

## ECE374 Assignment 4

Due 03/06/2023

### Group & netid

Chen Si                  chensi3

Jie Wang                jiew5

Shitian Yang        sy39

---

### Problem 2

2. Consider the following variants of the Towers of Hanoi. For each of 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 peg 1, and you need to move the disks to peg 2. In all the following variants, you are not allowed to put a bigger disk on top of a smaller disk.

(a)

- (a) Hanoi 1: Suppose you are forbidden to move any disk directly between peg 1 and peg 2, and every move must involve (the third peg) 0. Exactly (i.e., not asymptotically) how many moves does your algorithm make as a function of  $n$ ?

Solution:

Assume the basic MOVE function is defined as *MOVE* (*from*, *to*), which moves the top one disk from peg “*from*” to peg “*to*”.

HANOI\_1 ( $n$ ):

    hanoi\_1\_helper ( $n$ , 1, 0, 2)

hanoi\_1\_helper ( $n$ , *from*, *via*, *to*):

    if ( $n=1$ ):

        MOVE (*from*, *via*)

        MOVE (*via*, *to*)

    else if ( $n>1$ ):

        hanoi\_1\_helper ( $n-1$ , *from*, *via*, *to*)                      # ( $ABC, \_, \_ \rightarrow (C, \_, AB)$ )

        MOVE (*from*, *via*)    # ( $(C, \_, AB) \rightarrow (\_, C, AB)$ )

        hanoi\_1\_helper ( $n-1$ , *to*, *via*, *from*)                      # ( $(\_, C, AB) \rightarrow (AB, C, \_)$ )

        MOVE (*via*, *to*)    # ( $(AB, C, \_) \rightarrow (AB, \_, C)$ )

        hanoi\_1\_helper ( $n-1$ , *from*, *via*, *to*)                      # ( $(AB, \_, C) \rightarrow (\_, \_, ABC)$ )

Therefore, we have the running time as  $T(n) = 3 \cdot T(n-1) + 2$ , with  $T(1) = 2$

So, the steps of dealing with  $n$  disks is  $T(n) = 3^n - 1$ .

(b)

(b) Hanoi 2: Suppose you are only allowed to move disks from peg 0 to peg 1, from peg 1 to peg 2, or from peg 2 to peg 0.

Provide an upper bound, as tight as possible, on the number of moves that your algorithm uses.

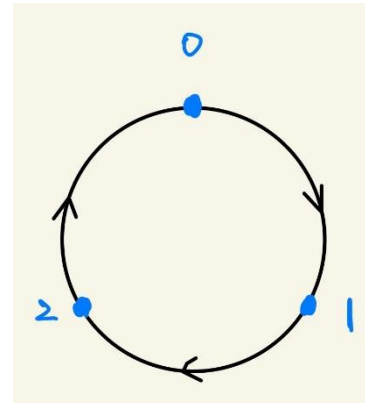
(One can derive the exact upper bound by solving the recurrence, but this is too tedious and not required here.)

Solution:

Intuition:

Consider the setup of these three pegs as left image.

Let  $Q(n)$  be the min number of steps to move  $n$  disks to the next peg,  $T(n)$  be the min number of steps to move  $n$  disks to the second next peg.



We have:

(1)  $Q(n)$

To move  $n$  pegs to the next peg (e.g.  $0 \rightarrow 1$ ), we need to:

- (a) Move  $n-1$  disks from 0 to 2  $\rightarrow T(n-1)$ ;
- (b) Move the largest disk from 0 to 1  $\rightarrow 1$ ;
- (c) Move  $n-1$  disks from 2 to 1  $\rightarrow T(n-1)$

Therefore,  $Q(n) = 2T(n-1) + 1$

(2)  $T(n)$

To move  $n$  pegs to the second next peg (e.g.  $0 \rightarrow 2$ ), we need to:

- (a) Move  $n-1$  disks from 0 to 2  $\rightarrow T(n-1)$ ;
- (b) Move the largest disk from 0 to 1  $\rightarrow 1$ ;
- (c) Move  $n-1$  disks from 2 to 0  $\rightarrow Q(n-1)$ ;
- (d) Move the largest disk from 1 to 2  $\rightarrow 1$ ;
- (e) Move  $n-1$  disks from 0 to 2  $\rightarrow T(n-1)$ ;

Therefore,  $T(n) = T(n-1) + 1 + Q(n-1) + 1 + T(n-1) = 2T(n-1) + Q(n-1) + 2$

Thus, the algorithm is

HANOI\_3 ( $n$ ):

$T(n, 1, 0, 2)$

$Q(n, \text{from}, \text{via}, \text{to})$ :

if ( $n=1$ ):

    MOVE ( $\text{from}, \text{to}$ )

else:

$T(n-1, \text{from}, \text{to}, \text{via})$

MOVE (from, to)

T (n-1, via, from, to)

T (n, from, via, to):

if (n=1):

MOVE (from, via)

MOVE (via, to)

else:

T (n-1, from, via, to)

MOVE (from, via)

Q (n-1, to, via, from)

MOVE (via, to)

T (n-1, from, via, to)

The run-time could be obtained by solving the equation:

$$Q(n) = 2T(n-1) + 1, Q(1) = 1$$

$$T(n) = 2T(n-1) + Q(n-1) + 2 = Q(n) + Q(n-1) + 1, T(1) = 2$$

Solving it, and we get  $Q(n) = \frac{((1+\sqrt{3})^{n+1} - (1-\sqrt{3})^{n+1})}{2\sqrt{3}} - 1$ ,  $T(n) = \frac{((1+\sqrt{3})^{n+1} - (1-\sqrt{3})^{n+1})}{2\sqrt{3}} +$

$\frac{((1+\sqrt{3})^n - (1-\sqrt{3})^n)}{2\sqrt{3}} - 1$ , and therefore  $HANOI_3(n) = T(n, 1, 0, 2) \in O((1+\sqrt{3})^n - (1-\sqrt{3})^n)$ .

As  $(1-\sqrt{3})^n$  is always between -1 and 1, we could treat it as a constant and thus  $T(n, 1, 0, 2) \in O((1+\sqrt{3})^n)$ .

(c)

(c) Hanoi 3: Finally consider the disappearing Tower of Hanoi puzzle where the largest remaining disk will disappear if there is nothing on top of it. The goal here is to get all the disks to disappear and be left with three empty pegs (in as few moves as possible).

Provide an upper bound, as tight as possible, on the number of moves your algorithm uses.

Solution:

Intuition:

HANOI\_3 (n):

from = 1

via = 0

to = 2

while (n > 2):

hanoi_3_helper (n-2, from, via, to)	# (ABCDEF, _, _) → (EF, _, ABCD)
MOVE (from, via)	# (EF, _, ABCD) → (F, E, ABCD)
DISAPPEAR (from)	# (F, E, ABCD) → (_, E, ABCD)
DISAPPEAR (via)	# (_, E, ABCD) → (_, _, ABCD)
n = n - 2	
SWAP (from, via)	

if (n = 2):

MOVE (from, via)	# (AB, _, _) → (A, B, _)
DISAPPEAR (from)	
DISAPPEAR (via)	

# if n=1, it would directly disappear

if (n=1):

DISAPPEAR (from)

# Traditional Hanoi Tower Movement

hanoi\_3\_helper (n, from, via, to):

if (n=1):

MOVE (from, to)

else if (n>1):

hanoi_3_helper (n-1, from, to, via)	# (ABC, _, _) → (C, AB, _)
MOVE (from, to)	# (C, AB, _) → (_, AB, C)
hanoi_3_helper (n-1, via, from, to)	# (_, AB, C) → (_, _, ABC)

The time complexity of this algorithm is:

$$T(n) = T(n-2) + c(n-2)^2, T(0)=0$$

Therefore, the time complexity of this algorithm has an upper bound of  $O(n^3)$ .