CIT 596: ALGORITHMS & COMPUTATION

A Recursive Algorithm: Towers of Hanoi

Iterative vs Recursive Algorithms

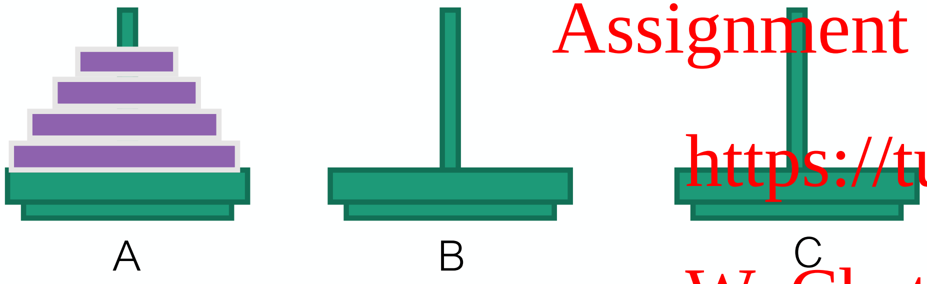
- **Iterative ("bottom-up"):** A loop structure builds up to a solution.

SumSquaresIterative(n)
    $sum = 0$
    for $i = 1$ **to** n
        $sum = sum + i^2$
    return sum

- **Recursive ("top-down"):** The algorithm calls itself on smaller instances of the same problem.

SumSquaresRecursive(n)
    if $n == 1$
        return 1
    else
        return SumSquaresRecursive($n - 1$) $+ n^2$

ANOTHER EX: TOWERS OF HANOI (T0H)



3 pegs:

N disks in decreasing size order on peg A.

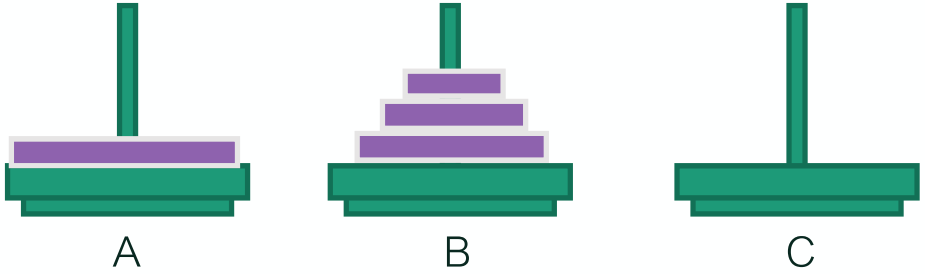
Move one disk at a time; Never place a bigger disk on a smaller disk.

Move all N disks to peg C.

- Move top $n-1$ disks from peg A to peg B
- Move disk 1 from peg A to peg C
- Move the $n-1$ disks from peg B to peg C
- Why is this correct? Largest disk at bottom of peg A or peg C does not interfere
- How long does this take?

How can this be analyzed with **induction**?

Step 1

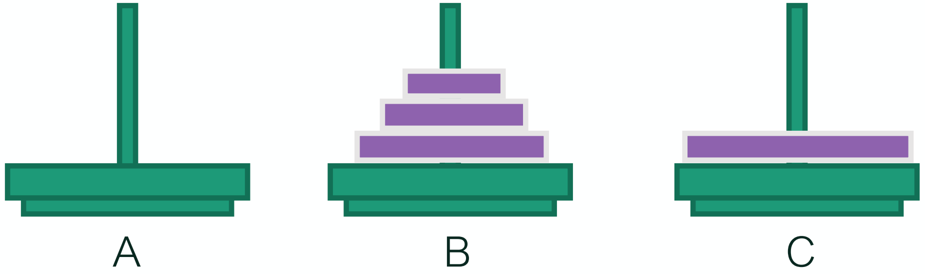


Move $n-1$ recursively

- Move top $n - 1$ disks from peg A to peg B
- Move disk 1 from peg A to peg C
- Move the $n - 1$ disks from peg B to peg C
- Why is this correct? Largest disk at bottom of peg A or peg C does not interfere
- How long does this take?

  How can this be analyzed with **induction**?

Step 2

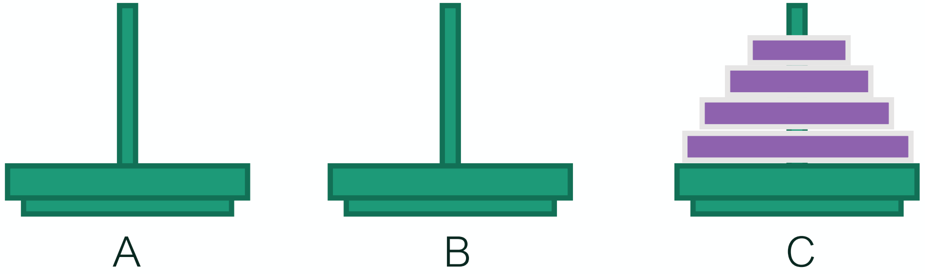


Move last brick

- Move top $n-1$ disks from peg A to peg B
- Move disk 1 from peg A to peg C
- Move the $n-1$ disks from peg B to peg C
- Why is this correct? Largest disk at bottom of peg A or peg C does not interfere
- How long does this take?

  How can this be analyzed with **induction**?

Step 3



Move $n-1$ recursively again

Specifying the Algorithm

Input: Pegs A, B, C and a number $n \geq 1$ such that A has at least n properly ordered rings on top.

Output: The top n rings from A have been moved to C, and those rings have never been placed out of order with each other.

HANOI(A, B, C, n)

    if $n == 1$
        move the top ring on A to C
    else
        HANOI$(A, C, B, n-1)$
        HANOI$(A, B, C, 1)$
        HANOI$(B, A, C, n-1)$

A Proof Technique for Recursive Algorithms

- For each $n \geq 1$, define $P(n)$, a careful statement of what it means for the algorithm to be correct on inputs of size n.
- **Base case:** Directly prove that $P(1)$ is true.
- **Inductive step:** Prove that for all $n \geq 2$, if $P(1)$ through $P(n-1)$ are all true, then $P(n)$ is also true.
  - Let n be an arbitrary integer ≥ 2.
  - Assume $P(1), \ldots, P(n-1)$. **(IH)**
  - Use the IH to prove $P(n)$.
- Conclude that $P(n)$ is true for all $n \geq 1$.

Comparing to Loop Invariant Technique

Although both use induction, the induction hypothesis has a different form.

- IH for iterative algorithms: "Partway through its execution, the algorithm is on track to be correct."

- IH for recursive algorithms: "The algorithm is correct on all smaller inputs."

Proof of Correctness for HANOI

Input: Pegs A, B, C and a number $n \geq 1$ such that A has at least n properly ordered rings on top.

Output: The top n rings from A have been moved to C, and those rings have never been placed out of order with each other.

For each $n \geq 1$, define the statement $P(n)$: *If the input condition is met for an input of size n, then the output conditions will be met.*

Base case: When $n = 1$, we move A's top ring to C. The input condition states that A has at least one ring, so there is a top ring to move, and it is impossible for a single ring to be out of order with itself. This proves $P(1)$.

Proof of Correctness for HANOI

Inductive step: Fix $n \geq 2$, and assume $P(1), \ldots, P(n-1)$.

Suppose we call $\text{Hanoi}(A, C, B, n)$ and that the input condition is met. By IH, $\text{Hanoi}(A, C, B, n-1)$ moves the top $n-1$ rings from A to B while maintaining proper order among those rings. After that, the top ring on A is larger than those $n-1$ rings.

By IH $\text{Hanoi}(A, B, C, 1)$ moves A's top ring to C, and then $\text{Hanoi}(B, A, C, n-1)$ moves B's top $n-1$ rings to C, maintaining proper order among those rings. Since the ring moved from A to C was larger than those $n-1$ rings, proper order was maintained for all n rings. Thus, $P(n)$ is true.

By mathematical induction, we conclude that $P(n)$ is true for all $n \geq 1$. The algorithm is therefore correct.