Tutorial 2

Introduction

In this tutorial, students practice and improve their problem solving skill. Several problems are introduced with the example solutions. Students are asked to implement the algorithms and analyze the time complexity of these algorithms. In the exercise sections, students will try to improve the algorithms to get faster running time version.

Examples

1. Example 01 – Searching in an ordered array.
An ordered array is an array of integers that all elements are sorted:

A[1]<A[2]<A[3]<A[4]<…<A[n]

For an integer M, we want to determine if there exists an integer i such that A[i]=M.
The following algorithm (**algorithm 2.1**) solve this problem.

```
Algorithm: SimpleSearch(A,n,m)
Input: An array A stores n sorted integer. An integer m.
Output: An integer i that a[i]=m.
           -1 if m doesn't appear in the array.

For i ← 0 to n-1 do
   If (a[i] = m) then
      Return i;
```

Student should create a class that implements the algorithm 2.1 above, with the following functions (please refer to class ***SimpleSearch*** in the tutorial source code project):

- Ask user to input n, array A and an integer m using the keyboard
- Show the searching result

What is Big-Oh of the algorithm 2.1?

2. Example 02 – Evaluating the value of a polynomial

A polynomial of degree n is a function which has the following form:

$$f(x) = \sum_{i=0}^{n} c_i x^i$$

where ci are constants called the coefficients of the polynomial. The algorithm 2.2 below evaluate the value of the polynomial f(x).

Algorithm: PolyEvaluate(C,n,x)
Input: An array C stores n real numbers that are the coefficients of the polynomial. A real value x. Output: The value of f(x).

```
S ← 0
For i ← 0 to n-1 do
{
    P ← 1
    If (i <> 0)
        For (k ← 1 to i) do
            P ← p*x
    S ← s+C[i]*p
}
Return s;
```

Student should create a class that implements the algorithm 2.2 above, with the following functions (please refer to class ***PolyEvaluate*** in the tutorial source code project):

- Ask user to input n, array C (contains real values) and a real number x using the keyboard.
- Show the calculated value f(x).

What is Big-Oh of the algorithm 2.2?

3. Example 03 – Find the power value

The algorithm 2.3 below evaluates the value of

$$s = x^n$$

```
Algorithm: Pow(x,n)
Input: A real number x and an integer n.
Output: The value of s=x^n

S ← 1
If (n = 0) then
    Return s
For i ← 1 to n do
    S ← s*x
Return s;
```

Student should create a class that implements the algorithm 2.3 above, with the following functions (please refer to class ***SimplePow*** in the tutorial source code project):

- Ask user to input a real number x, and an integers n using the keyboard.
- Show the calculated value $s=x^n$.

What is Big-Oh of the algorithm 2.3?

4. Example 04 – Find the greatest common divisor (gcd) of two integers

The greatest common divisor (gcd) of two integers is the largest integer that divides both. For example the ***gcd(50,15)=5***. The algorithm 2.4 computes ***gcd(M,N)***, assuming M≥N.

```
Algorithm: SimpleGCD(m,n)
Input: Two integers m and n
Output: The value of gcd(m,n)

i ← n
While (i>1) do
    If (m is divisible by i) AND (n is divisible by i)
        Return i
    i--
Return 1
```

Student should create a class that implements the algorithm 2.4 above, with the following functions (please refer to class ***SimpleGCD*** in the tutorial source code project):

- Ask user to input two integers m and n using the keyboard.
- Show the gcd(m,n).

What is Big-Oh of the algorithm 2.4?

Exercises

1. Exercise 1

Improve the algorithm 2.1 to get more time efficient algorithm? What is the Big-Oh of the new algorithm?

2. Exercise 2

Improve the algorithm 2.2 to get more time efficient algorithm? What is the Big-Oh of the new algorithm?

3. Exercise 3

Improve the algorithm 2.3 to get more time efficient algorithm? What is the Big-Oh of the new algorithm?

4. Exercise 4

Improve the algorithm 2.4 to get more time efficient algorithm? What is the Big-Oh of the new algorithm?