INTRO TO COMPETITIVE PROGRAMMING

Prasanga Dhungel

Outline Of Todays Discussion

- **Algorithmic Complexity**
- **Prime Factorization**
- **Euler's GCD**
- **Fibonacci Sequence**
- **N Queen Problem**
- **Nim Game**

Algorithmic Complexity

- **Concerns about how fast or slow particular algorithm performs.**

- **Estimates the efficiency of algorithm asymptotically.**

Asymptotic Notation

- **Notation to express an algorithmic complexity (usually run time complexity) when the input size is sufficiently large.**

- **Time function T(n) is represented using "Big-Oh" notation(or simply Oh notation) O(g(n))**

- **T(n) = $O(n^2)$ means that an algorithm has a quadratic time complexity.**

Big-Oh Notation

- **Represents asymptotic upper bound of a function i.e worst case time complexity**

- **For a given function g(n), we denote by O(g(n)) as:**

  O(g(n) = {f(n) : there exist positive constant c and no such that $0 \leq f(n) \leq cg(n)$ for all $n > n_0$}

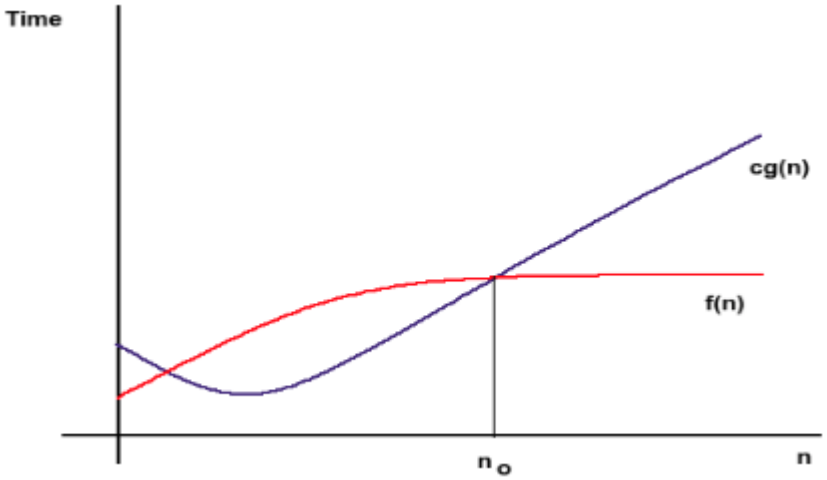

Fig: Graphical Representation of T(n) = O(g(n))

- **This means that function T(n) does not grow faster than cg(n) or, that function g(n) is an upper bound for f(n), for all sufficiently large $n \to \infty$**

Prime Factorization

- **Each number can be uniquely expressed as the product of prime numbers.e.g**

  15 = 5 * 3

  19 = 19

  28 = 2 * 2 * 7

  72930 = 2 * 3 * 5 * 13 * 11 * 17

  and so on.....

- **Prime Factorization means finding which prime numbers multiply together to make the original number.**

Lets go through code

Uses

- **Number of Factors of n**
- **Sum of Factors of n**
- **All factors of n**
- **Product of Factors of n**

If n can be expressed as $n = a_1^{p1} * a_2^{p2} * .... * a_n^{pn}$

where $a_1, a_2, ..., a_n$ are primes and $p_1, p_2, ..., p_n$ are exponents

Then,

number of factors of $n = (1 + p_1) * (1 + p_2) * ..... * (1 + p_n)$

sum of factors of n =

$[(a_1^{p1+1} - 1)/(a_1 - 1)] * [(a_2^{p2+1} - 1)/(a_2 - 1)] * ... * [(a_n^{pn+1} - 1)/(a_n - 1)]$

product of factors of $n = n^{(\text{number of factors of n})/2}$

Lets go through code

Euler's GCD

gcd(a, 0) = a

gcd(a, b) = gcd(b, a%b)

lcm(a, b) = (a * b)/ gcd(a, b)

Lets go through code

Fibonacci Sequence

- **Recursive algorithm is**

  fib(0) = 0

  fib(1) = 1

  fib(n) = fib(n -1) + fib(n -2)

- **Time Complexity of this algorithm is $O(1.618^n)$**

- **This approach takes approx. 30,000 years to compute 100th number in Fibonacci sequence.**

Using Memoization:

- **Much faster than naive solution with complexity of O(n).**

Using Matrix:

- **Fastest among all with complexity of O(log(n))**

N Queen Problem

- **Problem of placing N chess queens on an N×N chessboard so that no two queens attack each other**

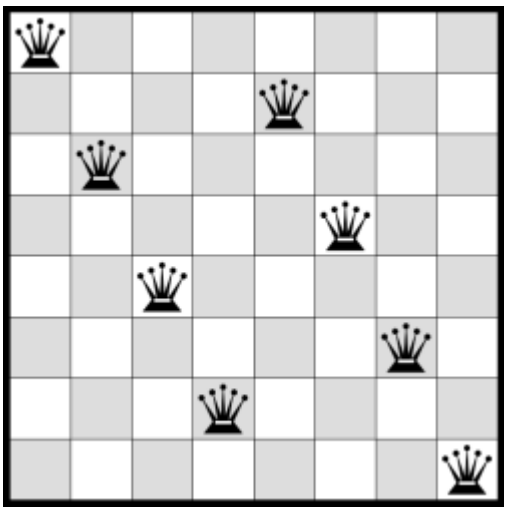

Fig:8 queen Problem

- **Solution requires that no two queens share the same row, column, or diagonal**

- **Can be solved using depth-first backtracking algorithm**

Lets go through code

Nim Game

- **There are n heaps, and each heap contains some number of sticks.**

- **The players move alternately, and on each turn, the player chooses a heap that still contains sticks and removes any number of sticks from it.**

- **The winner is the player who removes the last stick**

- **For given initial condition, the result of the Game can be determined if both player play optimally.**

Lets go through code

Thank You

Any Questions?

Contact: dhungel611@gmail.com