Cmpe 300 – Analysis of Algorithms
Fall 2013
Assignment 1

Due Date: Friday, November 22, 2013

Question 1

Here is a pseudocode for a custom recursive algorithm:

```
procedure Scramble(A[1:n]) recursive
input:    A[1:n] (an array of integers with size n)
output:   A[1:n] (array altered by the procedure)

        if n=3 then
                interchange(A[1],A[2])
                interchange(A[2], A[3])
        else
                m := n/3
                for i:=1 to m do
                        Temp1[i] := A[i]
                        Temp2[i] := A[i+m]
                        Temp3[i] := A[i+2*m]
                endfor
                Scramble(Temp1)
                Scramble(Temp2)
                Scramble(Temp3)
                for i:=1 to m do
                        A[i] := Temp3[i]
                        A[i+m] := Temp1[i]
                        A[i+2*m] := Temp2[i]
                endfor
        endif
```

Scramble algorithm takes an array with n elements. What it does is not important. Assume n is a positive power of 3. The built-in procedure: interchange swaps the values of two variables, and it has constant complexity. As you may notice from the code, if the size of the given array is 3, the time required to scramble that array will be a constant, say C operations.

T(3) = C.

-Write the recursion T(n), which states the time required to Scramble an array with n elements.
-Solve the recursion and state the complexity of Scramble algorithm in big O notation.

Answer 1)

$$T(n) = 3 * T\left(\frac{n}{3}\right) + 2n$$

$$= 3 * \left[3T\left(\frac{n}{9}\right) + 2.\frac{n}{3}\right] + 2n$$

$$= 9 * T\left(\frac{n}{9}\right) + 4n$$

$$= 9 * \left[3T\left(\frac{n}{27}\right) + 2\frac{n}{9}\right] + 4n$$

$$= 27\,T\left(\frac{n}{27}\right) + 6n$$

$$= 3^k\,T\left(\frac{n}{3^k}\right) + 2kn \longrightarrow \text{say, } n = 3.3^k,\ k = \log_3\left(\frac{n}{3}\right)$$

$$= \frac{n}{3} * \underset{(c)}{T(3)} + 2 * \log_3\left(\frac{n}{3}\right) * n$$

$$= \frac{n}{3} * c + 2 * n * \log_3\left(\frac{n}{3}\right)$$

$$O(n) + O(n.\log n) = O(n.\log n)$$

Question 2

You are given two lists, both with size n. First list contains n different positive prime numbers.

-Write an algorithm which checks if the second list contains the squares of all the numbers in the first list. For example if the given lists are: [3, 5, 11, 23, 17] and [529, 289, 9, 25, 121], your algorithm should return true. Or if they are [2, 5, 3] and [4, 13, 9], your algorithm should return false. Note that the second list doesn't have to contain the squares in order.

-Analyse and state the complexity of your algorithm.

Note: If you need, you can call well-known algorithms like sorting algorithms, binary search, etc. in your algorithm. You can refer to their names and complexities. No need to analyze them.

Submission

Bring hard copies of your answers during the deadline day, to Mustafa Tuğrul Özşahin's desk in ETA 41, or his mailbox in the secretarial office.

Answer:

Function Squarecheck (A[1:n], B[1:n])

input: A[1:n] (An array of positive integers)
B[1:n] (An array of positive integers)

output Boolean (True if second array contains all squares)

```
for i:= 1 to n do                          } O(n)
    A[i] = A[i] * A[i]
end for
Quicksort(B), Quicksort(A)                 } O(n log n)
for i:= 1 to n do                          } O(n) in the worst case
    if (A[i] != B[i]) then                 } O(1) in the best case
        return false
    end if
end for
return true
```

Overall = O(n log n)