CS-203: Design and Analysis of Algorithms

Assignment I

Course Instructor: Dr. Dibyendu Roy

Due: Oct 31, 2023, 11:59 pm

Instructions: Answers must be handwritten. Clearly justify your answers. Upload the scanned copy of your handwritten answers. Write name and roll number on the top of every page. The file name of your code will be YOUR ROLL-NO.pdf

1. Each of n users spends some time on a social media site. For each $i = 1, \ldots, n$, user i enters the site at time a_i and leaves at time $b_i \geq a_i$. You are interested in the question: how many distinct pairs of users are ever on the site at the same time? (Here, the pair (i, j) is the same as the pair (j, i)).

   Example: Suppose there are 5 users with the following entering and leaving times:

User	Enter time	Leave time
1	1	4
2	2	5
3	7	8
4	9	10
5	6	10

   Then, the number of distinct pairs of users who are on the site at the same time is three: these pairs are $(1, 2), (4, 5), (3, 5)$.

   (a) Given input (a_1, b_1), (a_2, b_2), ..., (a_n, b_n) as above, there is a straightforward algorithm that takes about n^2 time to compute the number of pairs of users who are ever on the site at the same time. Give this algorithm and explain why it takes time about n^2.

   (b) Give an $O(n \log(n))$-time algorithm to do the same task and analyze its running time.

2. Consider the following algorithm that is supposed to sort an array of integers. Provide a proof that this algorithm is correct.

```
# Sorts an array of integers.
Sort(array A):
    for i = 1 to A.length:
        minIndex = i
        for j = i + 1 to A.length:
            if A[j] < A[minIndex]:
                minIndex = j
        Swap(A[i], A[minIndex])
# Swaps two elements of the array. You may assume this function is correct.
Swap(array A, int x, int y):
    tmp = A[x]
    A[x] = A[y]
    A[y] = tmp
```

Code 1: Algorithm

3. Give a linear-time (that is, an $O(n)$-time) algorithm for finding the minimum of n values (which are not necessarily sorted).

4. Consider the following recursive algorithm to find the minimum of a set of n items.

Algorithm 1: FindMinimum

Input: List $A = [a_1, \ldots, a_n]$ of n items;
Output: $\min\{a_1, \ldots, a_n\}$;
if $n = 1$ **then**
  return a_n;
end
$A_1 = A[0 : n/2]$;
$A_2 = A[n/2 : n]$;
return *min(FindMinimum(A_1), FindMinimum(A_2))*

Briefly argue that the algorithm is correct. Analyze the running time of this recursive algorithm.

5. Solve the following recurrence relations

   (a) $T(n) = T(n/2) + T(n/4) + T(n/8) + n$, where $T(1) = 1$.
   (b) $T(n) = 2T(n/2) + 3n$.
   (c) $T(n) = 3T(n/4) + \sqrt{n}$.
   (d) $T(n) = 7T(n/2) + \Theta(n^3)$.
   (e) $T(n) = 2T(\sqrt{n}) + 1$, where $T(2) = 1$.

6. Using the definition prove that $n|\sin(\pi n/2)| = O(n|\cos(\pi n/2)|)$.

7. Using the definition prove that $n|\sin(\pi n/2)| = \Omega(n|\cos(\pi n/2)|)$.

8. Consider the procedure SUM-ARRAY on the facing page. It computes the sum of the n numbers in array $A[1 : n]$. State a loop invariant for this procedure, and use its initialization, maintenance, and termination properties to show that the SUM-ARRAY procedure returns the sum of the numbers in $A[1 : n]$.

```
SUM-ARRAY(A,n)
     sum = 0
     for i=1 to n
         sum = sum+A[i]
return(sum)
```

Code 2: Algorithm

9. You can also think of insertion sort as a recursive algorithm. In order to sort $A[1 : n]$, recursively sort the subarray $A[1 : n - 1]$ and then insert $A[n]$ into the sorted subarray $A[1 : n - 1]$. Write pseudocode for this recursive version of insertion sort. Give a recurrence for its worst-case running time.

10. Show how to multiply the complex numbers $a + ib$ and $c + id$ using only three multiplications of real numbers. The algorithm should take a, b, c, and d as input and produce the real component $ac - bd$ and the imaginary component $ad + bc$ separately.

11. Suppose that you have $\Theta(n^\alpha)$ -time algorithm for squaring $n \times n$ matrices, where $\alpha \geq 2$. Show how to use that algorithm to multiply two different $n \times n$ matrices in $\Theta(n^\alpha)$ time.

12. Sketch the recursion tree corresponding to the recurrence $T(n) = 4T(n/2) + n$, and guess a good asymptotic upper bound on its solution.