COST BASED TOWER OF HANOI

A

Mini Project Report

Submitted for the course

Design and Analysis of Algorithms Lab

Semester-IV, CSE-B

By

B. AKHIL1602-18-733-063

L. SACHIN1602-18-733-094

****

**Department of Computer Science and Engineering**

**VASAVI COLLEGE OF ENGINEERING (AUTONOMOUS)**

**9-5-81, Ibrahimbagh, Hyderabad-31**

**2019-20**

**ACKNOWLEDGEMENT**

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

We respect and thank Mrs. V. Sireesha mam, for providing me an opportunity to do the project work and giving us support and guidance which made me complete the project duly. We are extremely thankful to her for providing such a nice support and guidance, although she had busy schedule managing the corporate affairs.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from all Teaching staffs of CSE which helped us in successfully completing our project work.

Also, we would like to extend our sincere esteems to all staff in laboratory for their timely support.

**TABLE OF CONTENTS**

[1. ABSTRACT 1](#_Toc40730329)

[2. INTRODUCTION 2](#_Toc40730330)

[2.1 Origin 2](#_Toc40730331)

[2.2 Recursive Solution 3](#_Toc40730332)

[2.3 Cost of Transfer 4](#_Toc40730333)

[3. DESIGN STRATEGY 5](#_Toc40730334)

[3.1 Dynamic Programming 5](#_Toc40730335)

[3.2 Comparision 5](#_Toc40730336)

[3.3 Algorithm 6](#_Toc40730337)

[4. IMPLEMENTATION 7](#_Toc40730338)

[5. RESULTS 14](#_Toc40730339)

[5.1 Output 1 14](#_Toc40730340)

[5.2 Output 2 15](#_Toc40730341)

[5.3 Output 3 16](#_Toc40730342)

[5.4 Output 4 17](#_Toc40730343)

[6. CONCLUSIONS 18](#_Toc40730344)

[7. REFERENCES 19](#_Toc40730345)

# **ABSTRACT**

The Tower of Hanoi also called the Tower of Brahma or Lucas is a mathematical game or puzzle. It consists of three rods, and a number of disks of different sizes which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at top, thus making a conical shape.

The puzzle can be with any number of disks, although many toy versions have around seven to nine of them. The minimum number of moves required to solve a Tower of Hanoi puzzle is 2n-1, where n is the number of disks.

**Applications:**

1. The Tower of Hanoi is frequently used in psychological research on problem solving.
2. It is also used as a Backup rotation scheme when performing computer data Backups where multiple tapes/media are involved.
3. Tower of Hanoi is popular for teaching recursive algorithms to beginning programming students.
4. The Tower of Hanoi is also used as a test by neuropsychologists trying to evaluate frontal lobe deficits.

This program uses recursion to show the movement of disk from one tower to another.

# **INTRODUCTION**

Tower of Hanoi is a mathematical puzzle where we have three rods and n disks. The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:

1) Only one disk can be moved at a time.

2) Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.

3) No disk may be placed on top of a smaller disk.

## 2.1 Origin

The puzzle was invented by the [French](https://en.wikipedia.org/wiki/French_people) [mathematician](https://en.wikipedia.org/wiki/Mathematician) [Édouard Lucas](https://en.wikipedia.org/wiki/%C3%89douard_Lucas) in 1883. Numerous myths regarding the ancient and mystical nature of the puzzle popped up almost immediately.[[2]](https://en.wikipedia.org/wiki/Tower_of_Hanoi#cite_note-2) There is a story about an [Indian](https://en.wikipedia.org/wiki/India) temple in [Kashi Vishwanath](https://en.wikipedia.org/wiki/Kashi_Vishwanath_Temple) which contains a large room with three time-worn posts in it, surrounded by 64 golden disks. [Brahmin](https://en.wikipedia.org/wiki/Brahmin) priests, acting out the command of an ancient prophecy, have been moving these disks in accordance with the immutable rules of Brahma since that time. The puzzle is therefore also known as the Tower of [Brahma](https://en.wikipedia.org/wiki/Brahma) puzzle. According to the legend, when the last move of the puzzle is completed, the world will end.[[3]](https://en.wikipedia.org/wiki/Tower_of_Hanoi#cite_note-3)

If the legend were true, and if the priests were able to move disks at a rate of one per second, using the smallest number of moves it would take them 264 − 1 seconds or roughly 585 [billion](https://en.wikipedia.org/wiki/1,000,000,000) years to finish,[[4]](https://en.wikipedia.org/wiki/Tower_of_Hanoi#cite_note-4) which is about 42 times the current age of the universe.

There are many variations on this legend. For instance, in some tellings the temple is a [monastery](https://en.wikipedia.org/wiki/Monastery), and the priests are [monks](https://en.wikipedia.org/wiki/Monk). The temple or monastery may be said to be in different parts of the world—including [Hanoi](https://en.wikipedia.org/wiki/Hanoi), [Vietnam](https://en.wikipedia.org/wiki/Vietnam)—and may be associated with any [religion](https://en.wikipedia.org/wiki/Religion). In some versions other elements are introduced, such as the fact that the tower was created at the beginning of the world, or that the priests or monks may make only one move per day.

## 2.2 Recursive Solution

The key to solving a problem recursively is to recognize that it can be broken down into a collection of smaller sub-problems, to each of which that same general solving procedure that we are seeking applies, and the total solution is then found in some simple way from those sub-problem’s solutions. Each of thus created sub-problems being "smaller" guarantees that the base case(s) will eventually be reached. Thence, for the Towers of Hanoi:

* label the pegs A, B, C,
* let n be the total number of disks,
* number the disks from 1 (smallest, topmost) to n (largest, bottom-most).

Assuming all n disks are distributed in valid arrangements among the pegs; assuming there are m top disks on a source peg, and all the rest of the disks are larger than m, so they can be safely ignored; to move m disks from a source peg to a target peg using a spare peg, without violating the rules:

1. Move m − 1 disks from the source to the spare peg, by the same general solving procedure. Rules are not violated, by assumption. This leaves the disk m as a top disk on the source peg.
2. Move the disk m from the source to the target peg, which is guaranteed to be a valid move, by the assumptions — a simple step.
3. Move the m−1 disks that we have just placed on the spare, from the spare to the target peg by the same general solving procedure, so they are placed on top of the disk m without violating the rules.
4. The base case being to move 0 disks (in steps 1 and 3), that is, do nothing – which obviously doesn't violate the rules.

The full Tower of Hanoi solution then consists of moving n disks from the source peg A to the target peg C, using B as the spare peg.

This approach can be given a rigorous mathematical proof with [mathematical induction](https://en.wikipedia.org/wiki/Mathematical_induction) and is often used as an example of recursion when teaching programming.

## 2.3 Cost of Transfer

The standard Tower of Hanoi problem is explained [here](https://www.geeksforgeeks.org/c-program-for-tower-of-hanoi/). In the standard problem, all the disc transactions are considered identical. Given a 3×3 matrix costs[][] containing the costs of transfer of disc between the rods where costs[i][j] stores the cost of transferring a disc from rod i to rod j. Cost of transfer between the same rod is 0. Hence the diagonal elements of the cost matrix are all 0s. The task is to print the minimum cost in which all the N discs are transferred from rod 1 to rod 3.

**Examples**

**Input:** N = 2

costs = {  
{ 0, 1, 2},  
{ 2, 0, 1},  
{ 3, 2, 0}}

**Output:** 4  
There are 2 discs, the smaller one is on the bigger one.  
Transfer the smaller disc from rod 1 to rod 2.  
Cost of this transfer is equal to 1  
Transfer the bigger disc to from rod 1 to rod 3.  
Cost of this transfer is equal to 2.  
Transfer the smaller disc to from rod 2 to rod 3.  
Cost of this transfer is equal to 1  
Total minimum cost is equal to 4.

**Input:** N = 3

costs = {  
{ 0, 1, 2},  
{ 2, 0, 1},  
{ 3, 2, 0}}

**Output:** 12

# **DESIGN STRATEGY**

## 3.1 Dynamic Programming

Dynamic programming approach is similar to divide and conquer in breaking down the problem into smaller and yet smaller possible sub-problems. But unlike, divide and conquer, these sub-problems are not solved independently. Rather, results of these smaller sub-problems are remembered and used for similar or overlapping sub-problems.

Dynamic programming is used where we have problems, which can be divided into similar sub-problems, so that their results can be re-used. Mostly, these algorithms are used for optimization. Before solving the in-hand sub-problem, dynamic algorithm will try to examine the results of the previously solved sub-problems. The solutions of sub-problems are combined in order to achieve the best solution.

So we can say that −

* The problem should be able to be divided into smaller overlapping sub-problem.
* An optimum solution can be achieved by using an optimum solution of smaller sub-problems.
* Dynamic algorithms use Memorization.

## 3.2 Comparision

In contrast to greedy algorithms, where local optimization is addressed, dynamic algorithms are motivated for an overall optimization of the problem.

In contrast to divide and conquer algorithms, where solutions are combined to achieve an overall solution, dynamic algorithms use the output of a smaller sub-problem and then try to optimize a bigger sub-problem. Dynamic algorithms use Memoization to remember the output of already solved sub-problems.

**Example:**

The following computer problems can be solved using dynamic programming approach −

* Fibonacci number series
* Knapsack problem
* Tower of Hanoi
* All pair shortest path by Floyd-Warshall
* Shortest path by Dijkstra
* Project scheduling

## 3.3 Algorithm

Approach: Idea is to use Top-down [Dynamic programming](http://www.geeksforgeeks.org/dynamic-programming/).

Let’s say mincost(idx, src, dest) be the minimum cost for transferring the discs of indices idx to N from rod src to rod dest. The third rod which is neither the source nor the destination rod would have the value rem = 6 – (i + j) as the rod numbers are 1, 2 and 3 and their sum is 6. If 1st and 3rd rods are the source and destination respectively then the auxilary rod will have number as 6 – (1 + 3) = 2.  
Now break the problem into its subproblems as follows:

* Case 1: First transfer all the discs with index (idx + 1) to N to the remaining rod. Now transfer the largest disc to destination rod. Again transfer all the discs from remaining rod to the destination rod. This process would cost as.

Cost = mincost(idx + 1, src, rem) + costs[src][dest] + mincost(idx + 1, rem, dest)

* Case 2: First transfer all the discs with index (idx + 1) to N to the destination rod. Now transfer the largest disc to remaining rod. Again transfer all the discs from destination rod to the source rod. Now transfer the largest disc from remaining rod to the destination rod. Again transfer the discs from source rod to destination rod. This process would cost as:

Cost = mincost(idx + 1, src, dest) + costs[src][rem] + mincost(idx + 1, dest, src) + cost[rem][dest] + mincost(idx + 1, src, dest)

* Answer would be equal to the minimum of the above two cases.

# **IMPLEMENTATION**

// C Source Code to implement cost based Tower of Hanoi

#include <stdio.h>

#define RODS 3

#include<stdlib.h>

#include<ctype.h>

#include<math.h>

int N;

int dp[10][RODS + 1][RODS + 1];

int c=0;

int min(int a,int b)

{

return a<b? a:b;

}

void initialize()

{

int i,j,k;

for (i = 0; i <= N; i += 1) {

for (j = 1; j <= RODS; j++) {

for (k=1; k <= RODS; k += 1) {

dp[i][j][k] = INT\_MAX;

}

}

}

}

// Function to return the minimum cost

int mincost(int idx, int src, int dest,

int costs[RODS][RODS])

{

if (idx > N)

return 0;

if (dp[idx][src][dest] != INT\_MAX)

return dp[idx][src][dest];

int rem = 6 - (src + dest);

int ans = INT\_MAX;

// printf("Move disk %d from rod %d to rod %d \n",idx ,src,dest);

int case1 =

mincost(idx + 1, rem, dest, costs)

+ costs[src - 1][dest - 1]

+ mincost(idx + 1, src, rem, costs);

int case2 = costs[src - 1][rem - 1]

+ mincost(idx + 1, src, dest, costs)

+ mincost(idx + 1, dest, src, costs)

+ costs[rem - 1][dest - 1]

+ mincost(idx + 1, src, dest, costs);

ans = min(case1, case2);

/\* if (case1<case2)

{

}

else

{

printf("Move disk %d from rod %d to rod %d \n", idx ,src,rem);

printf("Move disk %d from rod %d to rod %d \n", idx ,rem,dest);

}\*/

dp[idx][src][dest] = ans;

return ans;

// printf("Move disk %d from rod %d to rod %d \n", idx ,src,dest);

}

/\*void path()

{

int i,j,k;

for (i = 0; i <= N; i += 1) {

for (j = 1; j <= RODS; j++) {

for (k=1; k <= RODS; k += 1) {

printf("\n%d ",dp[i][j][k]);

}

}

}

} \*/

void towerOfHanoi(int n, char from\_rod,

char to\_rod, char aux\_rod)

{

if (n == 1)

{

printf("Move disk 1 from rod %c to rod %c\n",from\_rod,to\_rod);

return;

}

towerOfHanoi(n - 1, from\_rod, aux\_rod, to\_rod);

printf("Move disk %d from rod %c to rod %c \n", n ,from\_rod,

to\_rod);

towerOfHanoi(n - 1, aux\_rod, to\_rod, from\_rod);

}

void moves(int n)

{

int i;

int h[100];

h[1]=1;

for (i=2;i<=n;i++)

{

h[i]=2\*h[i-1]+1;

}

printf("No of moves required: %d\n", h[n]);

}

int main()

{

int ch,y;

int i,j;

int costs[RODS][RODS];

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TOWER OF HANOI\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n");

printf("\n");

printf("Enter the no of discs: ");

scanf("%d", &N);

do

{

printf("Choose the below options:\n");

printf("1.Minimum no of Moves\n2.Minimum cost to transfer all the discs\n");

scanf("%d", &ch);

switch(ch)

{

case 1: moves(N);

printf("\nMoves to transfer all the discs:\n");

printf("\n");

towerOfHanoi(N, 'A', 'C', 'B');

break;

case 2:

printf("\nEnter the cost Matrix:\n");

for (i=0;i<RODS;i++)

{

for(j=0;j<RODS;j++)

{

scanf("%d", &costs[i][j]);

}

}

initialize();

printf("Minimum Cost to transfer all the discs is %d", mincost(1, 1, 3, costs));

//path();

break;

default:printf("\nInvalid Choice\n");

}

printf("\nEnter 1 to go to menu or 0 to exit:");

scanf("%d", &y);

}

while(y==1);

printf("\nProgram Terminated");

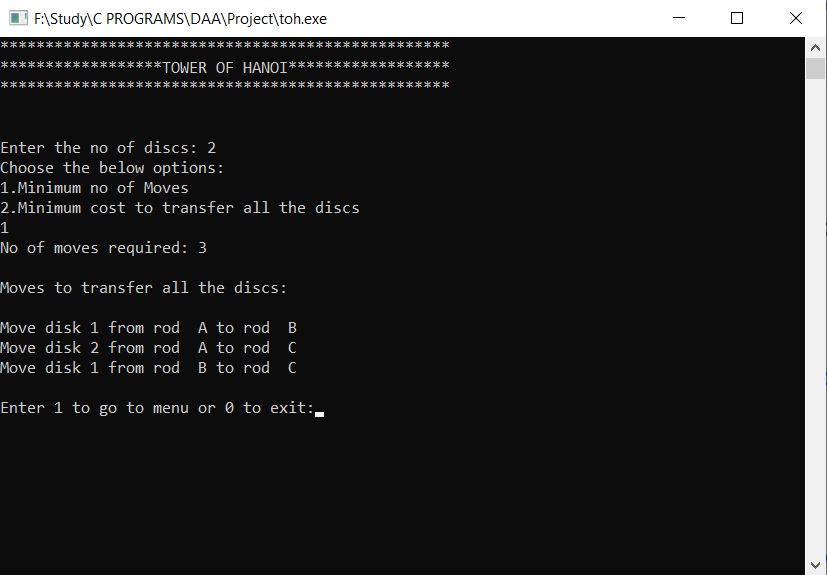
return 0;

}

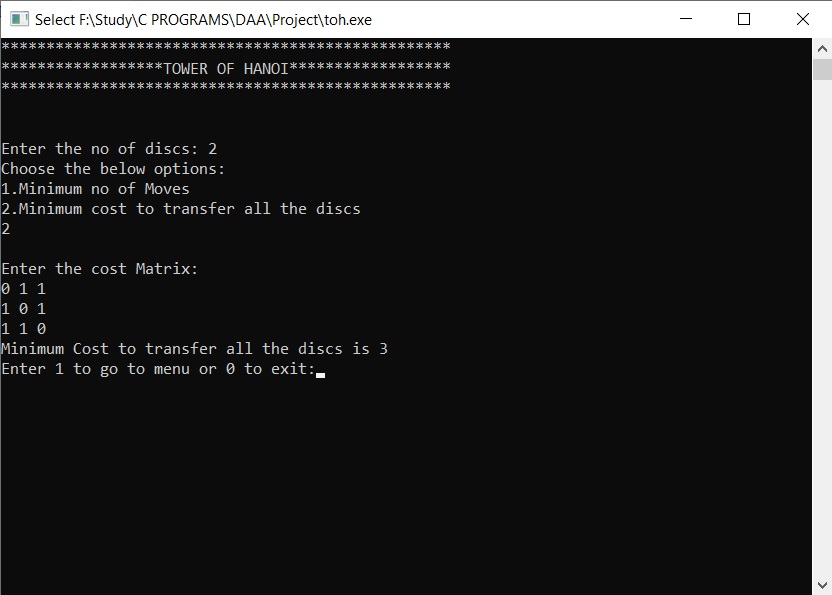
**Time Complexity:** O(N) where N is the number of discs in given rod.

# **RESULTS**

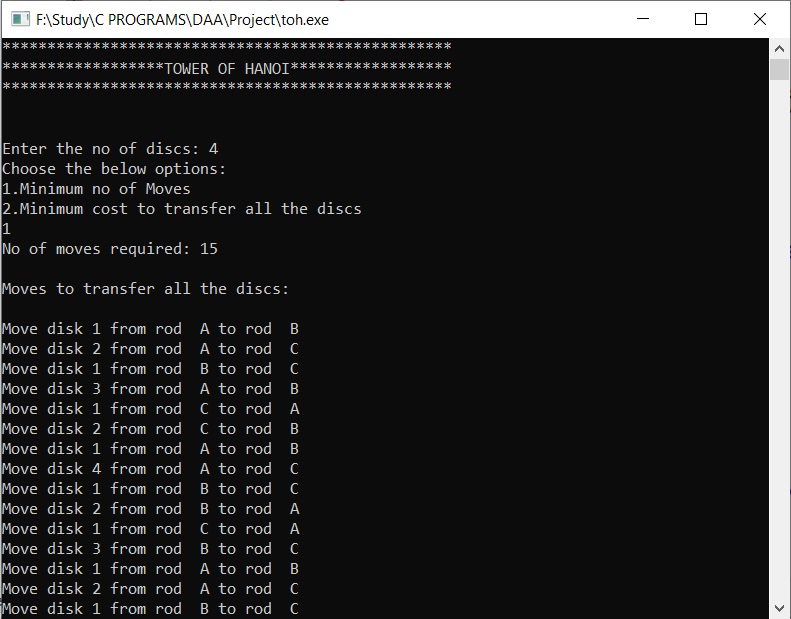
## 5.1 Output 1

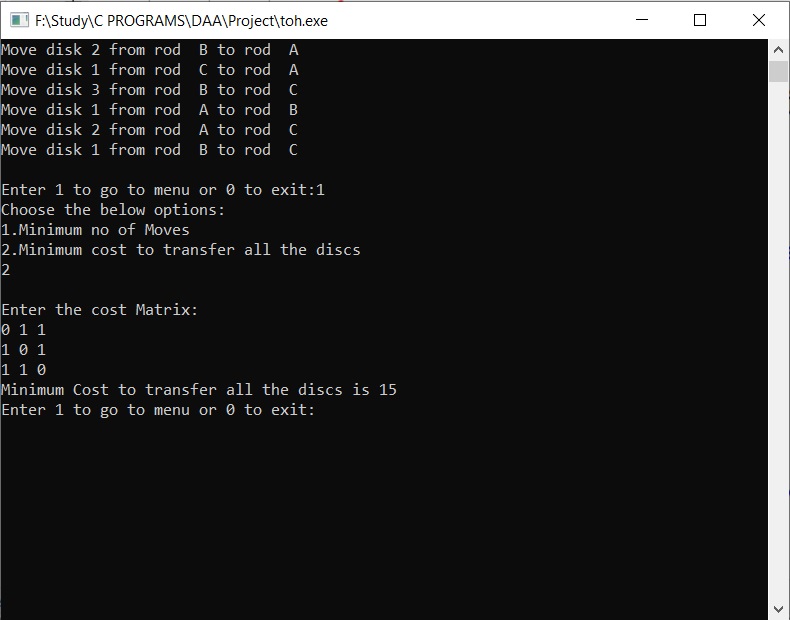


## 5.2 Output 2

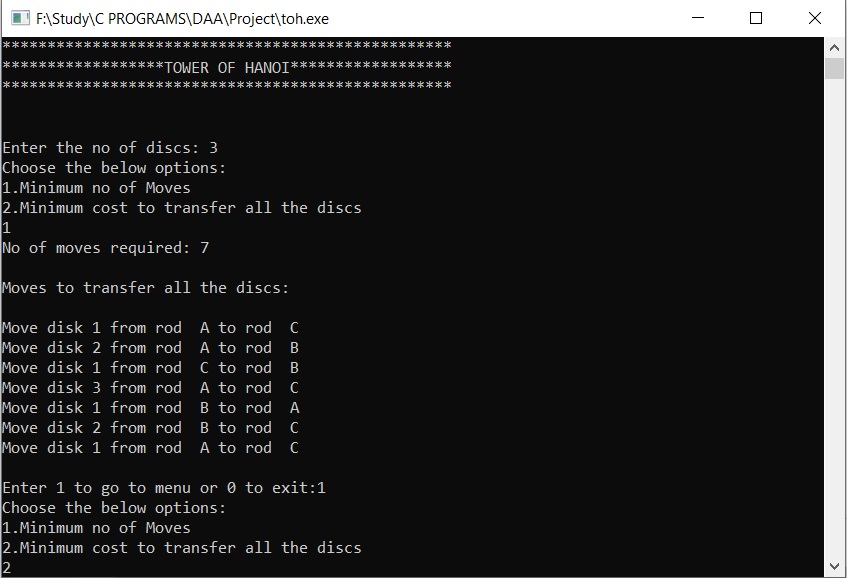


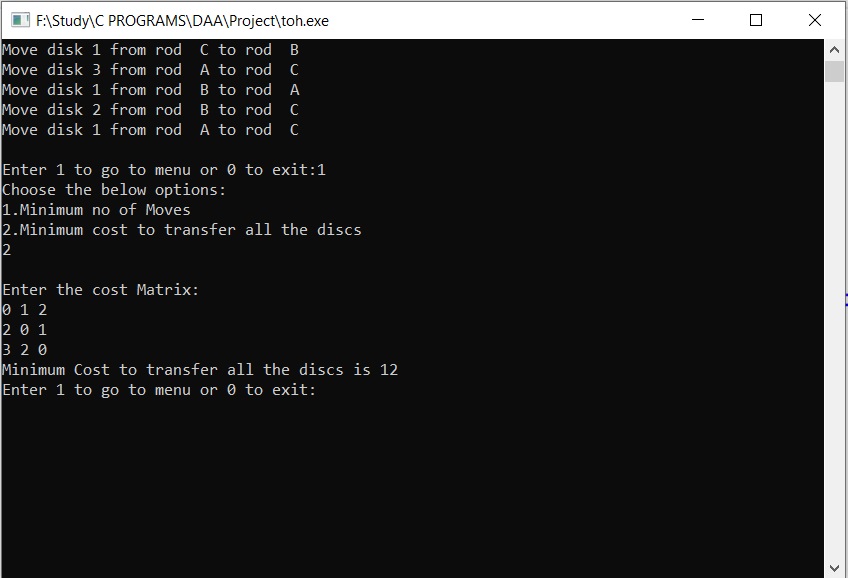
## 5.3 Output 3





## 5.4 Output 4





# **CONCLUSIONS**

The Tower of Hanoi is frequently used in psychological research on [problem solving](https://en.wikipedia.org/wiki/Problem_solving). There also exists a variant of this task called [Tower of London](https://en.wikipedia.org/wiki/Tower_of_London_Test) for neuropsychological diagnosis and treatment of executive functions.

The Tower of Hanoi is also used as a [backup rotation scheme](https://en.wikipedia.org/wiki/Backup_rotation_scheme) when performing computer data [backups](https://en.wikipedia.org/wiki/Backups) where multiple tapes/media are involved.

As mentioned above, the Tower of Hanoi is popular for teaching recursive algorithms to beginning programming students. A pictorial version of this puzzle is programmed into the [emacs](https://en.wikipedia.org/wiki/Emacs) editor, accessed by typing M-x hanoi. There is also a sample algorithm written in [Prolog](https://en.wikipedia.org/wiki/Prolog" \o "Prolog).

The Tower of Hanoi is also used as a test by neuropsychologists trying to evaluate [frontal lobe](https://en.wikipedia.org/wiki/Frontal_lobe) deficits.[[26]](https://en.wikipedia.org/wiki/Tower_of_Hanoi#cite_note-26)

In 2010, researchers published the results of an experiment that found that the ant species [Linepithema humile](https://en.wikipedia.org/wiki/Linepithema_humile" \o "Linepithema humile) were successfully able to solve the 3-disk version of the Tower of Hanoi problem through non-linear dynamics and pheromone signals.[[27]](https://en.wikipedia.org/wiki/Tower_of_Hanoi#cite_note-27)

In 2014, scientists synthesized multilayered palladium nanosheets with a Tower of Hanoi like structure.

# **REFERENCES**

* <https://en.wikipedia.org/wiki/Tower_of_Hanoi>
* <https://www.geeksforgeeks.org/cost-based-tower-of-hanoi/?ref=rp>
* <https://www.tutorialspoint.com/data_structures_algorithms/dynamic_programming.htm>