ECE374 SP23 HW4

Contributors

Zhirong Chen (zhirong4)

Ziyuan Chen (ziyuanc3)

Problem 2

Consider the following variants of the Towers of Hanoi. For each variant, describe an algorithm to solve it in as few moves as possible. Prove that your algorithm is correct.

- Initially, all the n disks are on P_1, and you need to move the disks to P_2.
- You are not allowed to put a bigger disk on top of a smaller disk.

(a) Hanoi_1: You are forbidden to move any disk directly between P_1 and P_2. Exactly how many moves does your algorithm make as a function of n?

(b) Hanoi_2: You are only allowed to move disks from P_0 to P_1, from P_1 to P_2, or from P_2 to P_0. Provide an upper bound, as tight as possible, on the number of moves that your algorithm uses.

(c) Hanoi_3: The largest remaining disk disappears if there is nothing on top of it. The goal is to get all the disks to disappear. Again, provide a tight upper bound on the number of moves.

Solution

Base Algorithm

$$
\begin{aligned}
&\text{Hanoi}(n, \text{src}, \text{dst}, \text{tmp}) \\
&\qquad \text{if } n > 0 \\
&\qquad\qquad \text{Hanoi}(n-1, \text{src}, \text{tmp}, \text{dst}) \\
&\qquad\qquad \text{MoveOne}(\text{src}, \text{dst}) \\
&\qquad\qquad \text{Hanoi}(n-1, \text{tmp}, \text{dst}, \text{src})
\end{aligned}
$$

Complexity

$$
\begin{aligned}
T(n) &= 2T(n-1) + 1 \\
T(n) + 1 &= 2(T(n-1) + 1) \\
T(n) + 1 &= 2^n \\
T(n) &= 2^n - 1
\end{aligned}
$$

The algorithms are called as $\text{Hanoi}_i(n, P_1, P_2, P_0)$ in all the problem variants. src, dst, tmp are parameters instead of specific pegs.

(a) $3^n - 1$

$$
\begin{aligned}
&\mathrm{Hanoi}_1(n, \mathrm{src}, \mathrm{dst}, \mathrm{tmp}) \\
&\quad \text{if } n > 0 \\
&\qquad \mathrm{Hanoi}_1(n-1, \mathrm{src}, \mathrm{dst}, \mathrm{tmp}) \\
&\qquad \mathrm{MoveOne}(\mathrm{src}, \mathrm{tmp}) \\
&\qquad \mathrm{Hanoi}_1(n-1, \mathrm{dst}, \mathrm{src}, \mathrm{tmp}) \\
&\qquad \mathrm{MoveOne}(\mathrm{tmp}, \mathrm{dst}) \\
&\qquad \mathrm{Hanoi}_1(n-1, \mathrm{src}, \mathrm{dst}, \mathrm{tmp})
\end{aligned}
$$

Complexity

$$
\begin{aligned}
T_1(n) &= 3T_1(n-1) + 2 \\
T_1(n) + 1 &= 3(T_1(n-1) + 1) \\
T_1(n) + 1 &= 3^n \\
T_1(n) &= 3^n - 1
\end{aligned}
$$

Step	P_1	P_0	P_2
0	$[1:n]$	$\emptyset$	$\emptyset$
1	n	$\emptyset$	$[1:n-1]$
2	$\emptyset$	n	$[1:n-1]$
3	$[1:n-1]$	n	$\emptyset$
4	$[1:n-1]$	$\emptyset$	n
5	$\emptyset$	$\emptyset$	$[1:n]$

Correctness. Moves at the base case are

- $\mathrm{src} \to \mathrm{tmp} \to \mathrm{dst}$ (Steps 1, 5)
- $\mathrm{dst} \to \mathrm{tmp} \to \mathrm{src}$ (Step 3)

All the moves are legal.

(b) $O(4^n)$

$$
\begin{aligned}
&\mathrm{Hanoi}_2(n, \mathrm{src}, \mathrm{dst}, \mathrm{tmp}) \\
&\quad \text{if } n > 0 \\
&\qquad \mathrm{Hanoi}_2(n-1, \mathrm{src}, \mathrm{dst}, \mathrm{tmp}) \\
&\qquad \mathrm{Hanoi}_2(n-1, \mathrm{dst}, \mathrm{tmp}, \mathrm{src}) \\
&\qquad \mathrm{MoveOne}(\mathrm{src}, \mathrm{dst}) \\
&\qquad \mathrm{Hanoi}_2(n-1, \mathrm{tmp}, \mathrm{src}, \mathrm{dst}) \\
&\qquad \mathrm{Hanoi}_2(n-1, \mathrm{src}, \mathrm{dst}, \mathrm{tmp})
\end{aligned}
$$

Complexity

$$
\begin{aligned}
T_2(n) &= 4T_2(n-1) + 1 \\
O(n) &= 4^n
\end{aligned}
$$

Step	P_1	P_0	P_2
0	$[1:n]$	$\emptyset$	$\emptyset$
1	n	$\emptyset$	$[1:n-1]$
2	n	$[1:n-1]$	$\emptyset$
3	$\emptyset$	$[1:n-1]$	n
4	$[1:n-1]$	$\emptyset$	n
5	$\emptyset$	$\emptyset$	$[1:n]$

Correctness. Moves at the base case are

- $\text{src} \to \text{dst}$ (Steps 1, 5)
- $\text{dst} \to \text{tmp}$ (Step 2)
- $\text{tmp} \to \text{src}$ (Step 4)

All the moves are legal.

(c) $O(2^n)$

$$
\begin{aligned}
&\text{Hanoi}_3(n, \text{src}, \text{dst}, \text{tmp}) \\
&\qquad \text{if } n > 1 \quad // \text{ The final plate will disappear by itself} \\
&\qquad\qquad \text{Hanoi}(n-1, \text{src}, \text{dst}, \text{tmp}) \quad // \text{ Call the base algorithm} \\
&\qquad\qquad // \text{ The } n^{th} \text{ plate on src disappears} \\
&\qquad\qquad \text{Hanoi}_3(n-1, \text{dst}, \text{src}, \text{tmp}) \quad // \text{ Swap src and dst, recurse}
\end{aligned}
$$

Complexity

$$
\begin{aligned}
S(n) &= 2S(n-1) + 1 \quad // \text{ Base algorithm} \\
&= 2^n - 1 \\
T_3(n) &= \sum_{k=1}^{n-1} S(k) \\
&= \sum_{k=1}^{n-1} 2^k - \sum_{k=1}^{n-1} 1 \\
&= 2^n - n - 1 \\
O(n) &= 2^n
\end{aligned}
$$

Correctness. The intuition is to "toss" the upper plates back and forth between P_1 and P_2.