8 Sept 2023

Warm-up:

$$T(n) = \begin{cases} c, & n = 0 \\ T(n-1) + d, & n \geq 1 \end{cases}$$

1. What is the closed form?

   $T(n) = c + nd$

2. Asymptotic Form?

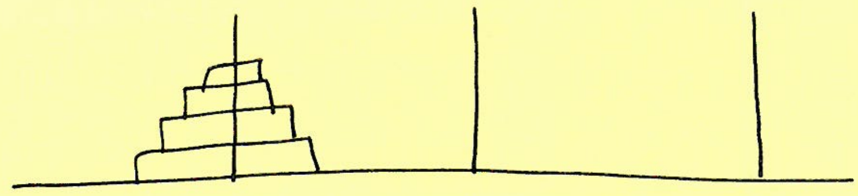
   $T(n) \in \Theta(n)$, $T(n) = \Theta(n)$, $T(n)$ is $\Theta(n)$.

3. Name a potential algorithm with this recurrence relation.

   We could write linear search as a recursion w/ this recurrence relation.

   Also, find max/min.

Towers of Hanoi



src   temp, the extra peg   dest

How do I solve this?

```
            HANOI (n, src, dest, temp)
Θ(1)    1:  if n > 0
T(n-1)  2:      HANOI (n-1, src, temp, dest)
Θ(1)    3:      move nth "disk" from src to dest
T(n-1)  4:      HANOI (n-1, temp, dest, src)
Θ(1)    5:  end if
```

Let $T : \mathbb{N} \to \mathbb{R}$ be the function that takes as input $n \in \mathbb{N}$ and returns the runtime of HANOI$(n, *, *, *)$.

"solving Hanoi of size n"

$$T(n) = \begin{cases} 0, & n = 0 \\ 2T(n-1) + \Theta(1), & n > 0 \end{cases}$$

Needs a proof! INDUCTION

closed form: $T(n) = 2^n - 1$

$T(n) \in \Theta(2^n)$ Exponential!!

TO prove asymptotic form:

1. Could prove closed form (via induction), then "just" take the asymptotic form of that.

2. TO prove directly, via def'n of Θ-notation (which will also be a proof by induction) AKA - skip the closed form.

CLASS

Recursion Invariants

CSCI 432, Fall 2021

September 8, 2023

Recursion Invariants: Proving Partial Correctness

Recall theHANOI algorithm (given in Algorithm 1); see [1, Ch. 1].

Algorithm 1 HANOI($n, src, dest, tmp$)

Input: $n \in \mathbb{N}$, and three towers with disks: $src, dest, tmp$ such that P
Output: R (see below)

```
if n > 0 then
    HANOI(n − 1, src, tmp, dest)
    move top disk from src to dest
    HANOI(n − 1, tmp, dest, src)
end if
```

1. Suppose we have n disks total. What are the assumptions on the input to the initial call HANOI($n, src, dest, tmp$)?

   Answer: NOTE: N≠n !

   Going forward, we will call these assumptions P.

2. What does it mean for $\textsc{Hanoi}(n, src, dest, tmp)$ to execute correctly? What does it return / what does it need to accomplish?

   Answer:

Going forward, we call this statement Q.

3. For a general call to the recursive algorithm what are the assumptions on the input (For convenience, suppose the call is: $\textsc{Hanoi}(k, A, B, C)$).

   Answer: k smallest discs are on peg A.

3b: What does it mean for HANOI (k, A, B, C) to execute correctly?

Note: when making a recursive call, we must justify that we have met these conditions.

4. What is the recursion invariant?

   The recursion invariant is statement (i.e., a sentence that can be evaluated to TRUE/FALSE). In particular, R is (in general) the statement, *Each recursive call* $\textsc{Hanoi}(k, A, B, C)$ *executes correctly.* What does that mean in our case? Well, it means

   - There are currently no violations of smaller disks on larger disks. (Note: the world would crumble if this were violated at any time), AND
   - the k smallest disks are now on B, AND
   - no other disks besides the k smallest have moved (since right before this call).

   We use the recursion invariant to prove INITIALIZATION, MAINTENANCE, and END. So, sometimes we may see the invariant right away. Other times, we may need to try the proofs then revisit the invariant (as we might realize that we forgot something).

5. INITIALIZATION This is like the base case of induction. Colloquially, we ask "Why is this true for the smallest input?" (And, what are those inputs that would allow us to return without a recursive call?) More formally, we can say: If $n_0 = 0$, then after the call to $\textsc{Hanoi}(n_0, A, B, C)$, the recursion invariant R is satisfied.

   Answer:

Note: sometimes, just as in induction, there may be more than one base case.

6. MAINTENANCE: This part is JUST like the inductive step of induction. Let $k \in \mathbb{N}$ such that $k \geq n_0$. Assume that, for all $k' \in \mathbb{N}$ such that $n_0 \leq k' \leq k$, the recursion invariant R holds after a call to $\textsc{Hanoi}(k', *, *, *)$. (That was the equivalent to the inductive assumption). Now, we must prove that R holds after $\textsc{Hanoi}(k+1, A, B, C)$. (Hint: for this, we will almost always need to use the line numbers as we walk through the algorithm to explain how the maintenance step works).

   Answer:

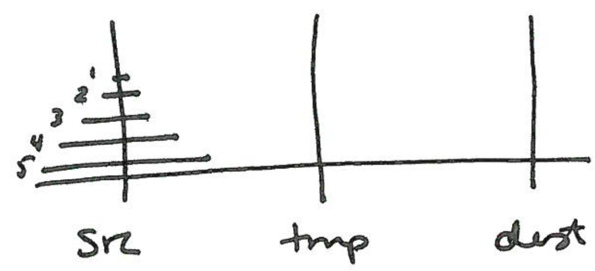
7. END: This is where we diverge from induction. Since algorithms are finite, we can't go on forever. Colloquially, we say "if the initial call $\textsc{Hanoi}(n, src, dest, tmp)$ finishes executing, then all n disks (which were initially on src) have moved to dst." More formally, we can phrase this as: If the recursion invariant holds, if the preconditions P are satisfied, and if the algorithm completes execution, then the post-condition Q is satisfied. (For shorthand, I might write $R \wedge P \wedge T \implies Q$).

   Answer:

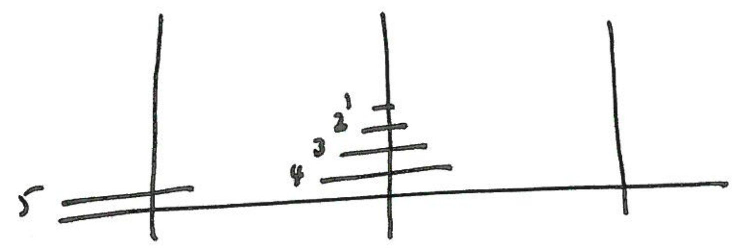
References

[1] Erickson, J. *Algorithms.* Independently published open access, June 2019.

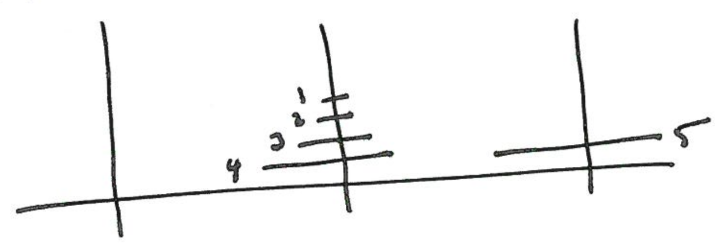
Example:



AFTER 1st recursive call: HANOI(4, src, tmp, dest)



Move Disc 5



AFTER 2nd recursive call HANOI (4, tmp, dest, src)