NAME: MIRZA USMAN

ROLL NO: FC-175

CLASS: BSCS-4^{TH}

SUBJECT:

DESIGN&ANALYSIS OF ALGORITHM

ASSIGNMENT NO.2

SUBMITTED TO: MAM SAIRA

Question No 1:

Solution:

Recurrence relations are mathematical expressions that define a sequence of values based on previous terms in the sequence. In simpler terms, they describe how to generate the next term in a sequence using one or more of the previous terms.

Fibonacci Sequence:

The Fibonacci sequence is perhaps the most famous example of a recurrence relation. It starts with two initial terms, 0 and 1, and each subsequent term is the sum of the two preceding terms. Mathematically, it can be defined as:
$F(n)=F(n-1)+F(n-2)$ for $n\geq 2$, with initial conditions $F(0)=0$ and $F(1)=1$.

Factorial Function:

The factorial function is defined recursively as the product of all positive integers up to a given integer n. Mathematically:
$n!=n\times(n-1)!$ for $n>0$, with base case $0!=1$.

Towers of Hanoi:

The Towers of Hanoi is a classic problem in computer science and mathematics. It involves moving a tower of disks from one peg to another, following certain rules, using a third peg as an auxiliary. The minimum number of moves required to solve the problem can be expressed recursively:
$T(n)=2T(n-1)+1$ for $n>0$, with base case $T(0)=0$.

Binary Search:

Binary search is a search algorithm that finds the position of a target value within a sorted array. It works by repeatedly dividing the search interval in half. The algorithm can be represented recursively:

$$BSearch(A,x,low,high) = \begin{cases} -1 & \text{if } low > high \\ BSearch(A,x,low,mid-1) & \text{if } A[mid] > x \\ BSearch(A,x,mid+1,high) & \text{if } A[mid] < x \\ mid & \text{if } A[mid] = x \end{cases}$$

Merge Sort:

Merge sort is a sorting algorithm that follows the divide-and-conquer strategy. It divides the input array into two halves, sorts each half independently, and then merges the sorted halves. The recurrence relation for merge sort can be expressed as: $T(n)=2T(n/2)+O(n)$, where $T(n)$ represents the time complexity of sorting an array of size n.

These examples demonstrate how recurrence relations can be used to define and analyze various mathematical problems and algorithms. They provide a powerful tool for understanding the behavior and complexity of algorithms and sequences.

Question No 2:

Solution:

Solving recurrence relations involves finding a closed-form expression for a sequence defined recursively. Several methods are commonly used:

1. **Substitution Method:** Guessing a form for the solution and then proving it correct by induction. This method is intuitive but may not always lead to the solution.

2. **Recurrence Tree Method:** Visualizing the recurrence as a tree, then analyzing the structure and summing up the costs. This method is useful for recurrences describing the cost of recursive algorithms.

3. **Master Theorem:** A powerful tool for solving recurrence relations that arise in the analysis of divide-and-conquer algorithms. It provides a straightforward way to determine the asymptotic behavior of solutions.

4. **Characteristic Equation Method:** Transforming the recurrence relation into a polynomial equation using generating functions or substitution. This method is particularly effective for linear homogeneous recurrence relations with constant coefficients.

5. **Generating Functions:** Representing sequences as power series and manipulating them algebraically. This method is versatile and can handle various types of recurrences, but it may involve complex algebraic manipulations.

6. **Matrix Method:** Representing the recurrence relation as a matrix equation and finding its eigenvalues and eigenvectors. This method is suitable for solving linear homogeneous recurrence relations with constant coefficients.

7. **Guess and Verify Method:** Proposing a solution based on observation or intuition, then proving its correctness by substitution into the recurrence relation.

Question No 3:

Solution:

Recurrence relations are equations that recursively define a sequence in terms of its previous terms. The general form of a linear recurrence relation of degree 2 is:

$$T(n)=a\cdot T(n-1)+b\cdot T(n-2)+c$$

Where:

- $T(n)$ is the value of the sequence at index n,
- a, b and c are constants,
- $T(n-1)$ and $T(n-2)$ are the values of the sequence at indices $n-1$ and $n-2$ respectively.

The iterative method to solve such recurrence relations involves starting from the initial values of the sequence and iteratively calculating subsequent values until the desired value $T(n)$ is reached.

Here's how the iterative method works:

1. **Base Cases**: First, handle the base cases. These are the initial values of the sequence that are provided explicitly. For example, $T(0)$ and $T(1)$.
2. **Iteration**: Iterate over the indices starting from the index corresponding to the next value after the last initial value up to the desired value n. At each step, calculate the current value of the sequence using the recurrence relation based on the previous terms.
3. **Updating Previous Terms**: Update the values of the previous terms as you progress through the iteration. This ensures that the calculation for each step uses the correct values of the sequence.
4. **Return**: Finally, return the value of $T(n)$ calculated after the iteration.