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, _, _)→(C, _, AB)
        MOVE (from, via)                             # (C, _, AB)→(_, C, AB)
        hanoi_1_helper (n-1, to, via, from)          # (_, C, AB)→(AB, C, _)
        MOVE (via, to)                               # (AB, C, _)→(AB, _, C)
        hanoi_1_helper (n-1, from, via, to)          # (AB, _, C)→(_, _, ABC)
```

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

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

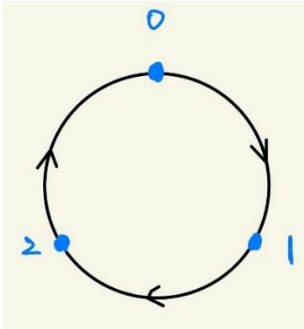
(b)

(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→1), we need to:

(a) Move n-1 disks from 0 to 2 → T(n-1);

(b) Move the largest disk from 0 to 1 → 1;

(c) Move n-1 disks from 2 to 1 → 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→2), we need to:

(a) Move n-1 disks from 0 to 2 → T(n-1);

(b) Move the largest disk from 0 to 1 → 1;

(c) Move n-1 disks from 2 to 0 → Q(n-1);

(d) Move the largest disk from 1 to 2 → 1;

(e) Move n-1 disks from 0 to 2 → 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, from, via, to):
    if (n=1):
        MOVE (from, to)
    else:
        T (n-1, from, to, 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:

$$Q(n) = \frac{\left((1+\sqrt{3})^{n+1} - (1-\sqrt{3})^{n+1}\right)}{2\sqrt{3}} - 1,\ T(n) = \frac{\left((1+\sqrt{3})^{n+1} - (1-\sqrt{3})^{n+1}\right)}{2\sqrt{3}} + \frac{\left((1+\sqrt{3})^{n} - (1-\sqrt{3})^{n}\right)}{2\sqrt{3}} - 1$$

Therefore, $\text{HANOI}_3(n) = T(n,1,0,2) \in O(\left(1+\sqrt{3}\right)^n - \left(1-\sqrt{3}\right)^n)$.

As $\left(1-\sqrt{3}\right)^n$ is always between -1 and 1, we could treat it as a constant

Thus, $T(n,1,0,2) \in O(\left(1+\sqrt{3}\right)^n)$.

(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: It is easy to point out that, the traditional Hanoi can solve the problem, with a time complexity of $O(n^2)$, **which remove one largest disk each time.**

However, it is not the most efficient way to solve the problem.

Since there are 3 pegs, we can at most remove two largest disks every time, e.g., the 1st largest one is removed at peg1, the 2nd largest one is removed at peg2, all others are left at peg3.

With this intuition, we can design the algorithm as follows:

```
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 (n, from, via, to):
    if (n=1):
        MOVE (from, to)
    else if (n>1):
        Hanoi (n-1, from, to, via)          # (ABC, _, _)→(C, AB, _)
        MOVE (from, to)                     # (C, AB, _)→(_, AB, C)
        Hanoi (n-1, via, from, to)          # (_, AB, C)→(_, _, ABC)
```

The time complexity of this algorithm is:

$T(n) = T(n-2) + Hanoi(n-2) + 1 = T(n-2) + 2^{n-2} - 1 + 1,\ T(0) = 0, T(2) = 1$

Where **Hanoi(n-2)** is the complexity of traditional Hanoi Tower

$Hanoi(n-2) = 2^{n-2} - 1 \rightarrow O(2^n)$

Referring to Saad's paper on the fifth variation of the Tower of Hanoi, we can get:

$$T(n) = \frac{(-1)^n + 2^{n+1} - 3}{6} \rightarrow O(2^n)$$

Exploding Questions

Exploding(n, x, y, z)
  if $n > 1$
    then Hanoi$(n-2, x, z, y)$
      Move$(1, x, z)$
      disks n *and* $n-1$ *explode*
      Exploding$(n-2, y, x, z)$

Given this algorithm, we establish the recurrence:

$$a_n = a_{n-2} + 2^{n-2}$$

and change it into a homogeneous one by annihilation of 2^{n-2} as follows:

$$a_n = a_{n-2} + 2^{n-2}$$

$$2 \cdot a_{n-1} = 2 \cdot a_{n-3} + 2 \cdot 2^{n-3}$$

$$a_n - 2a_{n-1} = a_{n-2} - 2a_{n-3}$$

to finally obtain $a_n = 2a_{n-1} + a_{n-2} - 2a_{n-3}$ and the characteristic equation $x^3 = 2x^2 + x - 2$. By observing that $r_1 = 1$ is a root, we express the characteristic equation as $(x-1)(x^2 - x - 2) = 0$ and solve the quadratic equation for the other two roots. The three distinct roots will be $r_1 = 1$, $r_2 = -1$, and $r_3 = 2$. Therefore, $a_n = c_1 + c_2(-1)^n + c_3 2^n$, and since

$$a_0 = c_1 + c_2 + c_3 = 0$$

$$a_1 = c_1 - c_2 - 2c_3 = 0$$

$$a_2 = c_1 + c_2 + 4c_3 = 1$$

we have $c_1 = -1/2$, $c_2 = 1/6$, and $c_3 = 1/3$. Finally,

$$a_n = \frac{(-1)^n + 2^{n+1} - 3}{6}$$

Therefore, the time complexity of this algorithm has an upper bound of $O(2^n)$.

Reference:

Saad Mneimneh (2018), *Simple Variations on the Tower of Hanoi to Guide the Study of Recurrences and Proofs by Induction*