CM1035 Topic 8 Recursion

Ian Sanders

April 2025 Session

UNIVERSITY OF LONDON

Recap

In the module videos we have seen

- decrease and conquer (factorial, etc.),
- a recursive algorithm for linear search,
- a recursive algorithm for bubble sort,
- a recursive algorithm for insertion sort, and
- some discussion of the call stack.

In this webinar we will look at take another look at some of these concepts.

UNIVERSITY OF LONDON

Consider the process of reversing the order of the letters in an English word.
For example turning star into rats.
This can be done by removing the first letter of the word, reversing the rest of the word and then appending the letter which was removed.

```
00   remove first letter
00   reverse rest of word
00   append removed letter
```

UNIVERSITY OF LONDON

Reversing a word

We would then do the operation reverse rest of word in exactly the same way except that we would be using a shorter word.
This process is a recursive operation – it describes how to reverse a sequence of letters in terms of how to reverse a slightly shorter sequence of letters.
In this case on each recursive call a single letter is removed, thus simplifying the function's input.
The limiting case (or base case) is when the word has only one letter.
At this point there is no further recursive call.
The recursion then bubbles out.

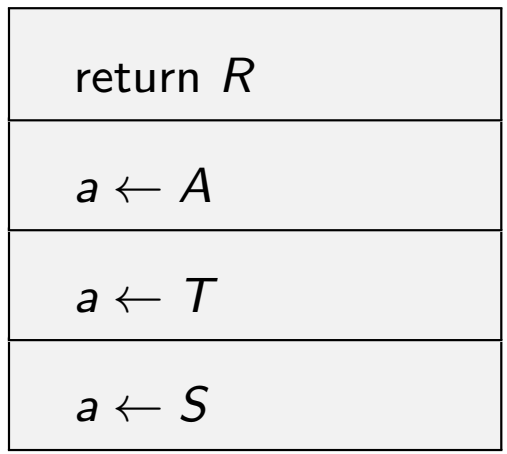
Reversing a word

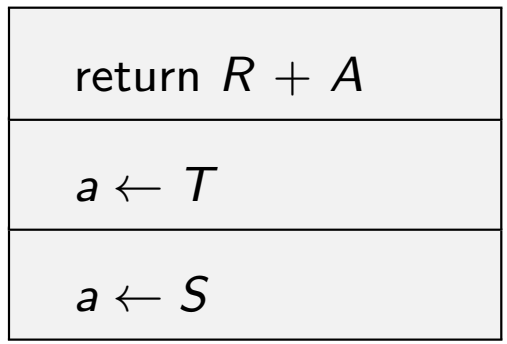
The complete algorithm for reversing a sequence of letters is

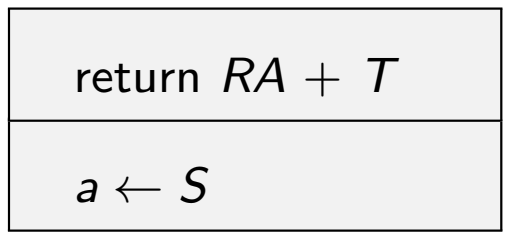
```
function ReverseWord
    Enter word
    Reverse(word)

function Reverse (sequence)
    IF sequence contains only one letter
        THEN
            return sequence
        ELSE
            a ← first letter of sequence
            new ← Reverse (rest of sequence)
            return new concatenated with a
```

UNIVERSITY OF LONDON

The Call Stack







return $RAT + S$

Reversed word is RATS

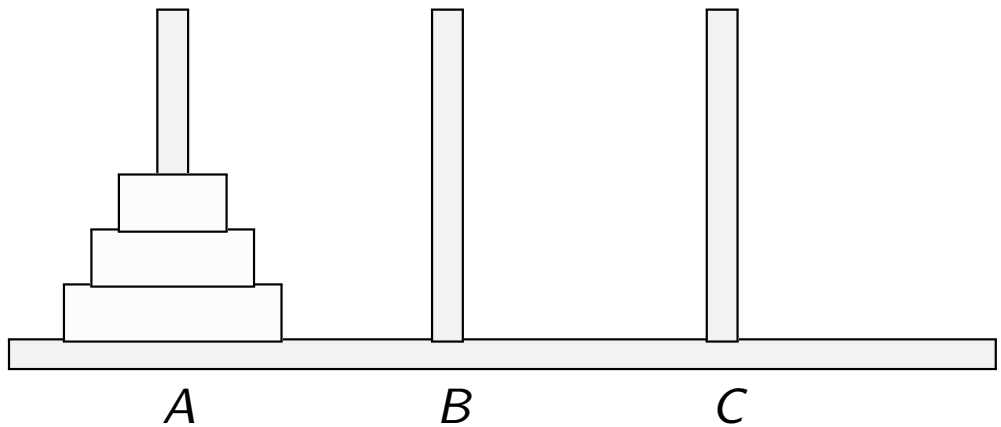
The Towers of Hanoi

The ancient puzzle known as the *Towers of Hanoi* originated in a monastery in Tibet.
The puzzle is as follows:
Suppose we are given three towers, or three pegs, A, B and C.
On the first peg there a three rings of decreasing size (the largest at the bottom, the smallest at the top).
The other pegs are empty.
The object of the game is to move the rings from peg A to peg B and have them in the same order on peg B.
The rules are:

- Peg C may be used in the moving process,
- Rings are moved one at a time
- At no instant may a larger ring be atop a smaller one

UNIVERSITY OF LONDON

The towers of Hanoi



The Towers of Hanoi

A recursive algorithm to solve the problem:

```
function Towers
   Enter Number
   move(Number, A, B, C)

funtion move(N, X, Y, Z)
   If N = 1
      THEN
         output X → Y
      ELSE
         move(N-1, X, Z, Y)
         output X → Y
         move(N-1, Z, Y, X)
```

This algorithm will not actually make the moves it will simply print out a list of moves which will solve the puzzle.

UNIVERSITY OF LONDON

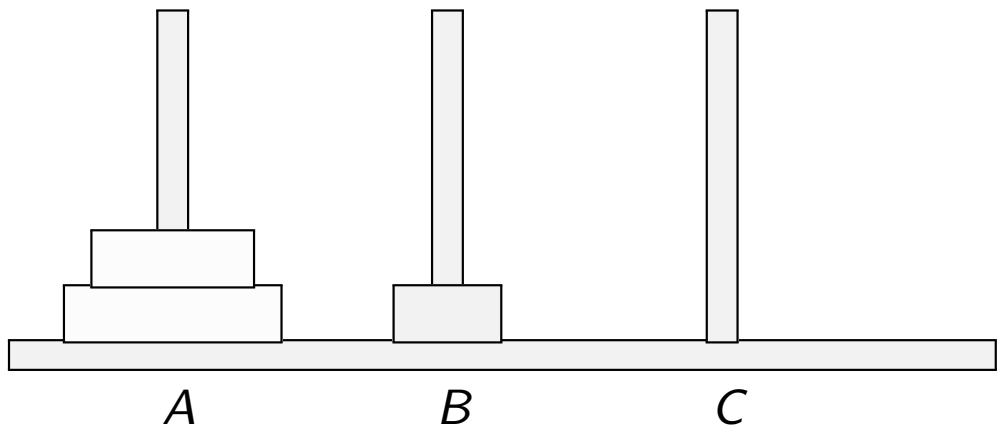
The Towers of Hanoi

Solving the three disk problem:
Entering function
Moving 3 disks from A to B using C
→ Entering function
→ Moving 2 disks from A to C using B
→ → Entering function
→ → Moving 1 disks from A to B using C
→ → Move A to B
→ → Exiting function
→ Move A to C
→ → Entering function
→ → Moving 1 disks from B to C using A
→ → Move B to C
→ → Exiting function
→ Exiting function
Move A to B

→ Entering function
→ Moving 2 disks from C to B using A
→ → Entering function
→ → Moving 1 disks from C to A using B
→ → Move C to A
→ → Exiting function
→ Move C to B
→ → Entering function
→ → Moving 1 disks from A to B using C
→ → Move A to B
→ → Exiting function
→ Exiting function
Exiting function

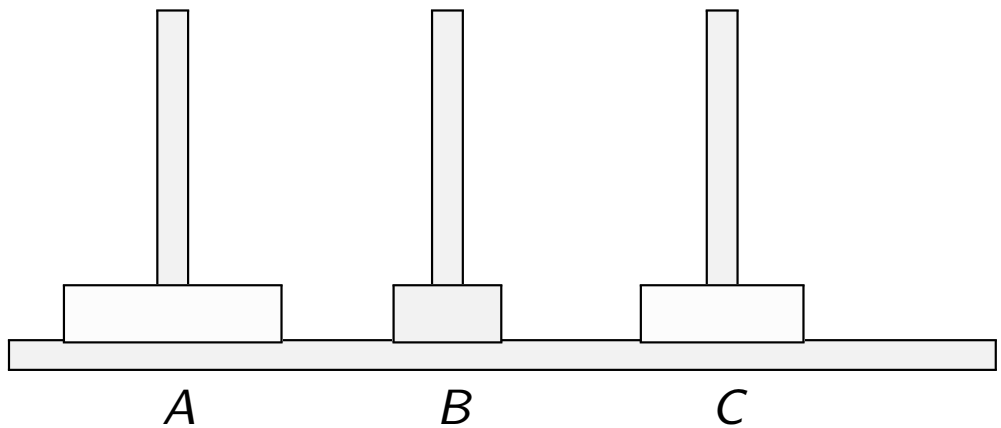
The towers of Hanoi – 3 disk example

A to B



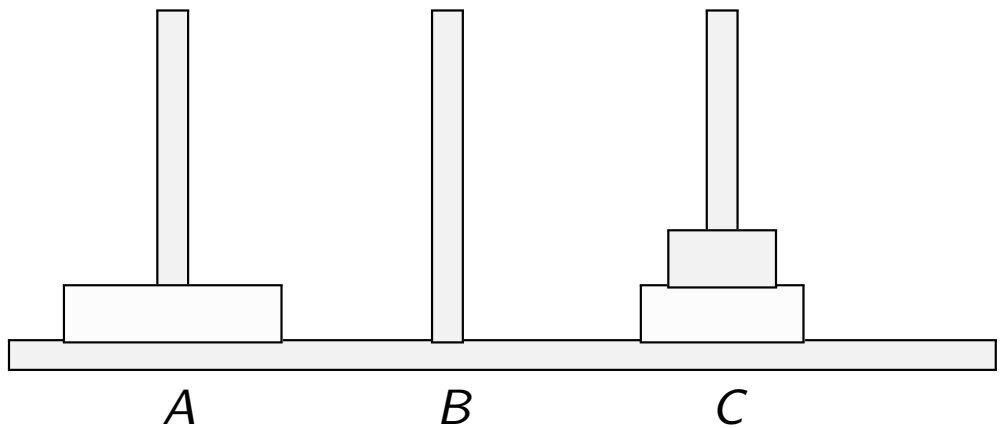
The towers of Hanoi – 3 disk example

A to C



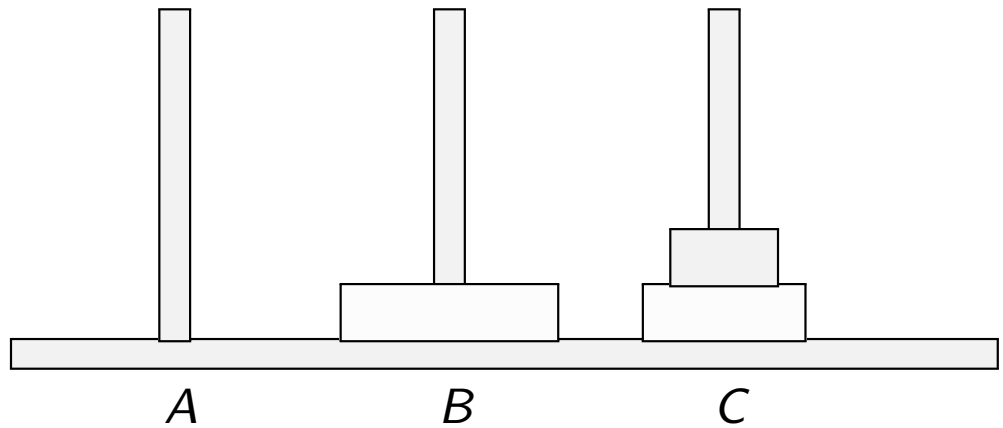
The towers of Hanoi – 3 disk example

B to C



The towers of Hanoi – 3 disk example

A to B



And so on.

This solution is again a *decrease and conquer* recursive solution. The original problem is solved in terms of solving two slightly smaller problems – each with one fewer disk.

Exercise: Make sure you understand how the recursive algorithm given above works.
Exercise: If you are keen, try to implement the Towers of Hanoi problem iteratively.

Finally

I will put the slides up on my webpage.
I will put a small JS program of the Towers of Hanoi problem on my webpage.
I will put the webinar recording on the webpage when I get it.
If there are any questions, please feel free to post on the discussion forum.

UNIVERSITY OF LONDON