DATE | | PAGE

Performance analysis:

- Space Complexity of an algorithm is the amount of memory it needs to run to completion.

- Time Complexity of an algorithm is the amount of computer time it needs to run to completion.

Performance Evaluation

① Priori Estimates
② Posterior testing.

Total Computation time
= Execution time for instruction * frequency count.

Now Execution time for instruction varies from machine to machine. It depends on
① machine on which we execute our program,
② Instruction set of the machine language
③ time required to execute an instruction
④ Compiler used for translating the high level language to m/c level language.

Frequency Count: is the no of times an instruction is executed in the execution of the program.

Performance analysis:

- Space Complexity of an algorithm is the amount of memory it needs to run to completion.

- Time Complexity of an algorithm is the amount of computer time it needs to run to completion.

Performance Evaluation

1. Priori Estimates
2. Posterior testing.

Total Computation time
= Execution time for instruction & frequency count.

Now Execution time for instruction varies from machine to machine. It depends on
1. machine on which we execute our program.
2. Instruction set of the machine language
3. time required to execute an instruction
4. Compiler used for translating the high level language to m/c level language.

Frequency Count: is the no of times an instruction is executed in the execution of the program.

Space Complexity:

① Fixed part: that is independent of the characteristics of the inputs and outputs.
- Space for instruction, simple variables, Space for constants.

② Variable part:
Reference variables
Recursion stack.

```
Algorithm Sum (a,n)
{   s = 0.0.
    for i=1 to n do
        s = s + a[i];

    return s;
}.
```

$S_{sum}(n) \geq (n+3)$
: n for a and one each for n, i and s.

```
Algorithm Rsum (a,n)
{ if (n≤0) then return 0.0.
  else return Rsum(a, n-1) + a[n];
}.
```

Return address requires one word of memory
Each call requires three words
- one for n
- one for return address
- one for pointer to a[]

Since depth of the recursion is $n+1$
Time Complexity will be $> 3(n+1)$.

Time Complexity:

Frequency Count: is the number of times an instruction is executed in the execution of the program.

Eg: Algorithm Add (a, b, c, m, n)

Algorithm	frequency
{	
for i = 1 to m do	$m+1$
for j = 1 to n do	$m(n+1)$
c[i][j] = a[i][j] + b[i][j];	mn
}	
	$2mn+2m+1$

Asymptotic Notation.

Big O, Θ, Ω : refer Sahani

① $3n+2 = O(n)$

$3n+2 \leq 4n$ for all $n \geq 2$

$\therefore 3n+2 = O(n)$.

② Prove $3n+2 = \Theta(n)$

Now $3n+2 \geq 3n$ for all $n \geq 2$

and $3n+2 \leq 4n$ for all $n \geq 2$

ie $C_1 = 3$, $C_2 = 4$

$3n \leq 3n+2 \leq 4n$ for all $n \geq 2$

$\therefore 3n+2 = \Theta(n)$

(3) Prove $3n+2 = \Omega(n)$

Now $3n+2 \geq 3n$ for all $n \geq 2$

$\therefore 3n+2 = \Omega(n)$

(4) Prove $10n^2+4n+2 = O(n^2)$

as $10n^2+4n+2 \leq 11n^2$ for all $n \geq 5$

$10n^2+4n+2 = O(n^2)$.

(5) Prove $100n+6 = \Omega(n)$.

as $100n+6 > 100n$

$100n+6 = \Omega(n)$

(6) Prove $6 * 2^n + n^2 = \Omega(2^n)$.

as $6*2^n + n^2 > 2^n$

$6*2^n+n^2 = \Omega(2^n)$.

$O(1) < O(\log n) < O(n) < O(n^2) < O(n^3) \ldots < O(2^n)$.

(7) Show that $5n^2 - 6n = \Theta(n^2)$.

Now. $5n^2 - 6n > 4n^2$ $\quad n \geq 6$.

and $5n^2 - 6n < 6n^2$ $\quad n \geq 2$.

$\therefore 4n^2 \leq 5n^2 - 6n \leq 6n^2$ for all $n \geq 6$.

$\therefore 5n^2 - 6n = \Theta(n)$.

Check whether following algorithm is correct or not.

```
Algorithm Exp(x, n)
{
    m = n ; power = 1 ; z = x;
    while (m > 0) do
    {
        while ((m mod 2) == 0) do
        { m = m/2 ; z = z^2; }
        m = m-1; power = power * z;
    }
    return power;
}
```