CSD101: Introduction to computing and programming (ICP)

Recursion

- Recursion is a mechanism that allows us to define a function in terms of earlier values of the same function together with one or more base cases that are defined explicitly.
- A standard example is the factorial function:

```
factorial(0) = 1
factorian(n) = n * factorial(n-1)
```

Another example is the Fibonacci function:

```
Fibonacci(0) = 1
Fibonacci(1) = 1
Fibonacci(n) = Fibonacci(n-1) + Fibonacci(n-2)
```

- The base cases are important since they allow a recursion to terminate. Otherwise the recursion can go on infinitely or till overflow/underflow halts the program.

Recursion can be expensive

- Recursion can be computationally expensive.
- Recursive version of finding the n^{th} Fibonacci number can lead to exponentially many calls to the base cases due to tree recursion.
- Iteration should be preferred in such cases.
- Tail recursion - refers to the case when the recursive call is the last call inside the recursive function. This is equivalent to iteration in terms of computational cost. See examples of Fibonacci functions in the code.

Recursion can be elegant

- Tower of Hanoi problem - the recursive solution is extremely elegant and easy to understand - see code.
- An iterative solution is very hard to conceptualize and write.
- Similarly, quick sort and merge sort algorithms are easy to write and understand - see code. They have a clean divide-and-conquer structure.