Ain Shams University – Faculty of Engineering I-CHEP

Tower of Hanoi Report

CSE245 - Advanced Algorithms and Complexity

April 24, 2025

Submitted to

Dr. Gamal Ebrahim

Eng. Sally Shaker

Submitted by:

- 23P0303 : Ahmed Mohamed Fahmy
- 23P0305 : Ahmed Mohamed Naguib
- 23P0288 : Yousef Amr Said
- 23P0294 : Omar Yasser Alashker
- 23P0258 : Zeyad Tamer Darwish
- 23P0387 : Rosette Atteya Isaac

TABLE OF CONTENT

1.0 Introduction

The Tower of Hanoi is a well-known puzzle where disks must be moved between pegs following specific rules. While the classic version uses three pegs, the four peg version, known as Reve's Puzzle, is significantly more complex and requires smarter strategies to minimize moves.

This report focuses on solving the four-peg puzzle with eight disks, aiming to reach the known optimal solution of 33 moves. Two approaches are explored: a Divide and Conquer method and a Dynamic Programming algorithm based on the Frame-Stewart heuristic.

The report includes problem analysis, implementation, and a comparison of both methods in terms of correctness, performance, and efficiency.

1.1 Problem Statement

Task 3

There are eight disks of different sizes and four pegs. Initially, all the disks are on the first peg in order of size, the largest on the bottom and the smallest on the top. Use divide and conquer methods to transfer all the disks to another peg by a sequence of moves. Only one disk can be moved at a time, and it is forbidden to place a larger disk on top of a smaller one. Does the Dynamic Programing algorithm can solve the puzzle in 33 moves? If not, then design an algorithm that solves the puzzle in 33 moves. Then design a Dynamic Programing algorithm to solve any number of disks of different sizes and four pegs puzzle. Discuss the challenge: Can the puzzle be solved in 33 moves?

Rules of the Puzzle:

- Only one disk can be moved at a time.
- A disk can only be placed on an empty peg or on top of a larger disk.
- No larger disk may be placed on a smaller disk.
- All disks start on the first peg in descending size (largest at the bottom).

Goals:

- Determine whether the puzzle with 8 disks can be solved in **33 moves** using **Dynamic Programming.**
- If not, design an optimized algorithm that achieves the 33-move solution.
- Implement two algorithms:
  - **Divide and Conquer (DAC)**
  - **Dynamic Programming (DP)**
- Extend the DP algorithm to solve the four-peg Tower of Hanoi problem for **any number of disks.**

1.2 Theoretical Background

- **Minimum Moves Formula for 4 Pegs**
    - The minimum number of moves T(n) for n disks using 4 pegs is calculated using: $T(n) = \min \{1 \leq k < n\} [2T(k) + 2^{(n-k)} - 1]$
    - Based on existing research and computed results, the optimal number of moves to solve the 4-peg Tower of Hanoi with 8 disks is 33 moves.
    - Any algorithm that aims to solve this problem optimally must match or improve upon this move count.
- **Importance of the Fourth Peg:**

The additional peg allows the puzzle to be solved in fewer moves than with just three pegs by enabling better intermediate storage and transitions.

- **Why 33 Moves for 8 Disks?**

Based on existing research and computed results, the optimal number of moves for solving the 4-peg Tower of Hanoi with 8 disks is 33 moves. Any algorithm that aims to be optimal must match this move count.

2.0 Algorithm Design

2.1 Divide and Conquer (DAC) Approach

The Divide and Conquer (DAC) method implemented here recursively reduces the problem by removing the top n - 2 disks, then moving the bottom two disks in a fixed sequence and finally solving the remaining subproblem. It makes use of all four pegs but does not compute the optimal number of disks to move at each step (k).

The basic idea is:

1. Recursively move n - 2 disks to a temporary peg.
2. Move the bottom two disks to the target peg using three steps.
3. Move the n - 2 disks from the temporary peg to the destination.

This approach is easier to implement but may result in a suboptimal solution.

2.2 Dynamic Programming (DP) Approach

The DP approach for the Tower of Hanoi with four pegs uses the Frame-Stewart algorithm. The recurrence relation is $T(n) = \min \{1 \leq k < n\} [2T(k) + 2^{(n-k)} - 1]$

The approach works as follows:

- **Find the optimal k**: For each n, evaluate all possible values of k and choose the one that minimizes the number of moves.
- **Memorization**: Store results in kMemo (optimal k values) and movesMemo (minimum moves for each n) to avoid redundant calculations.
- **Divide the problem**: The problem is split into three subproblems:
    1. Move k disks to an intermediate peg.
    2. Move the remaining n - k disks to the destination peg.
    3. Move the k disks from the intermediate peg to the destination peg.
- **Reuse results**: Use previously computed values from movesMemo and kMemo to solve larger problems efficiently.

This method ensures the minimum number of moves is achieved by reusing computed results and applying the Frame-Stewart recurrence efficiently.

3.0 Implementation

- **Programming Language**: C++
- **Main Header File**: TowerOfHanoi.h
- **Function Implementation**: TowerOfHanoi.cpp

Key Functions:

- moveDisk(from, to): Moves the top disk from one peg to another.
- solve3Peg(n, from, to, aux): Solves the classic 3-peg Tower of Hanoi recursively.
- solveDAC(n, from, to, tempPeg1, tempPeg2): Applies the Divide and Conquer strategy for 4 pegs.
- solveDP(n, from, to, tempPeg1, tempPeg2): Solves using Dynamic Programming and memorization based on the Frame-Stewart algorithm.
- optimalK(n): Determines and memorize the best value of k that minimizes moves.
- calcMinMoves(n): Recursively computes and memorize the minimum number of moves required for n disks.

Data Structures:

- **vector<stack<int>> pegs:** Represents the four pegs and the disks on them.
- **numDisks:** Total number of disks in the current problem instance.
- **moves:** Counter for the number of moves made during the solution.
- **minMoves:** Stores the minimum number of moves required to solve the problem optimally
- **movesMemo[21]:** Stores the minimum number of moves for each n (used in DP).
- **kMemo[21]:** Stores the optimal k value for each n to avoid recalculating.

3.1 Algorithms Analysis

3.1.1 Results and Output

The program was executed for 8 disks starting on peg 0 and ending on peg 3. The output shows the number of moves and confirms the correctness of each method:

- **DAC Approach:** 49 moves (not optimal)
- **DP Approach:** 33 moves (matches known optimal solution)

This verifies that the Dynamic Programming algorithm using the Frame-Stewart heuristic achieves the best possible result for the 4-peg Tower of Hanoi with 8 disks.

3.1.2 Comparison of DAC and DP

Criteria	**Divide and Conquer (DAC)**	**Dynamic Programming (DP)**
Total Moves (n=8)	49	33 (Optimal)
Time Complexity	$O(2^n)$	$O(n^2)$
Code Complexity	Simpler recursive logic	More complex (uses memorization)
Performance	Slower for large `n`	Faster and scalable
Optimality	Not guaranteed	Guaranteed optimal

Observation: The DAC method is simpler but inefficient for large n. The DP method is more efficient and suitable for larger problems, though it uses more memory.

4.0 Conclusion

In this report, we analyzed and implemented two algorithms to solve the 4-peg Tower of Hanoi puzzle with eight disks: a Divide and Conquer (DAC) approach and a Dynamic Programming (DP) approach based on the Frame-Stewart heuristic. The DAC method, while easier to implement, resulted in a non-optimal solution with 49 moves. In contrast, the DP algorithm successfully achieved the optimal solution in 33 moves, as verified by known results.

Through this comparison, it is evident that Dynamic Programming is a more efficient and scalable strategy for solving the 4-peg Tower of Hanoi problem, especially for larger values of n, due to its reduced time complexity and reuse of subproblem solutions. This study highlights the importance of choosing the right algorithmic strategy to balance simplicity, performance, and optimality.