**Design and Analysis of Algorithms(MTCS-101)**

(Comprehensive list of questions)

(i) Write a code to generate the Fibonacci sequence until a user-specified limit.

(ii) Write a code to determine the number of function calls required to build the Fibonacci sequence until the specified limit.

(iii) Find the largest Fibonacci number less than a given number ‘n’ assuming:

1. A list of Fibonacci numbers is available
2. No such list is available

1. (Zeckendorf representation)

(i) (Greedy Strategy) Develop an algorithm to find a unique representation of a number using the Fibonacci sequence.

(ii) Determine the time complexity of the developed algorithm.

(i) Given a sorted array *A* in ascending order and an element *x*, determine whether *x* belongs to *A*.

(ii) If *x* does belong to *A*, determine the rightmost occurrence of *x* in *A*. Make sure to use ONLY a WHILE loop and NO recursion.

(iii) Prove the correctness of your algorithm.

1. Using the idea of Binary Search, search for the largest Fibonacci number less than a given positive integer.
2. By the 12th-century Indian mathematician Hemachandra, it is well-known that the Fibonacci sequence represents the number of ways one can express a number as the sum of 1s and 2s.

(i) Provide a recursive algorithm to enumerate all possible ways of expressing a given number N as a sum of 1s and 2s.

(ii) Develop an iterative algorithm to achieve the same goal as in (i).

1. (i) Give two positive integers m and n, determine the GCD(m,n).

(ii) Verify that GCD(Fm ,Fn)=FGCD(m,n), where Fn is the (n+1)th Fibonacci number. What is the minimum number of cases to verify in order to build confidence? Can this confidence be quantified with probability?

1. (i) Given two positive integers a and b, write an iterative algorithm to determine the quotient and remainder when a is divided by b. You should write a while loop and should not use the division and remainder operators.

(ii) (non-coding question) Relating the Euclid’s Division Algorithm developed on (i) to the Fibonacci numbers, obtain a beautiful pattern on the ratio of Fibonacci numbers. (Golden ratio)

1. (i) (Interpolation Search) Assuming that the given list of numbers are growing linearly, perform an efficient “jump-based” search (similar to the binary search except that the choice of m is based on the linear growth of the data).

(ii) Using the growth rate of the Fibonacci sequence, can you improve the above algorithm? What would be its time complexity?

1. Using the ‘yield’ keyword in Python, generate “infinitely many” Fibonacci numbers. That is, write a generator function that returns a generator object which yields the fibonacci sequence.
2. Given a sorted list *A* in ascending order and an element *x*,find the leftmost occurrence of *x* in *A*.