Name: Vivek Vikram Pundkar

Roll no: 77

GR no: 11910860

Division: C

Batch: 3

Algorithm:

1. Josephus Problem:
2. Pass the number of people ‘n’ and the number of people to be skipped after each killing ‘k’ to the Josephus function.
3. If value of n is not 1, call the josephus function recursively and pass ‘n-1’ and k as functions.
4. Return (josephus(n - 1, k) + k-1) % n + 1
5. Stop when n = 1 and return 1.
6. GCD:
7. Pass the two numbers ‘a’ and ‘b’ to the gcd function.
8. Recursively call gcd function again passing a%b and a as parameters.
9. Stop when a = 0 and return b as final answer.
10. Exponential:
11. Pass the two numbers ‘x’ and ‘n’ to the gcd function.
12. Calculate m by calling gcd function recursively and passing x and n/2 as parameters.
13. Stop calling recursive function when n = 0 and return 1.
14. Return m\*m\*x for every recursion if n is odd.
15. Return m\*m for every recursion of n is even.

Analysis:

1. Josephus problem:

This problem is solved using recursive functions. For n people, the recursive function will be called n-1 times. For every recursive call we only return a single answer. So, time complexity for the return statement will be a constant i.e., O(1). Let’s assume the total time required to be T(n). But for recursively calling the function n-1 times the time complexity would be T(n-1). Thus, we can say,

T(n) = T(n-1) + O(1)…. (1)

T(n-1) = T(n-2) + O(1)…. (2)

T(1) = O(1) …. (3)

After substituting values, we get,

T(n) = O(n)

So, from the above calculations we can say the total time required for Josephus problem will be O(n).

1. GCD:

So the algorithm to find gcd will keep on running until b=0 in gcd(b, a%b). Let’s assume there are n no of steps performed till b becomes 0. Assuming a>b, by using the principle of mathematical induction we can prove that value of a will be at least f(n+2) and value of b will be at least f(n+1) where f(n) is the nth term in the Fibonacci series. So,

A>=f(n+2) & b>=f(n+1) …. (1)

Now according to the Binet formula,

F(n) = {((1 + √5)/2)n – ((1 – √5)/2)n}/√5  
          or  
f(n) ≈ ∅n

From this we can say,

N ≈ log∅(f(n)) …. (2)

Combining (1) and (2),

F(n+1) ≈ min(a,b)

N+1 ≈ log∅min(a,b)

O(n) = O(n+1) = log(min(a,b)) …. (3)

Thus, we can say the time required for this algorithm will be O(log(min(a,b))).

1. Exponential:

This problem is solved using the recursive function. The 2nd parameter in the expo() function gets divided by 2 for every single recursive call until n becomes 0. Let the total time required be T(n). Then time required for recursive function will be T(n/2) and the time required for the return statement will be O(1). So, we can say:

T(n) = T(n/2) + O(1)….(1)

T(n/2) = T(n/4) + O(1)….(2)

From the above equations,

T(n) ~ O(logn)

So, the time required to find exponential using divide and conquer will be O(logn).