Correspondence

A Simple Iterative Algorithm for the Towers of Hanoi Problem

Fouad B. Chedid and Toshiya Mogi

***Abstract*—A simple iterative optimal algorithm for the towers of Hanoi problem is presented.**

I. Introduction

The towers of Hanoi game is well known [2]–[4]. The basic version, a favorite example for many authors, is often used in introductory textbooks on computer programming to demonstrate the elegance of writing recursive code. However, it is known that recursion has its cost; that is, 1) the additional time needed by the system to manage the creation and deletion of numerous activation records at run time, and 2) the upper bound imposed on the size of the problem instance that can be handled; subject to the physical size of the run time stack on a particular machine. In this paper, we present a simple iterative optimal algorithm ITERATIVE_HANOI to solve the towers of Hanoi problem. Besides being simple, ITERATIVE_HANOI uses very little space. For an instance of n disks, it uses $2n$ memory cells.

Following [1], the game is played with three pegs and a pile of disks of different sizes. We start with all disks stacked on one peg, as shown in Fig. 1.

The object of the game is to move the entire stack from peg 1 to peg 3, while obeying the two rules:

1) Only one disk can be moved at a time.
2) A larger disk can never go on top of a smaller one.

The towers of Hanoi problem is to write an algorithm that moves a stack of height n from peg 1 to peg 3. We assume that the disks are numbered 1 through n from top to bottom. If we let $f(n)$ be the smallest number of moves required to move n disks from one peg to another. Then, the smallest number of moves required to move n disks from peg 1 to peg 3 is equal to $2f(n-1)+1$; that is, move $(n-1)$ disks from peg 1 to peg 2, move the nth disk from peg 1 to peg 3, and move $(n-1)$ disks from peg 2 to peg 3. Solving the recurrence relation gives $f(n) = 2^n - 1$.

II. Abstraction

Given the number of disks n, we construct a full binary tree as follows: The root node involves disk number n, the children of the root node involve disk number $(n-1)$, the children of the children of the root node involve disk number $(n-2), \cdots$, and the leaves nodes involve disk number 1. Following the argument presented in Section I (on computing the number of moves), the root node should represent the step halfway through the game, and it should be labeled something like [1–3] to, say, Move disk n from peg 1 to peg 3. Next, if a node is labeled $[x\text{-}y]$, then its left child and right child nodes will be labeled $[x\text{-}z]$ and $[z\text{-}y]$, respectively $(1 \le x, y, z \le 3)$. For example, we have drawn the tree for $n = 3$ in Fig. 2.

Manuscript received April 27, 1994; revised March 7, 1995.
The authors are with the Temple University Japan, Tokyo 192-03, Japan.
Publisher Item Identifier S 0018-9359(96)04256-2.

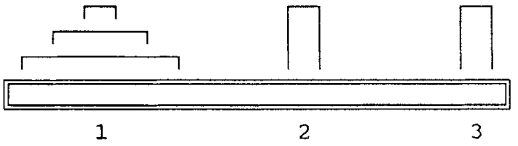

Fig. 1. General display of the towers of Hanoi game.

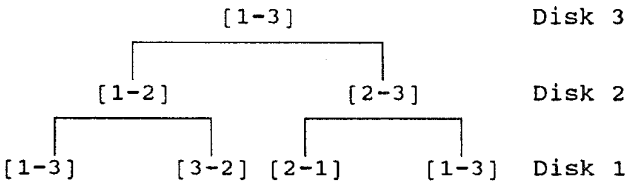

Fig. 2. Example of the abstract tree for $n = 3$.

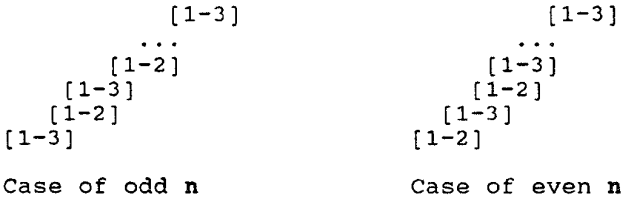

Fig. 3. Initial moves of all disks for odd/even n.

It is clear that an inorder traversal of the constructed tree is all that is needed to solve the tower of Hanoi problem. The complete algorithm, which implements this abstraction, is presented next.

III. ITERATIVE_HANOI

Given the number of disks n, ITERATIVE_HANOI uses two one-dimensional arrays, FromPeg and ToPeg, each of size n. Array element FromPeg[i] is used to describe the current location of disk i. Array element ToPeg[i] is used to describe the next location of disk i. Initially, all n disks are at peg 1, and as such FromPeg[i] is set to one, $\forall i (1 \le i \le n)$. Eventually, all FromPeg[i] $(1 \le i \le n)$ should reach the value 3; that is, to say all n disks have been moved to peg 3. As for the initial values of ToPeg, ToPeg[i] is set to what will be the first move taken by disk i $(1 \le i \le n)$. It can be easily verified that the initial values of Topeg is the sequence $3, 2, 3, 2, \cdots, 3$ if n is odd and $2, 3, 2, 3, \cdots, 3$ if n is even. Fig. 3 shows the relevant parts of the abstract trees for odd and even n.

The complete algorithm is shown below.

Procedure ITERATIVE_HANOI:

```
Let FromPeg, ToPeg be array [1 . . n] of 1 . . 3;

{ initialize array elements }
[1] Set FromPeg to {1, 1, 1, ···, 1}
[2] If n is odd then
        Set ToPeg to {3, 2, 3, 2, ···, 3}
    Else
        Set ToPeg to {2, 3, 2, 3, ···, 3}
```

```
{ Generate the moves }
[3] For i := 1 to 2^n - 1 do { repeat optimal number of times }
    begin
      { find which disk is involved at step i}
      Find the largest d (1 ≤ d ≤ n) such that i mod 2^(d-1) = 0;
      Move disk d from peg FromPeg[d] to peg ToPeg[d];

      { prepare the correct data for the next move of disk d}
      case [FromPeg[d] - ToPeg[d]] of
        [1-2]: FromPeg[d] := 2; ToPeg[d] := 3;
        [2-1]: FromPeg[d] := 1; ToPeg[d] := 3;
        [1-3]: FromPeg[d] := 3; ToPeg[d] := 2;
        [3-1]: FromPeg[d] := 1; ToPeg[d] := 2;
        [2-3]: FromPeg[d] := 3; ToPeg[d] := 1;
        [3-2]: FromPeg[d] := 2; ToPeg[d] := 1;
      end; { case }
    end; { for }
```

For example, we have traced ITERATIVE_HANOI for $n = 3$, shown below.

FromPeg	ToPeg	i	d	MOVE
1,1,1	3,2,3	1	1	move disk 1 from peg 1 to peg 3
3,1,1	2,2,3	2	2	move disk 2 from peg 1 to peg 2
3,2,1	2,3,3	3	1	move disk 1 from peg 3 to peg 2
2,2,1	1,3,3	4	3	move disk 3 from peg 1 to peg 3
2,2,3	1,3,2	5	1	move disk 1 from peg 2 to peg 1
1,2,3	3,3,2	6	2	move disk 2 from peg 2 to peg 3
1,3,3	3,1,2	7	1	move disk 1 from peg 1 to peg 3
3,3,3	2,1,2			

IV. ITERATIVE_HANOI Generates the Right Moves

We show this through two lemmas:

Lemma 1: In generating the moves for the towers of Hanoi with n disks, the disk d involved at step i $(1 \leq i \leq 2^n - 1)$ is given by the largest $d(1 \leq d \leq n)$ such that $i \bmod 2^{d-1} = 0$.

Proof: Let T_n be the abstract complete binary, which corresponds to the towers of Hanoi game with n disks (see Section II). By definition, the inorder traversal of T_n visits the root of T_n at the step exactly halfway through the traversal; that is, at step 2^{n-1}. Also at that halfway step, it is the largest disk that must be dealt with $(d = n)$ (by construction). In general, for all subtrees rooted at node $i(1 \leq i \leq 2^n - 1)$, it is the largest disk d (i a multiple of 2^{d-1}) that must be dealt with at step i.

Lemma 2: Let x, y, z be three different elements from $\{1, 2, 3\}$. If $[x\text{-}y]$ holds the data for the ist move of disk d, then the $(i+1)$st move of disk d is found in $[y\text{-}z]$.

Proof: By Construction (see Section II).

V. Concluding Remarks

We have presented a simple iterative algorithm ITERATIVE_HANOI to solve the towers of Hanoi problem. ITERATIVE_HANOI can be used in teaching to illustrate two concepts: 1) A simple way to replace recursion and 2) the relationship between abstract thinking (using trees) and actual coding (using arrays).

Acknowledgment

The authors would like to thank the anonymous referees for their helpful comments, which improved the presentation of this paper.

References

[1] D. Cooper and M. Clancy, *Oh! Pascal!* New York: Norton, 1985).
[2] J. Wu and R. Chen, "The towers of Hanoi problem with parallel moves," *Inform. Process. Lett.* no. 44, pp. 241–243, 1992
[3] M. D. Atkinson, "The cyclic towers of Hanoi," *Inform. Process. Lett.* no. 13, pp. 118–119, 1981.
[4] P. Buneman and L. Levy, "The towers of Hanoi problem," *Inform. Process. Lett.* no. 10, pp. 243–244, 1980.

System Design Via Small Mobile Robots

Jill D. Crisman, *Member, IEEE*

***Abstract*—In the Department of Electrical and Computer Engineering at Northeastern University, we have developed an upper-level course that focuses on the creative design and implementation of a complete system. In this course, the students design small mobile robots to perform a task specified at the beginning of the quarter. The designs are evaluated by how well they complete the specified task. The mobile robot systems are developed through a sequence of subsystem design projects. These subsystems are developed and demonstrated in the laboratory component of the course. The lectures focus on providing the information required to design the system, on the more practical aspects of engineering practice, and on showing how more theoretical concepts learned in other courses can be applied in practice. This course has been offered twice and we deduce from the students' designs and evaluations that it has been successful.**

I. Introduction

A four-credit course is offered in the Department of Electrical and Computer Engineering at Northeastern University that challenges teams of two students to design a small mobile robot to perform a specific task. We have offered this course for two consecutive years in the Spring quarter (immediately before graduation for many students). In the first year, the task was to have a small mobile robot track a winding path and stop at the end of the path. The path was formed by laying white masking tape on a dark surface. In the second year, we added branches on the masking tape path and the students' robots were to find a target lying on the path. For each of these tasks, there were specifications provided for the cost, minimal speed of the robot, path and target appearance, and many other constraints (this will be discussed more in Section V).

The students are free to choose any approach or components for their final robots as long as it performs the required task within the time and cost constraints. Photographs of several of the robots that have been built and task environments are shown at the end of this paper.

The benefits of design projects are known, which is why ABET requires a proportion of the undergraduate curriculum to have laboratory and experimental components. Experimental work gives students

Manuscript received March 30, 1994; revised January 8, 1995.

The author is with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115 USA.

Publisher Item Identifier S 0018-9359(96)04250-1.