18CSC204J- DESIGN AND ANALYSIS of ALGORITHMS

Record Work

Register Number : RA2011033010057

Name of the Student : Prem Kumar

Semester / Year : IVth / 2nd

Department : CSE - SWE

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR -603 203

BONAFIDE CERTIFICATE

Register No : RA2011033010057

Certified to be the bonafide record of work done by Prem Kumar of CSE-SWE T1, B.Tech Degree course in the 18CSC204J – Algorithm Analysis and Design in SRM Institute of Science and Technology, Kattankulathur during the academic year 2021-2022.

Date: 26/06/2022

Lab Incharge: S. Joseph James

Submitted for University Examination held in _____SRM

Institute of Science and Technology, Kattankulathur.

Head of Department (HOD):

TABLE OF CONTENT

Project Name: Tower Of Hanoi

Mini- Project
Design and Analysis of Algorithm

Submitted By-
Team – Geek Mode:
Gaurav Raj (RA2011033010063)
Anubhav Vats (RA2011033010062)
Prem Kumar (RA2011033010057)

Of
SRMI INSTITUTE OF SCIENCE AND TECHNOLOGY S.R.M. NAGAR, KATTANKULATHUR -603203

Contribution Table:

Work	Done By
1. Development of Algorithm	Gaurav Raj (063), Prem Kumar(057)
2. Implementation	Gaurav Raj (063)
3. Documentation	Prem Kumar(057),Anubhav Vats(062)

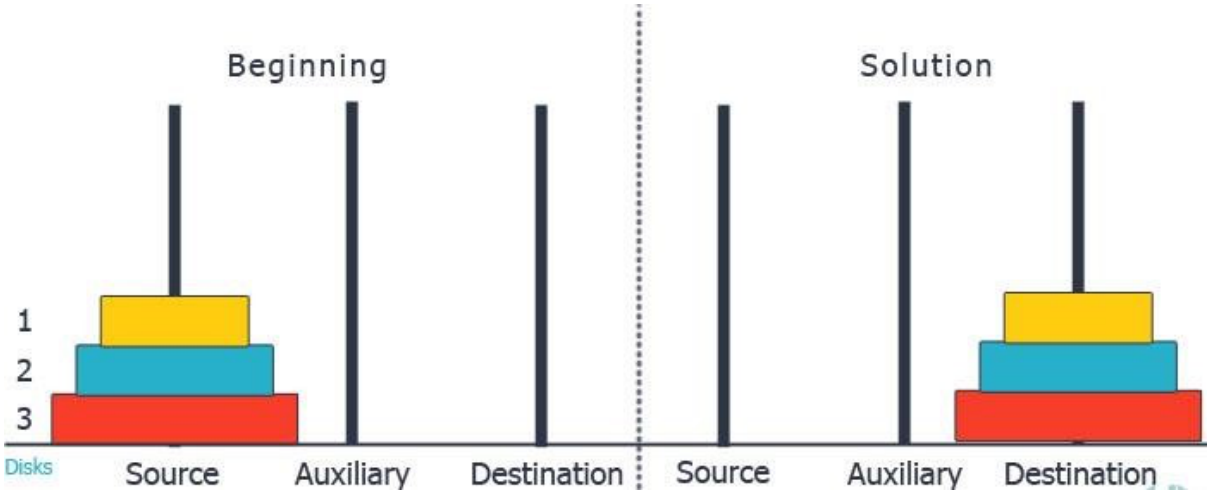
Problem Definition:

The **Tower of Hanoi** is a mathematical puzzle. It consists of **three rods** and **N** disks. The task is to move all **disks** from source tower to destination tower.

Problem Explanation:

The rules to be followed while moving the disks from one tower to another tower are as follows:

1. Only one disk can be moved at a time.
2. Only the uppermost disk can be moved from one stack to the top of another stack or to an empty rod.
3. Larger disks cannot be placed on top of smaller disks.



Design Technique Used:

Divide And Conquer

A divide and conquer algorithm is a strategy of solving a large problem by

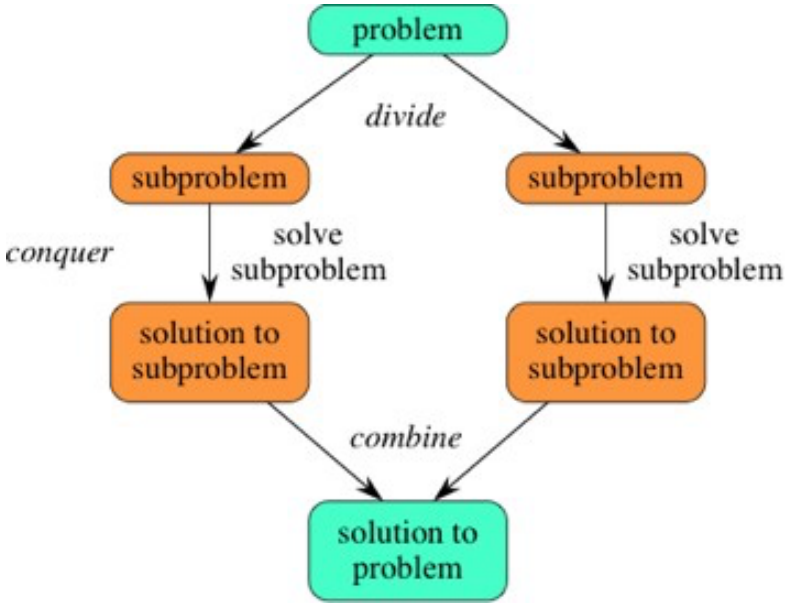
1. breaking the problem into smaller sub-problems
2. solving the sub-problems, and
3. combining them to get the desired output.

To use the divide and conquer algorithm, recursion is used while coding.

How Divide and Conquer Algorithms Work?

Here are the steps involved:

1. **Divide**: Divide the given problem into sub-problems using recursion.
2. **Conquer**: Solve the smaller sub-problems recursively. If the subproblem is small enough, then solve it directly.
3. **Combine:** Combine the solutions of the sub-problems that are part of the recursive process to solve the actual problem.



Algorithm:

Pseudo Code:

tower(disk, source, inter, dest)

IF disk is equal 1, THEN

move disk from source to destination

ELSE

tower(disk - 1, source, destination, intermediate) // Step 1

move disk from source to destination // Step 2 tower(disk - 1, intermediate, source, destination) // Step 3

END IF

END

Explanation:

Let tower 1 = 'A', tower 2 = 'B', tower 3 = 'C'.

An example with **2 disks** :

Step 1 : Shift first disk from 'A' to 'B'.

Step 2 : Shift second disk from 'A' to 'C'.

Step 3 : Shift first disk from 'B' to 'C'.

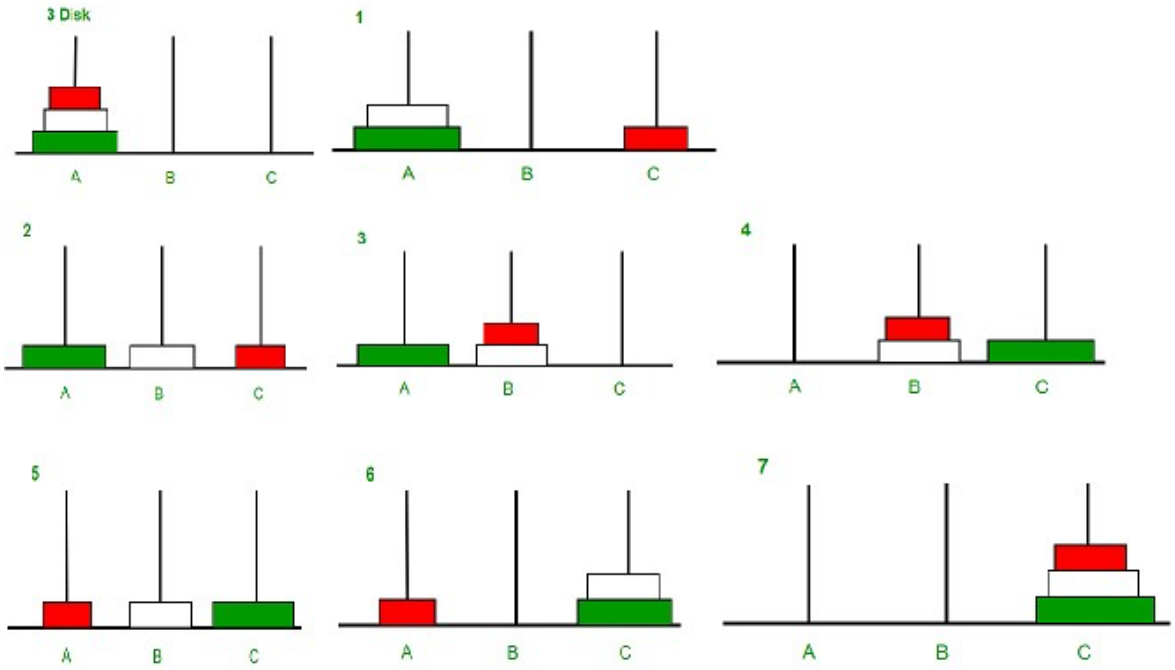
An example with **3 disks** :

Step 1 : Shift first disk from 'A' to 'C'.
Step 2 : Shift second disk from 'A' to 'B'.
Step 3 : Shift first disk from 'C' to 'B'.

Step 4 : Shift third disk from 'A' to 'C'.

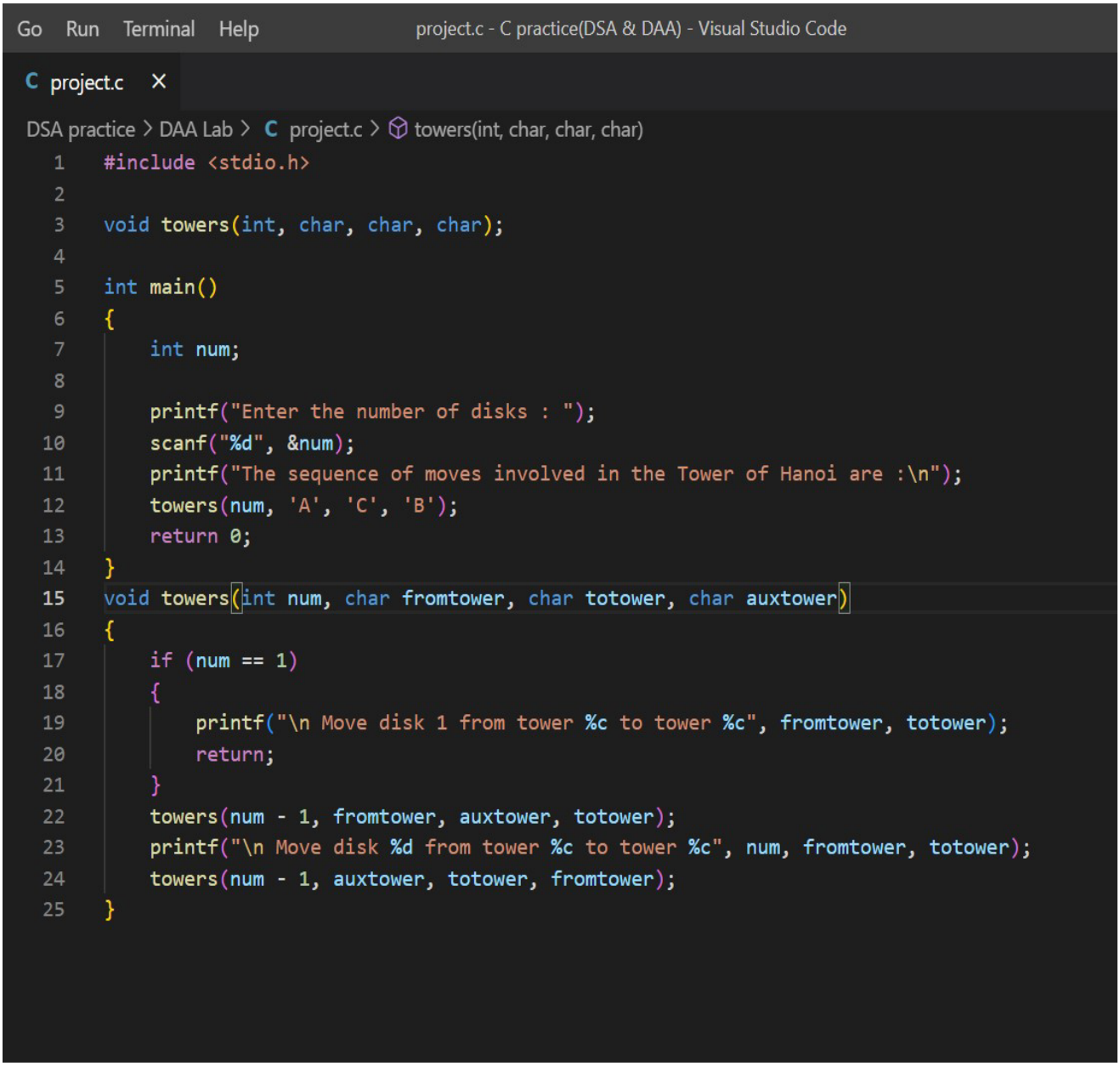
Step 5 : Shift first disk from 'B' to 'A'.
Step 6 : Shift second disk from 'B' to 'C'.
Step 7 : Shift first disk from 'A' to 'C'.
(Notice the gaps)
The pattern here is :

- Shift 'n-1' disks from 'A' to 'B', using C.
- Shift last disk from 'A' to 'C'.
- Shift 'n-1' disks from 'B' to 'C', using A.

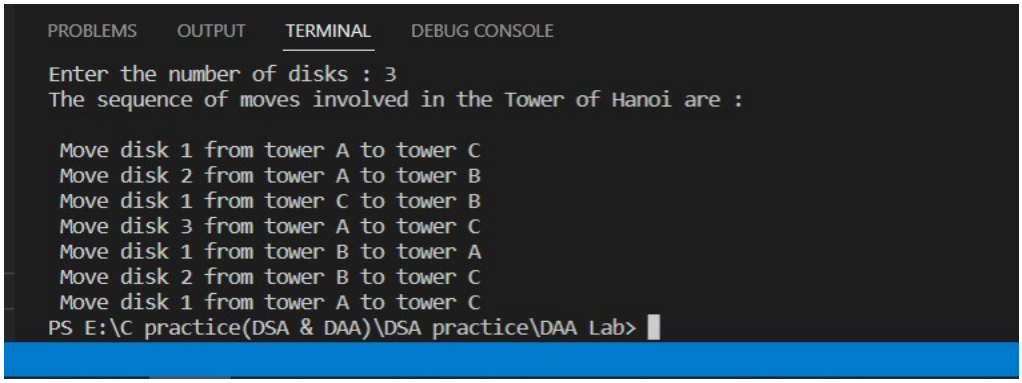


Coding:

Code:



Input/Output:



Complexity Analysis:

Recursive Equation : T(n) = 2T(n-1) + 1 ——-equation-1

Solving it by Back substitution :

T(n-1) = 2T(n-2) + 1 ———–equation-2 T(n-2)

= 2T(n-3) + 1 ———–equation-3

Put the value of T(n-2) in the equation–2 with help of equation-3

T(n-1)= 2(2T(n-3) + 1) + 1 ——equation-4

Put the value of T(n-1) in equation-1 with help of equation-4

T(n)= 2(2(2T(n-3) + 1) + 1) + 1

T(n) = 2^3 T(n-3) + 2^2 + 2^1 + 1

After Generalization :

T(n)= 2^k T(n-k) + 2^{(k-1)} + 2^{(k-2)} + ... +2^2 + 2^1 + 1

Base condition T(1) =1

n – k = 1 k = n-1 put,

k = n-1

T(n) =2^{(n-1)}T(1) + + 2^{(n-2)} + ... +2^2 +2^1 + 1

It is a GP series, and the sum is 2^n - 1

T(n)= O(2^n - 1) , or you can say O(2^n) which is exponential

for 5 disks i.e. n=5 It will take 2^5-1=31 moves.

Bibliography:

Books :-

1. C++ programming by E.balagurusamy

2.Data structure by t.schaum series Website

:-

1. www.tutorialspoint.com
2. www.geeksforgeeks.org

----------**The End**----------