**Algorithm Analysis**

Algorithm: is a clearly specified set of instructions to be followed to solve a certain problem.

Analysis: to determine how much computing resources the algorithm will require solving this problem, Resources: CPU and Memory.

CPU 🡪 Time

Memory🡪Space

As the algorithm needs less resources as it is better.

**Mathematical Definition:**

1. **Big O’** : T(n) = O(f(n)), if there are +ve c and n0 , such that T(n) <= c f(n), when n>= n0

N2 = O(N3)

O(f(n)) is the upper bound of the function

1. **Omega** : T(n) = $Ω$(g(n)) , if there are +ve c and n0 , such that T(n) >= c f(n), when n>= n0

N2 = $Ω$ (N)

$Ω$ (g(n)) is the lower bound of the function

1. **Theta** : T(n) = $θ$(h(n)), if there are +ve c1, c2 and n0 ,

such that c1 h(n) >= T(n) >= c2 h(n), when n>= n0

n2 = O(n3) = O(n5)

n2 = $Ω(n)$ = $Ω$(n0.5)

Rules: 1- if the algorithm is performed in ***n*** of steps, each step has a different time complexity T1(N)+ T2(N)+…+Tn(n)

Is the maximum, T(n) = max(T1(N), T2(N),…,Tn(n))

2- T1(n) \*T2(n) = O(f(n)\*g(n))

Ex. Assume we have the following code segment.

Line-by-line analysis

Int I,j,k,n; c1

N=1000000; c2

For(i=0; i<n; ++i) n

 For(j=0; j<n; ++j) n2

 For(k=0; k<n; ++k) n3

 Print(welcome); n3

Time = T(n) = c1+c2+n+n2+n3+n3 = O(n3)

Suppose CPU speed = 1GHz

Time = $\frac{10^{18}}{10^{9}}$ = 109 seconds = 32 years

Ex.

For(i=0; i<n; ++i)

O(n)

Ex.

For(i=0; i<n; ++i)

 For(j=0; j<n; ++j)

O(n2)

Ex.

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

}

For(j=0; j<n; ++j){ }

O(n)

Ex.

For(i=0; i<n; i+=2){

}

N/2 = O(n)

Ex.

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

For(j=0; j<=i; ++j){….; }

}

I j

0 0

1 1

2 1,2

3 1,2,3

…..

n 1,2,3,…..n

1+2+3+….+n = $\frac{n(n+1)}{2}$ = 0.5n+0.5n2=O(n2)

Ex.

Int p=0;

For(i=1; p<=n; ++i){

p=p+i;

}

The loop will continue running until p> n.

I p

1 1

2 1+2

3 1+2+3

…….

K 1+2+3+…+k

P = $\frac{k(k+1)}{2}$ > n

K2 > n

k>$\sqrt{n}$

O($\sqrt{n}$)

Ex.

For(i=1; i<n; i\*=2){….;}

I =1 , 2, 4, 8, ….. = 20, 21,22,23, …, 2k =

Will stop when 2k >= n

K = log n

O(log n)

Ex.

For(i=0; i\*i<n; ++i) {….;}

O($\sqrt{n}$)

Ex.

For(i=1; i<n; i\*=2){p++;} 🡺 log (n) i=1,2,4,8,…, 2K >= n, p=log (n)

For(j=1; j<p; j\*=2){….;} 🡺 log(log (n))

Ex.

For(i=0; i<n; ++i){ 🡺n

 For(j=n; j>1; j=j/2){ 🡺 log (n)

S1;

S2;

S3;

…

S100;

}

}

J = N =512, 256, 128, 64, 32, 16, 8,4,2,1

O(n log(n))

Ex.

A=1, B=1;

While(B<n){

 B=B+A;

 ++A;

}

A B

1 1

1 2

2 1+1+2

3 1+1+2+3

…..

K 1+1+2+3+4+5+ …+ k

B = K(k+1) / 2 >= n

O(sqrt(n))

Ex.

For(i=0; i<n; ++i){ 🡺 n

For(j=0; j<=i\*i; ++j){….; } 🡺 n2

}

I j

0 0

1 1

2 4

N n2

O(n3)

Ex.

For(i=0; i<n; ++i){ 🡺 n

For(j=i; j<=i\*i; ++j){….; } 🡺 n2 – n

}

= n \* (n2-n) = O(n3)