Name: Vivek Vikram Pundkar

Roll no: 77

GR no: 11910860

Division: C

Batch: 3

---

# Josephus Problem

**Algorithm**

i) Pass the number of people ‘n’ and the number of people to

ii) be skipped after each killing ‘k’ to the Josephus function.

iii) If value of n is not 1, call the josephus(n-1, k)

iv) Position returned by call will be considered and
return (josephus(n - 1, k) + k-1) % n + 1

v) 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**

i) Pass the two numbers ‘a’ and ‘b’ to the GCD function.

ii) 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.

iii) Now instead of subtraction, if we divide the smaller number a%b,

iv) 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,

$F(n) = \{((1 + \sqrt{5})/2)n – ((1 – \sqrt{5})/2)n\}/\sqrt{5}$ or $f(n) \approx \emptyset n$

From this we can say,

$N \approx \log\emptyset(f(n))$ …. (2)

Combining (1) and (2),

$F(n+1) \approx \min(a,b)$

$N+1 \approx \log\emptyset\min(a,b)$

$O(n) = O(n+1) = \log(\min(a,b))$ …. (3)

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

---

# Exponential

**Algorithm**

i) Pass the two numbers 'x' and 'n' to the gcd function.

ii) Calculate m by calling gcd function recursively and passing x and n/2 as parameters.

iii) Stop calling recursive function when n = 0 and return 1.

iv) Return m*m*x for every recursion if n is odd.

v) 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,

$T(n) \sim O(\log n)$

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