CSP2348/5243 Data Structures

Tutorial 02: Fundamentals of Algorithm Analysis

Related Objectives from Unit Outline:

- describe the general principles of algorithm complexity and performance

Objectives:

1. To observe the time efficiency of algorithms with different growth rates by testing a few examples;
2. To determine algorithms' time complexity using the big-O notation;

Tasks:
Complete the following.

Task 1: Create a spreadsheet to show the growth rates given in the following table. Observe the differences among these growth rates.

n	1	$\log(n)$	n	$n \times \log(n)$	n^2	n^3	n^{10}	2^n
1								
2								
4								
8								
10								
20								
40								
80								
100								
200								
400								
800								
1000								

Task 2: Suppose that the following expressions are the sum of characteristic operations of some algorithms. Determine the time complexity of each of these expressions by means of the big-*O* notation.

$n^{10} + 9 \times n^9 + 20 \times n^8$

$(n+1)^4$

$(n^2 + n)^2$

$n + 0.001 \times n^3$

$n^3 - 1000 \times n^2$

$n + \log_2(n)$

$n^2 + n \times \log_2(n)$

$2^n + n^2$

$(n^3 + 2 \times n)/(n+5)$

Task 3: Analyse the time complexity of the following methods/algorithms

(Source: exercise 2.4 on page 31 of reference textbook, Java Collections (2001))

The following Java methods implement matrix addition and multiplication. Each matrix is represented by an $n \times n$ two-dimensional array of float numbers.

```
  static void matrixAdd (int n, float[][] a,
        float [][] b, float [][] sum)  {
  // Set sum to the sum of the n×n matrices  a and b.
    for (int i = 0; i < n; i++)  {
      for (int j = 0; j < n; j++)  {
        sum[i][j] = a [i][j] + b[i][j];
      }
    }
  }
  static void matrixMult (int n, float[][] a,
        float [][] b, float [][] prod)  {
  // Set prod to the sum of the n×n matrices a and b.
    for (int i = 0; i < n; i++)  {
      for (int j = 0; j < n; j++)  {
        float s = 0.0;
        for (int k = 0; k < n; k++)  {
           s += a [i][k] * b[k][j];
        }
        prod[i][j] = s;
      }
    }
}
```