COMPSCI 180: Assignment #1

Due on Wednesday, April 06, 2016

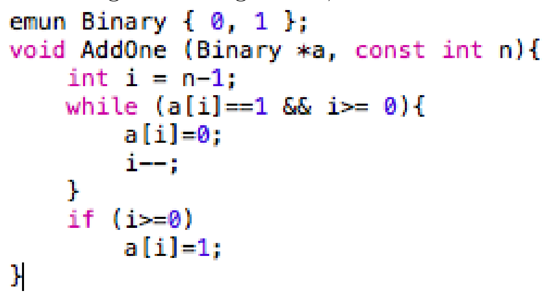
GAFNI, ELIEZER 4:00pm

Naiyan Xu

Problem 1

Prove the correctness of the following algorithm of binary addition.
Input: n (in binary representation)
Output: n + 1 (in binary representation)
find the first digit that is 0 starting from right to left. Denote the position of first 0 to i.
flip the ith digit to 1 and flip the 1, 2, . . . ,(i-1)th digit to 0
return the flipped number n'
Answer:

According to this algorithm, we can write a function as follow:



```
emun Binary { 0, 1 };
void AddOne (Binary *a, const int n){
    int i = n-1;
    while (a[i]==1 && i>= 0){
        a[i]=0;
        i--;
    }
    if (i>=0)
        a[i]=1;
}
```

The base case is when the input number is 0, which can be represented as (0). Applying this algorithm, since the first digit is 0, the result we get will be (1). Assume the for number n, kth digit is 0, then by the algorithm, a[k]=1 and all the digits on the right of kth digit are flipped, which is the binary representation of (n+1). The function terminates when it finds a zero digit or it reaches the last digit.

Problem 2

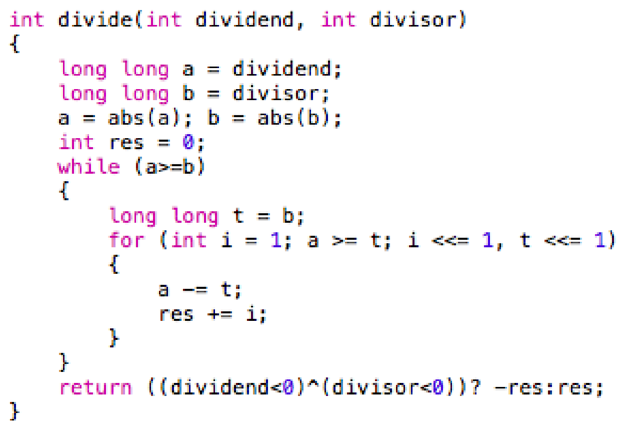
A binary tree is a tree that every node has at most 2 children. The depth of a tree is the longest path that starts from the root and ends at a leaf. Give a recursive and iterative algorithm that compute the depth of a binary tree. Justify the correctness and the complexity of your algorithms.
Answer:

(i)recursive
Input: root of the binary tree (a node)
Output: depth of the binary tree
If tree is empty then return 0
Else
(a) Get the max depth of left sub-tree recursively
(b) Get the max depth of right sub-tree recursively
(c) Get the max of max depths of left and right sub-trees and add 1 to it for the current node.
(d) Return $\max_d epth$
Time Complexity: O(n). Recursively calculate height of left and right sub-trees of a node and assign height to the node as max of the heights of two children plus 1.

(ii)iterative
Input: root of the binary tree
Output: depth of the binary tree
set the initial value of depth to be 0
for each level:
If number of nodes at this level is 0, return depth
else remove nodes of this level and add nodes of next level. And increase depth by 1.
Time Complexity: O(n). Traverse level by level. Whenever move down to a level, increment height by 1 (height is initialized as 0). Count number of nodes at each level, stop traversing when count of nodes at next level is 0.

Problem 3

You are given integers n and d in decimal representation. Describe a recursive and iterative version of elementary-school long division to divide n by d. Your primitives are addition, subtraction, multiplication of integers a and b, and integer division of a by b if a ¡ 10b. Analyze the complexity (number of operations) of your algorithm. An operation is using any of the primitives between a and b.
Answer:



```
int divide(int dividend, int divisor)
{
    long long a = dividend;
    long long b = divisor;
    a = abs(a); b = abs(b);
    int res = 0;
    while (a>=b)
    {
        long long t = b;
        for (int i = 1; a >= t; i <<= 1, t <<= 1)
        {
            a -= t;
            res += i;
        }
    }
    return ((dividend<0)^(divisor<0))? -res:res;
}
```

This algorithm basically subtract multiple times of a from b until we get a comparably small number. Since each subtraction takes O (log max(a, b)) time. therefore the total running time is O($\log^2 max(a, b)$).

Problem 4

Suppose you are playing the game of NIM. This game begins with a placement of n rows of matches on a table. Each row i has mi matches. Players take turns selecting a row of matches and removing any or all of the matches in the row. Whoever claims the final match from the table wins the game. This game has a winning strategy based on writing the count for each row in binary and lining up the binary numbers (by place value) in columns. We note that a table is favorable if there is a column with an odd number of ones in it, and the table is unfavorable if all columns have an even number of ones
(a) Prove that, for any favorable table, there exists a move that makes the table unfavorable. Prove also that, for any unfavorable table, any move makes the table favorable for ones opponent. Write the algorithm that on an input of favorable table outputs the row and the number of matches to remove from that row, to make the table unfavorable.
(b) Give a winning strategy for this game.
Answer:

Problem 4 continued on next page. . .

(a)
Assume the table is favorable. Then we can select the most significant column with an odd count and change one of the 1's in that column to 0. Then change other columns in that changed row to make all columns have even number of ones (an unfavorable table). We can always do those steps because we only change the number of matches in one row and since we change the most significant column form 1 to 0, the value of that row will always decrease, which follow the rules.
Assume the table is unfavorable. We can simply select any row with 1's and change one or more of the 1's into 0. So there exists at least one column with odd number of 1's. We can always do this for the same reason as mentioned above.

```
Algorithm:
Input: favorable table
Output: unfavorable table
select the most significant column with an odd count, denoted as j
select one of rows with 1 in column j, denoted as i
change the 1 on the position [i,j] of the table into 0
for other columns with odd number of 1's except j, change the ith row to make them even
```

For example, suppose we start off with three rows of matches of 7, 8 and 9. The binary representations of number of matches are 7 = 0111, 8 = 0111, 9 = 1001. By the algorithm above, A will win the game.
A:(7,8,9)-(1,8,9) unfavorable table
B:(1,8,9)-(1,8,4)
A:(1,8,4)-(1,5,4) unfavorable table
B:(1,5,4)-(1,4,4)
A:(1,4,4)-(0,4,4) unfavorable table
B:(0,4,4)-(0,4,2)
A:(0.4,2)-(0,2,2) unfavorable table
B:(0,2,2)-(0,2,1)
A:(0,2,1)-(0,1,1) unfavorable table
B:(0,1,1)-(0,1,0)
A:(0,1,0)-(0,0,0) unfavorable table

(b)
The winning strategy is to make sure every our move makes the favorable table into a unfavorable table.