Recursion II

CS 2110: Software Development Methods

April 10, 2019

Recursion

- Recursion breaks a difficult problem into one or more simpler versions of itself
- A definition is **recursive** if it is defined in terms of itself
- Questions to ask yourself:
    - How can we reduce the problem into smaller version of the same problem?
    - How does each call make the problem smaller?
    - What is the **base case**?
    - Will we always reach the base case?

Definitions

Base case

The case for which the solution can be stated nonrecursively

Recursive case

The case for which the solution is expressed in terms of a smaller version of itself

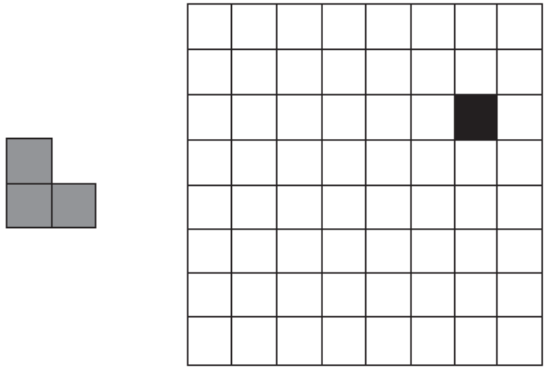
Views of Recursion

- **Recursive definition**: definition in terms of itself, such as $n! = n \times (n-1)!$
- **Recursive procedure**: a procedure that calls itself
  ex: `factorial(int n)`
- **Recursive data structure**: a data structure that contains a pointer to an instance of itself:

```
public class ListNode {
    Object nodeItem;
    ListNode next, previous;
    ...
}
```

Tromino Puzzle

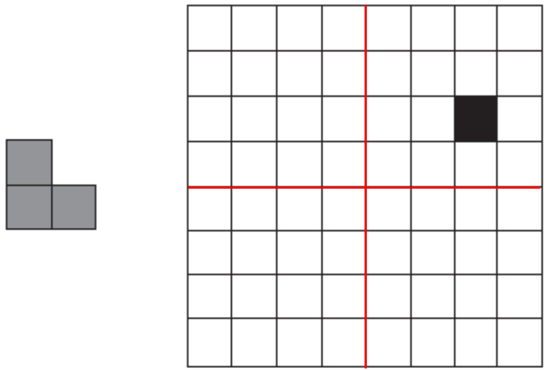
Given an 8×8 board with one piece missing, tile with L-shaped trominoes.



Interactive: http://goo.gl/npFQUD

Tromino Puzzle

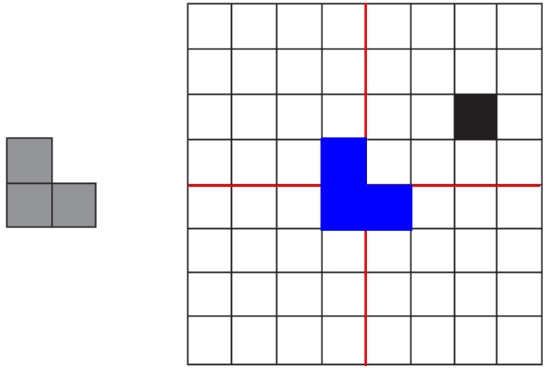
Given an 8×8 board with one piece missing, tile with L-shaped trominoes.



Interactive: http://goo.gl/npFQUD

Tromino Puzzle

Given an 8×8 board with one piece missing, tile with L-shaped trominoes.



Interactive: http://goo.gl/npFQUD

Other Recursive Examples

- Towers of Hanoi
- Euclid's Algorithm
- Fractals

Towers of Hanoi

The objective is to transfer entire tower A to the peg B, moving only one disk at a time and never moving a larger one onto a smaller one

- The algorithm to transfer n disks from A to B in general: We first transfer $n-1$ smallest disks to peg C, then move the largest one to the peg B and finally transfer the $n-1$ smallest back onto largest (peg B)
- The number of necessary moves to transfer n disks can be found by $T(n) = 2^n - 1$

Euclid's Algorithm

Calculating the greatest common divisor (gcd) of two positive integers is the largest integer that divides evenly into both of them

- E.g. greatest common divisor of 102 and 68 is 34 since both 102 and 68 are multiples of 34, but no integer larger than 34 divides evenly into 102 and 68
- Logic: If p > q, the gcd of p and q is the same as the gcd of q and $p\%q$ (where % is the remainder operator)
- Stop recursion once q becomes zero; at which point return p

Summary

- Recursion breaks a difficult problem into one or more simpler versions of itself
- Recursive definition: A definition in which something is defined in terms of smaller versions of itself
- Recursive problem can be broken into two parts:
  - Base case: The case for which the solution can be stated nonrecursively
  - Recursive case: The case for which the solution is expressed in terms of a smaller version of itself

Summary

Recursion is tricky!

- Always put base case first
- Base case should eventually happen given any input
- Recursive solution may not always be the best