Theoretical Assignment 1

Kaushik Raj V Nadar
Roll No.: 200499
ESO207A: Data Structures and Algorithms
February 13, 2022

Question 1: Guess the Number

You are playing a game on a machine. First of all the machine chooses a number N. You do not know this number and you need to make guesses to find it. Suppose you choose the number k, the machine tells whether k is less than, equal to or greater than N. The game ends if k is equal to N. If you guess a number which is greater than N, then it is considered a bad guess. You cannot make more than 1 bad guess.
(a) (20 points) Design an algorithm to find the number N in $O(\sqrt{N})$ guesses.
(b) (10 points) Analyze the complexity of your algorithm.

Solution

(a) We can implement an algorithm which traverses the squares of natural numbers for searching. When the guessed number is identified as first bad guess, We go back to the previous perfect square and search linearly from thereon.
Psuedocode :

```
input n
let i = 1
while true:
    let square = i*i
    if square=n :
        print "Number Found!"
        exit program or return square
    if square>n :
        break
    i = i+1
i = i-1
square = i*i
while true :
    if square = n :
        print "Number Found!"
        exit program or return square
    square = square+1
```

(b) Traversing though square of natural numbers till we reach a perfect square greater than N takes $\sqrt{N}$ steps in worst case. Ex: $1^2, 2^2, 3^2, ..., (i-1)^2, i^2$.
The worst case of the algorithm will happen when N is a number which is 1 less than a perfect square. so lets assume $N = i^2 - 1 \implies i = O(\sqrt{N})$ for higher values of N.
Therefore, in the first part there are i steps $\implies$ O($\sqrt{N}$) steps
In second part of the algorithm, there are $i^2 - (i-1)^2$ numbers between i^2 and $(i-1)^2$

$$\implies i^2 - (i-1)^2 steps = 2i - 1 steps = 2\sqrt{N} - 1 steps \implies O(\sqrt{N}) steps$$

Therefore, overall, in this algorithm:
Time Complexity is: $O(\sqrt{N})$
Space Complexity is: $O(1)$

THEORETICAL ASSIGNMENT 1

Kaushik Raj V Nadar
Roll No.: 200499
ESO207A: Data Structures and Algorithms
February 13, 2022

Question 2: Wasp pollination of figsy

A female wasp enters the syconium of an unripe fig through an opening known as the ostiole. The wasp lays eggs with some of the flowers in the syconium. Due to this interrelation, the wasp became a rare insect, because of global warming and industrialization, the number of fig trees decreasing. But we can find some of the wasps in different parts of the world in the present time. So scientists find an artificial technique to reproduce the wasps, for this; scientists take three wasps, mix their eggs and fertilize them into a lab. But they face only one problem: the three wasps should be the same family. Here same family means, if you take any two wasps, lets assume 1, 2 any two wasps and S is a set of common ancestors of both wasps. Then, the third wasp 3 should belong to the set S(3 S). Luckily scientists have the family structure of all the wasps in the form of a hierarchical tree (here, you can assume it as a binary search tree because each family member has a certain number as an ID).
(a) (20 points) Design an algorithm(pseudocode) to find out if the three wasps belong to the same family or not.
(b) (10 points) By mistake if the wasps are not from the same family and scientists mix their eggs then it can create a serious problem, for this your algorithm should always give correct results. Write down the proof of correctness.
(c) (10 points) As the family structure of the wasps is very big, your algorithm should handle all kinds of computation. So analyze the time complexity of your algorithm.(Hint: Instead of three if scientists want to mix n wasps , then will your code able to handle the computation)

Solution

(a) Psuedocode :

```
Let t to be the binary search tree of family members which is given as input

function search(T, x) :
    let p=T;
    let found=false
    while(found=false & p<>NULL) :
        if(value(p)=x):
            found=true
        else if(value(p)<x):
            p = p->right
        else
            p = p->left
    return p

function isExist(T, x):
    let p=T;
    let found=false
    while(found=false & p<>NULL) :
        if(value(p)=x):
            found=true
            return true;
        else if(value(p)<x):
            p = p->right
        else
            p = p->left
    return false

let subtree = NULL
let count=0
let k=3
input array w[k]
for i from 0 to n-1:
    subtree = search(t,w[i])
    for j from 0 to k-1 :
        if(j<>i & (isExist(subtree, w[j]) = true)):
            count = count + 1
    if(count=k-1):
        print "Same Family"
        end program
    else :
        count=0

print "Not same family"
end program
```

(b) We check for the base case, i.e. when any of the members is at the root node. It is obvious that algorithm will work as the other 2 will exist in the subtree which will be the whole tree in this case and hence will always be a family.
The algorithm checks for each member whether the other 2 members contain in the subtree of the parent. In this way, we get whether they form a family even if one match exists.
This proves that the algorithm is correct.

(c) The binary seach operation takes $O(Height(tree))$.
Since, height of BST is $O(log(n))$, we can say the number of steps in search operation is $O(logn)$.
There are constant number of binary search operations inside loop. Therefore, $O(logn)$ steps will be required.
Considering that only 3 wasps are to be mixed,
Time Complexity: $O(logn)$
Space Complexity: $O(n)$

For General Case (k wasps are mixed),
Time Complexity: $O(k * logn)$
Space Complexity: $O(n)$

Theoretical Assignment 1

Kaushik Raj V Nadar
Roll No.: 200499
ESO207A: Data Structures and Algorithms
February 13, 2022

Question 3. Count it! (Divide and Conquer)

Consider a square grid of side length N, i.e. there are N rows and N columns in the grid. Each cell of the grid is either coloured white or black. You are also given a non-negative integer k. A rectangle in the grid is considered good if both the properties hold

1. Sides of the rectangle are parallel to the sides of the grid and are along the borders of the cells.
2. Number of black cells in the rectangle is at most k.

(a) (30 points) Design an algorithm to compute the number of good rectangles in a given square grid.
(b) (10 points) Give the proof of correctness of your algorithm.
(c) (10 points) Give a formal proof of the complexity of your algorithm.

Solution

(a) Psuedocode :

```
function atmostBlack(M, start_i, start_j, end_i, end_j, k):
    let count = 0
    for i from start_i to end_i:
        for j from start_j to end_j:
            if(M[i][i]=='B'):
                count = count+1
    if(count<=k):
        return 1
    return 0

function allPossibleRects(M, r, c, start_i, start_j, end_i, end_j, k):

    if(start_i>end_i or start_j>end_j or (r=0 & c=0)):
        return 0

    let isBlck  = almostBlack(M, start_i, start_j, end_i, end_j, k)

    return
    (isBlck +
    allPossibleRects(M, r-1, c, start_i+1, start_j, end_i, end_j, k) +
    allPossibleRects(M, r, c-1, start_i, start_j+1, end_i, end_j, k) +
    allPossibleRects(M, r-1, c, start_i, start_j, end_i-1, end_j, k) +
    allPossibleRects(M, r, c-1, start_i, start_j, end_i, end_j-1, k))

input n
input 2D array M[n][n]
input k

let reqRectCount = allPossibleRects(M, n, n, 0, 0, n-1, n-1, k)
print reqRectCount
```

(b) The recursive program calculates the black squares in each of the possible rectangles in the Overall square grid of size N.
The base case is when number of rows = number of columns = 1, it checks whether the grid is black or red and increments 1 or 0 accordingly to the count.
For a general rectangle in the grid, the whole is traversed and the total count of all black squares is counted and accordingly the function returns 1 or 0 whether the number of black squares are atmost k or not.
The function recursively calculates the count of black squares in each possible rectangle of the grid and verifies the whether the count is atmost k for each of them. Hence, the algorithm is supposed to work correctly.

(c) For calculating the count of black squares and verifying whether it is atmost k, it takes $O(n^2)$ steps.
The total possible rectangles are calculated recursively by incrementing start or decrementing end coordinates of the rectangle separately,
Since, there are n columns and n rows, all possible rectangles are found in O(n) steps separately.Hence,
Time Complexity: $O(n^3)$
Space Complexity: $O(n^2)$

THEORETICAL ASSIGNMENT 1

Kaushik Raj V Nadar
Roll No.: 200499
ESO207A: Data Structures and Algorithms
February 13, 2022

Question 4. Sending message

Bob wants to send a message to his friend Alice. Bob is afraid that someone can attack the server and read his message, but Bob knows cryptography very well. So he decrypts his message into an integer(in decimal form) dec_msg, which is non-negative and represented as a string. But he faces only one problem: he cannot send this message through the available server because it has its capacity. So, he decides to reduce the size of dec_msg by deleting red_msg digits from the dec_msg. By deleting red_msg digits, he wants that the resultant number should be smallest possible, so that he can send it easily.

(a) (15 points) Design an efficient algorithm(pseudocode) to find the smallest possible number. Here you can assume that dec_msg don't have any leading zeros and number of digits in dec_msg is greater than red_msg. Also you can assume red_msg as an integer. You can delete the digits in any order.
(b) (15 points) Analyze the time complexity of your algorithm and give the proof of correctness.

Solution

(a) Algorithm :
The best digit to remove at each step is the leftmost digit that is greater than the one on its right. If there are none (digits are sorted), start removing from the right end.
(b)
Time Complexity: O(n)