**15CSE211 Design Analysis of Algorithms**

**Home Work 1 – Time Complexity Analysis**

**Date: 30.11.2018**

**Instruction: Please maintain a separate Home work (80 pages) Note or you can reuse DS Lab observation book to write the solution of Home work questions.**

**Submission date: 03. 12. 2018**

1. Arrange the following functions in the increasing order of their rates of growth

Hint: You can use Excel)

(sqrt(2))n, 2sqrt(n), n2log n, n(log n)2, (nlog n)2, nlog n, nsqrt(n), nn, (log n)n

1. Let f(n),g(n),f1(n),g1(n) be positive real-valued functions of natural numbers. Prove the following assertions:
   1. If f(n)=O(f1(n)) and g(n)=O(g1(n)),then f(n)+g(n)=O(f1(n)+g1(n))
   2. If f(n)=O(f1(n)) and g(n)=O(g1(n)),then f(n)g(n)= O(f1(n)g1(n)).
   3. f(n) + g(n) = O(max(f(n),g(n))).
2. Determine the time complexity of the following iterative function:

int f ( int A[SIZE][SIZE] , int n )

{

int i, j, sum = 0;

for (i=0; i<n; ++i) {

if (i % 2 == 0)

for (j=0; j<=i; j=j+1) sum = sum + A[i][j];

else

for (j=n-1; j>=i; j=j-1) sum = sum - A[i][j];

}

}

1. Write a function that accepts a positive integer n and prints all permutations of 1,2,3,...,n. Assume that printing a single integer is a basic operation and establish the time complexity of your function.
2. Suppose we want to compute the transpose of a matrix A and store the result in A itself. We do not assume A to be necessarily a square matrix.
   1. Write a function that takes an m x n matrix A as input, computes in a local matrix B the transpose of A and finally copies B back to A. What is the space complexity of this function?
   2. Write a function that computes At in A using only a constant amount of additional storage.
3. Write a function that takes a square matrix A as input and computes in A itself the matrix A - At using only O(1) additional storage. (Hint: The matrix A - At is anti-symmetric, i.e., its (j,i)-th element is the negative of its (i,j)-th element.)
4. Find the time complexity of the give loops
   * + 1. int f2(int n, int a[]) {

s = 0;   
 for (int i = 0; i < n; ++i) {

if (a[i]>n)

for (int j = 0; j < i; ++j) {

s = s + a[i]\*a[j];

}

}

return s;

}

* + - 1. int f3(int n) {

s = 0;

for (int i = 0; i < n; ++i) {

for (int j = 0; j < i\*i; ++j) {

s = s + j;

}

}

return s;

}

1. For each function, write another function that computes the same value and has a better asymptotic running time.
   1. Following function runs in time O(n): Assume that array index starts with value 0

int g1(int n, int a[]) {

s = 0;

for (int i = 1; i < n; ++i)

s = s + a[i] - a[i-1];

return s;

}

Write an equivalent function that runs in time O(1).

* 1. The following function runs in time O(n2):   
      int g2(int n, int a[]) {

s = 0;

for (int i = 0; i < n; ++i) {

for (int j = 0; j < n; ++j) {

s = s + a[i]\*a[j];

} //End of j-loop

} //End of i-loop

return s;   
 }   
 Write an equivalent function that runs in time O(n).

1. For each of the statements below indicate whether the given statement is True or False. Justify your answer.

(a) If f(n) = O(g(n)) then g(n) = Ω(f(n)).

(b) If f(n) = O(g(n)) then f(n) = o(f(n)).

(c) If f(n) = O(g(n)) and g(n) = O(h(n)) then f(n) = O(h(n)).

(d) If f(n) = O(g(n)) then for every n, f(n) ≤ g(n).

(e) If f (n) = O(g(n)) then f (n) + g(n) = O(g(n)).

1. Compare the following functions in terms of order of magnitude. In each case say whether f(n) = O(g(n)), f(n) = Ω(g(n)) or both.

|  |  |
| --- | --- |
| f(n) | g(n) |
| 2n | 2n+2 |
| 5n2 | n log n |
| log n | n0.1 |
| 1.01n | n3 log n |

1. Find the asymptotic time complexities of the following functions.

1. *f*(*n*) = 3*n2*+ 4*n*3/2

2. *f*(*n*) = (*n*-5)2

1. Express the time taken by the following codes as a polynomial of n. Calculate the time for both best case and worst case scenarios. Also specify the time complexity in asymptotic notation.

for (z = 0 to n-1){

for (y = 1 to z) {

for (x = y to n) {

1 operation;

}

}

1. a and b are integers, n is the size of the input, and basic\_op is a basic operation

Mystery(a,b) {

if (a == b) {

for (x = 1 to n+n)

basic\_op;

for (x = 1 to n\*n)

basic\_op;

}

else {

for (x = 1 to 10)

for (y = 1 to x)

basic\_op;

}

}