**How to Analyze an Algorithm**

|  |
| --- |
| I is initially 1 |
| 1x2 =2 |
| 2x2= 22 |
| 22 x 2=23 |
| :::::::::::::::::::  How much times??? |
| 2K |

**Analysis Piece of Code:**

For ( I= 1; I < n; I = I x 2)

{

Stmt;

}

Blindly, we cannot say that the loop is repeating n times as each time I is multiply by 2. Assume that the loop is executing 2K times, loop condition will terminate when I > = n since I = 2K

2K >= n => log2 2K >= log2n 🡺 K= log2n 🡺 f(n)= Θ(log2n)

Loop body statements will be executed Θ(log2n) times.

**Analysis Piece of Code:**

|  |  |
| --- | --- |
| For ( I= 1; I < n; I = I x 2  {  Smt;  }  I =2x2x2……….=n  2 is being multiplied until i.>=n  How much times 2 is being multiplied, let it be up to 2k times  2k= n 🡺 K=log2n | For ( I= 0; I < n; I ++)  {  Stmt;  }  I=1+1+1……………..= n  1 is added up until I >=n. 1 is added up to k times.  K=n |

|  |  |
| --- | --- |
| n=8 | n=10 |
| I | I |
| 1 | 1 |
| 2 | 2 |
| 4 | 4 |
| 8\* condition false | 8 |
|  | 16\* condition false |

For ( I= 1; I < n; I = I x 2)

{

Stmt;

}

log2 n = log2 8= log2 23 = 3 times loop body will

be executed.

log2 n = log2 10= 3.2= 4 timesIn this case, take ceil value of log2 n i.e log2 n

|  |
| --- |
| I =n initial value |
| n/2 |
| n/22 |
| n/23 |
| --------  How many times loop will execute |
| Let it be **n/2K** |

**Analysis Piece of Code:**

For ( I= n; I .>= 1; I = I /2)

{

Stmt;

}

**Loop will stop when I < 1**

**Since I = n/2K**

**n/2K < 1 🡺 n/2K = 1**

**n = 2K 🡺 K= log2n or** f(n)= Θ(log2n)

**Analysis Piece of Code:**

For ( I= 0; I x I < n; I ++)

{

Stmt;

}

Loop will be terminated as I x I > n 🡺 I2 > n 🡺 I2 = n

I= Γn f(n)= Θ (Γn)

Loop will be executed Γn times.

**Analysis Piece of Code:**

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

{

Stmt; \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ n times

}

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

{

Stmt; \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ n times

}

F(n) = 2n 🡺 F(n) = Θ(n)

**Analysis Piece of Code:**

**P=0**

For ( I= 1; I < n; I = I x 2)

{

P++; P evaluated here \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_log2 n times

}

**Analysis Piece of Code:**

For ( J= 1; J < P; J = J x 2) P used here

{

Stmt; \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ log2 P times

}

What is P? Since P is incrementing in each iteration, so P will take the value that number of times the loop is repeating i.e P = log2 n

F(n) = Θ(log2 P) = **Θ(log2 log2 n)**

**Analysis Piece of Code:**

For ( I= 0; I < n; I ++) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ n+1 times

{

For ( J= 1; J < n; J = J x 2) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ n x Log2 n

{

Stmt; \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ n x (Log2 n-1)

}

}

F(n)= 2 n Log2 n + n

F(n)= **Θ(** n Log2 n)

|  |  |
| --- | --- |
| For ( i= 0; i < n; i ++) | **Θ(** n) |
| For ( i= 0; i < n; i =i+2) | n/2 or **Θ(** n) |
| For ( i= n; i > 1; i-- ) | **Θ(** n) |
| For ( i= n; i > 1; i=i-2 ) | **n/2 or Θ(** n) |
| For ( i= 1; i < n; i =i\*2) | **Θ(** Log2 n) |
| For ( i= 1; i < n; i =i\*3) | **Θ(** Log3 n) |
| For ( i= n; i > 1; i =i/2) | **Θ(** Log2 n) |