Data Structures and Algorithms : ESO207A

Practice-sheet 1

Time complexity and Efficient Algorithms

Important Note: The sketch of the solution or the answer for each question is also provided on the last page of this sheet. But you are strongly advised to view it only after you have spent sufficient time on each question.

1. What is the time complexity of each of the following subroutines as a function of N ? Assume each instruction takes $O(1)$ time only. Give proper reasons.

   (a)
   ```
   int sum = 0;
   for (int i = 1; i < N; i *= 2)
      for(int j = 0; j < i; j++)
         sum++;
   ```

   (b)
   ```
   int sum = 0;
   for (int i = 1; i < N; i *= 2)
      for (int j = 0; j < N; j++)
         sum++;
   ```

2. There is an array A of size n storing a sequence of 0's followed by all 1's. Design an efficient algorithm to compute the smallest index in A containing a 1.

3. Primality problem is to determine whether a given positive integer is prime. Consider the following algorithm for Primality problem. Assume that the input integer n is greater than 2.

   ```
   IsPrime(n)
   {   x <- 2;
       isPrime <- true;
       while(x*x <=n and IsPrime)
       {     If(n mod x = 0) IsPrime <- false;
             x <- x+1;
       }
       If (IsPrime) print (n is a prime number)
       Else print (n is NOT a prime number)
   }
   ```

   What is its time complexity in terms of integer n? Is this time complexity a polynomial of the input size or exponential of the input size ?

   Optional: Do you know about the connection between IIT Kanpur and the Primality problem ? If not, find it out from the web.

4. For each of the following cases determine whether $f(n) = O(g(n))$ or $f(n) \neq O(g(n))$. You must provide formal arguments in support of your answer.

   (a) $f(n) = 10 \log_2 n$ and $g(n) = \log_{10} n$

   (b) $f(n) = 5^{n/2}$ and $g(n) = 2^n$

   (c) $f(n) = 2^{\sqrt{\log n}}$ and $g(n) = \sqrt{n}$

   (d) $f(n) = 100 \sum_{i=1}^{n} \frac{1}{i}$ and $g(n) = \log_{100} n$

5. For each of the following questions, answer Yes or No appropriately. Provide suitable arguments in support of your answer.

   (a) Let $f(n) = \lceil \log n \rceil !$ and $g(n) = n^{10}$. Is $f(n) = O(g(n))$?

   (b) Let $f(n) = O(g(n))$. Can we conclude that $2^{f(n)} = O(2^{g(n)})$?

   (c) For any two increasing functions $f(n)$ and $g(n)$, is it true that either $f(n) = O(g(n))$ or $g(n) = O(f(n))$?

6. (Revised) Give an algorithm to find the smallest and the largest elements from a list of N items using the minimum number of comparisons. Note that here we are interested in minimizing the exact number and not an upper bound in terms of big "O" notation.

7. Given a positive integer n and a list containing $n - 1$ distinct integers in the range $[1, n]$. So exactly one number from $[1, n]$ is missing. Design an $O(n)$ time algorithm to find the missing number. Your algorithm is not allowed to modify the list even temporarily, and your algorithm is allowed to use only $O(1)$ extra space. However, you may assume that any arithmetic operation involving numbers in the range $[0, n^c]$ takes $O(1)$ time for any constant c. Notice that this is slightly more powerful model than the word RAM model, but it is even closer to the real world computer. We shall discuss this model sometime later in the course.

8. (**this problem is the most nontrivial problem in this sheet.**)
   Analyze the time complexity of the following algorithm called Euclid's algorithm for GCD of two numbers. You may assume that each instruction of this algorithm takes $O(1)$ time for execution.

```
GCD(a,b)    // here a is greater than or equal to b.
{
   while b <> 0
   {
         t <- b
         b <- a mod b
         a <- t
   }
   return a
}
```

Answers or the sketch of the solutions for various questions is provided below.

1. (a) $O(N)$, (b) $O(N \log N)$
2. You may suitably modify the binary search algorithm to solve this problem in $O(\log n)$ time. You may further improve it $O(\log i)$ where i is the number of zero's in A.
3. Time complexity is $O(\sqrt{n})$. This time complexity is exponential in the input size since the input size is $\lceil \log_2 n \rceil$.
4. (a) Yes, (b) No, (c)Yes, (d)Yes.
5. (a) No, (b)No, (c)No.
6. Form pairs of the numbers, compare each pair, keep larger in set L and the smaller in set S. Observe that the smallest number will be in S and the largest number will be in L only. Now proceed. Total $1.5n$ comparisons suffice.
7. Compute the sum of all the numbers and then compare it with ... to deduce the missing number.
8. Visualize the numbers in their binary representation. What happens with the MSB (most significant bit) of a and b after each iteration. Alternatively, consider the following cases: If $b < 2a/3$, then what happens to value of a after the interation ? If $b > 2a/3$ then what happens to value of b after the iteration ?