Walchand College of Engineering,Sangli

Computer Science & Engineering

Third Year

Design and analysis of algorithm Lab

Lab course coordinator:

Mrs A M Chimanna

Week 3 Assignment

Tower of Hanoi

PRN : 21510111
Batch : T1

Q.1 What is the Tower of Hanoi?

The Tower of Hanoi is a classic mathematical puzzle and problem-solving game that involves moving a stack of disks from one rod to another, while adhering to certain rules. The puzzle consists of three rods and a number of disks of different sizes, which can be stacked onto any rod in decreasing order of size, with the largest at the bottom and the smallest on top.

The rules of the Tower of Hanoi puzzle are as follows:

1. Only one disk can be moved at a time.
2. Each move involves taking the top disk from one of the stacks and placing it on top of another stack.
3. A disk cannot be placed on top of a smaller disk.

The goal of the puzzle is to move all the disks from the starting rod to the target rod, using the other rod as an auxiliary, following the rules mentioned above.

The Tower of Hanoi puzzle is often used as an example in computer science and algorithm design to illustrate concepts like recursion and problem-solving strategies. It's also used to demonstrate the concept of exponential growth, as the number of moves required to solve the puzzle grows exponentially with the number of disks.

Q.2 Write a program to the Tower of Hanoi Problem in a recursive approach.

Algorithm:

In this algorithm, `n` represents the number of disks, and `source`, `auxiliary`, and `target` are the names of the three rods (or pegs) involved in the puzzle. The function `tower_of_hanoi` takes these parameters and prints the steps needed to solve the puzzle.

Here's how the algorithm works:

1. If `n` is 1, simply move the top disk from the source rod to the target rod.
2. If `n` is greater than 1, recursively solve the Tower of Hanoi puzzle for `n-1` disks by moving them from the source rod to the auxiliary rod, using the target rod as the auxiliary.
3. After moving the smaller `n-1` disks, move the remaining largest disk from the source rod to the target rod.
4. Finally, recursively solve the Tower of Hanoi puzzle for the `n-1` disks on the auxiliary rod, using the source rod as the auxiliary and the target rod as the target.

The algorithm follows the rules of the Tower of Hanoi puzzle, ensuring that no larger disk is placed on top of a smaller disk during the moves. The recursive nature of the algorithm helps break down the problem into smaller subproblems, ultimately leading to the solution.

Program:

```cpp
#include <bits/stdc++.h>
using namespace std;
int c=1;
void towerOfHanoi(int n,char source,char destination,char extra)
{
    if(n==1)
    {
        cout<<c<<") move disc 1 from "<<source<<" to "<<destination<<endl;
        c++;
        return ;
    }
    towerOfHanoi(n-1,source,extra,destination);
    cout<<c<<") move disc " << n << " from "<<source<<" to 
"<<destination<<endl;
    c++;
    towerOfHanoi(n-1,extra,destination,source);
}
int main()
{
    int n;
    cin>>n;
    towerOfHanoi(n,'A','C','B');
    return 0;
}
```

Q.3 What are the minimum moves to solve the Tower of Hanoi Problem.

The minimum number of moves required to solve the Tower of Hanoi problem for a given number of disks n is 2^n−1. This is a well-known mathematical result.

The formula 2^n−1 represents the optimal solution for the Tower of Hanoi puzzle. It's derived from the nature of the problem and the recursive approach used to solve it. Each move involves moving a single disk, and when we have nn disks, the total number of moves required can be calculated as follows:

- Moving n−1 disks to the auxiliary peg: 2^(n−1)−1 moves
- Moving the largest disk to the target peg: 1 move
- Moving n−1 disks from the auxiliary peg to the target peg: 2^(n−1)−1 moves

So, the total number of moves is:

2^(n−1)−1+1+2^(n−1)−1=2^(n−1)

This result shows that the Tower of Hanoi puzzle grows at an exponential rate as the number of disks increases. This is an interesting aspect of the puzzle and demonstrates its computational and mathematical significance

Q.4 What is the space complexity of the Tower of Hanoi Problem?

In the recursive solution for the Tower of Hanoi, the function `tower_of_hanoi` is called recursively for smaller subproblems until the base case is reached (when there's only one disk to move). Each function call adds a new frame to the call stack, which contains information such as the values of `n`, `source`, `auxiliary`, and `target`.

The maximum depth of the call stack is equal to the number of recursive calls made during the solution. In the Tower of Hanoi problem with `n` disks, the maximum depth of the call stack will be `n`. This is because, in each recursive call, the problem is reduced by moving `n-1` disks, and this continues until you reach the base case (moving one disk).

Therefore, the space complexity of the Tower of Hanoi problem is O(n) in terms of stack space, where 'n' is the number of disks. This means that the amount of memory required by the algorithm scales linearly with the number of disks.