Name: Vivek Vikram Pundkar

Roll no: 77

GR no: 11910860

Division: C

Batch: 3

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Josephus Problem**

**Algorithm**

1. Pass the number of people ‘n’ and the number of people to
2. be skipped after each killing ‘k’ to the Josephus function.
3. If value of n is not 1, call the josephus(n-1, k)
4. Position returned by call will be considered and

return (josephus(n - 1, k) + k-1) % n + 1

1. Stop when n = 1 and return 1.

**Analysis:**

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)

**Time Complexity: O(n)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**GCD**

**Algorithm**

1. Pass the two numbers ‘a’ and ‘b’ to the GCD function.
2. If we subtract a smaller number from a larger GCD doesn’t change, so if we keep subtracting repeatedly the larger of two, we end up with GCD.
3. Now instead of subtraction, if we divide the smaller number a%b,
4. Stop when a = 0 and return b as final answer.

**Analysis:**

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)

**Time Complexity: O(Log min(a, b))**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Exponential**

**Algorithm**

1. Pass the two numbers ‘x’ and ‘n’ to the gcd function.
2. Calculate m by calling gcd function recursively and passing x and n/2 as parameters.
3. Stop calling recursive function when n = 0 and return 1.
4. Return m\*m\*x for every recursion if n is odd.
5. Return m\*m for every recursion of n is even.

**Analysis:**

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)

**Time Complexity of optimized solution: O(logn)**