PH502: Scientific Programming Concepts

Irish Centre for High End Computing (ICHEC)

September 23, 2020

Overview

ICHEC
Irish Centre for High-End Computing

- When designing algorithms there are some more general classifications.
- These classes are not strictly about time complexity but of course there will be a relation.

Classification of Algorithms

ICHEC
Irish Centre for High-End Computing

- Iterative vs Recursive
  - Iterative: algorithms which use of loops
  - Recursive: algorithms which invoke itself repeatedly until a condition matches
- Serial vs Parallel vs Distributed
  - Serial: steps are executed in sequence, one at a time
  - Parallel: several steps are executed at the same time by the same processor
  - Distributed: steps are executed by different processors

Classification of Algorithms

- Deterministic vs Nondeterministic
  - Deterministic: every step in the algorithm is predictable
  - Nondeterministic: the steps vary from execution to execution
- Exact vs Approximate
  - Exact: algorithm reaches the solution
  - Approximate: algorithm is not guaranteed to reach the solution, but seeks an approximation to the solution

Example: Factorial

Problem: Find $n!$ for $n \geq 1$.

- The factorial of an integer n is the product of all integers in range $[1, n]$.

$$n! = 1 \times 2 \times 3 \times 4 \times \cdots \times (n-1) \times n$$

Factorial - Iterative Algorithm

- For each integer in range $[1, n]$, aggregate the product and output the final result.

$$n! = \prod_{i=1}^{n} i$$

IterativeFactorial(n)
result=1
for i = 2 to n **do**
    result = result $\times i$
end for
return result

- $\Theta(n)$

Factorial - Recursive Algorithm

- Factorial of n is n times the factorial of $(n-1)$, unless n is 0, for which the result is 1.

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ (n-1)! \times n & \text{if } n > 0. \end{cases}$$

RecursiveFactorial(n)
if $n = 0$ **then**
  return 1
else
  return $n\times$ RecursiveFactorial($n-1$)
end if

- $\Theta(n)$

Recursive Functions

- C has no distinction between recursive and non-recursive functions.
- In Fortran you must explicitly state that the function is recursive.

```
program rf
  implicit none
  interface
     recursive function f(n) result(answer)
        integer (kind=4) :: n,answer
     end function
  end interface

  write(6,*) f(10)
end program

recursive function f(n) result(answer)
  integer (kind=4) :: n,answer
  answer = 1
  if (n .gt. 0) answer = n * f(n-1)
  return
end function
```

Summary

- This week we discussed:
  1. Algorithms
  2. and their properties,
  3. time complexity,
  4. classifications.