Searching all instances of a value:- O(n) (as we have to travel n nodes)
    c. Deleting last node:-O(n)
    d. Deleting a positition:-linear with respect to the position we give.
    e. Deleting First node:-Constant
    f. Deleting a value:-linear
    g. Deleting all instances of a value:-linear.

Made by:-

AASHISH AMBER

11010201