**Introduction to Algorithms- Day 02**

**Time Complexity Examples: -**

1. **For(i=1;i<n; i=i\*2)**

**{**

**Stmt;**

**}**

**In this case,**

**Initally i=1, then its multiplied by 2 for each iteration until i>=n//**

**So,**

**i= 1\*2\*2……=n**

**2^k=n**

**Time complexity=K=log 2n= [logn]//**

**Its denoted as ceil value to avoid the fraction//**

1. **For(i=1;i<=n;i++)**

**{**

**Stmt;**

**}**

**For this case,**

**Initally, i=1 and then it goes on incrementing by 1 until i>=n**

**So,**

**i= 1+1+1+1+……. +1=n**

**time complexity=k=n//**

1. **For( i=1;i>=1;i=i/2)**

**{**

**Stmt;**

**}**

**In this case,**

**It starts from i/n.. n/2..n/2^2…n/2^3….. n/2^k**

**It executes until,**

**Assume i<1,**

**Therefore, n/2^k<1**

**n/2^k=1**

**n= 2^k**

**k= log2n//**

**time complexity= O(logn)//**

1. **For(i=0;i\*i<n;i++)**

**{**

**Stmt;**

**}**

**In this case,**

**I\*i<n**

**It will execute until,**

**I\*i>=n**

**By Equating we get,**

**I^2=n**

**I=//**

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

**{**

**Stmt; -------------------------- n**

**}**

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

**{**

**Stmt; -------------------------n**

**}**

**Total Time= 2n//**

**For independent loops, the loops go till n only and hence the time complexity is O(n)//**

1. **P=0**

**For(i=1;i<n;i=i\*2)**

**{**

**P++;-------------------------------- p=log n**

**}**

**For(j=1;j<p;j=j\*2)**

**{**

**Stmt; -------------------------------- log p**

**}**

**Time complexity= O (log logn)//**

**Note:- Since we have i=i\*2 it directly evaluates to log n as seen previously//**

1. **For(i=0;i<n:i++)--------------------------- n**

**{**

**For(j=1;j<n;j=j\*2)-------------------------n \* log n**

**{**

**Stmt;--------------------------------------n\* log n**

**}**

**}**

**Total = 2nlogn +n**

**Time Complexity= O(n log n)//**

**Time complexity of majority of examples until now:-**

1. **For(i=0;i<n;i++)-------------------------- O(n)//**
2. **For(i=0;i<n;i=i+2) ----------------------- n/2= O(n)//**
3. **For(i=n;i>1;i--) ----------------------------- O(n)//**
4. **For(i=1;i<n;i=i\*2) ------------------------- O(log2n)//**
5. **For(i=1;i<n;i=i\*3) -------------------------- O(log3n)//**
6. **For(i=n;i>1;i=i/2) -------------------------- O(log2n)//**

**Time Complexity Analysis of If and While Loops**

**1)**

**i=0--------------------- 1**

**while(i<n)------------ N+1**

**{**

**Stmt; --------------- n**

**I++; ----------------- n**

**}**

**Total time= 3n+2**

**F(n) =3n+2**

**= O(n)//**

**2)**

**A=1;**

**While(a<b)**

**{**

**Stmt;**

**a=a\*2;**

**}**

**Analysis:-**

**A**

**1**

**1\*2=2**

**2\*2=2^2**

**2^2\*2=2^3**

**:**

**:**

**:**

**2^k**

**Now, the loop will terminate when a>=b**

**a>=b**

**therefore, a=2^k**

**2^k >=b**

**2^k=b**

**K=log2b**

**Time complexity= O (log n)//**

**3)**

**I=n**

**While(i>1)**

**{**

**Stmt;**

**I=i/2;**

**}**

**Time complexity= O (log n)//**

**As mentioned, wherever we have i=i/2 or i=i\*2 we can directly interpret it as O (log n)//**

**4)**

**i=1**

**k=1**

**while(k<n)**

**{**

**Stmt;**

**K=k+i;**

**I++;**

**}**

**For this case, lets analyze,**

|  |  |
| --- | --- |
| **i** | **k** |
| **1** | **1** |
| **2** | **1=1=2** |
| **3** | **2+2=4** |
| **4** | **2+2+3=7** |
| **5** | **2+2+3+4=11** |
| **..** | **..** |
| **m** | **m(m+1)/2 (sum of first m natural numbers)//** |
|  |  |

**Lets assume it will execute until k>=n,**

**m(m+1)/2>=n**

**m^2>=n**

**m= //**

**5) GCD of 2 Numbers m and n analysis:-**

**While (m! =n)**

**{**

**If(m>n)**

**{**

**m=m-n;**

**}**

**Else**

**{**

**N=n-m;**

**}**

**Lets analyze the algorithm,**

**For the above example:-**

**It will execute until m=n that means m!=n,**

**Case I:-**

**When m=4 n=2**

**Iteration 01: m=2 and n=2 (executes only once)**

**When m=2 n=2**

**Both are equal only so it will execute 0 times//**

**When m=16 and n=2m it will execute 7 times ,i.e almost (n/2) times//**

**Time complexity= O(n/2)= O(n) (degree of the polynomial)//**

**Minimum time complexity= O (1)//**

**6)**

**Random example with if:-**

**Algorithm Test(n)**

**{**

**If(n<5)**

**{**

**Printf(“%d”,n) ------------------- 1 --- O(1)---(if its True) (Best Case)**

**}**

**Else**

**{**

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

**{**

**Printf(“%d”,i) -----------------------n--- O(n)--- (if its False) (Worst Case)**

**}**

**}**

**}**

**If a loop is using conditional statement, the time complexity varies depending upon the condition that is being utilized for the execution//**

**Its not a rule that if the conditional statement is there then best case will always be O(1) like in the example below:-**

**Algorithm Test(n)**

**{**

**If(n<5)**

**{**

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

**{**

**Printf(“%d”,i) ---------------------- n = O(n)= (Best Case)//**

**}**

**}**

**}**

**For conditional statements within loops the outputs will vary depending upon the condition//**

**Types (Classes) of Time Functions: -**

1. **O(1)----------------------- Constant**

**Example:-**

**F(n)=2, f(n)=5, f(n)=5000= O(1)//**

1. **O(log n)------------------- Logarithmic**
2. **O(n)------------------------ Linear**

**Example: -**

**F(n)=2n+3, f(n)=500n+700, f(n)=n/5000 + 6= O(n)//**

1. **O(n^2)-------------------- Quadratic**
2. **O(n^3)--------------------- Cubic**
3. **O(2^n), O(3^n), O(n^n) -------------------- Exponential**

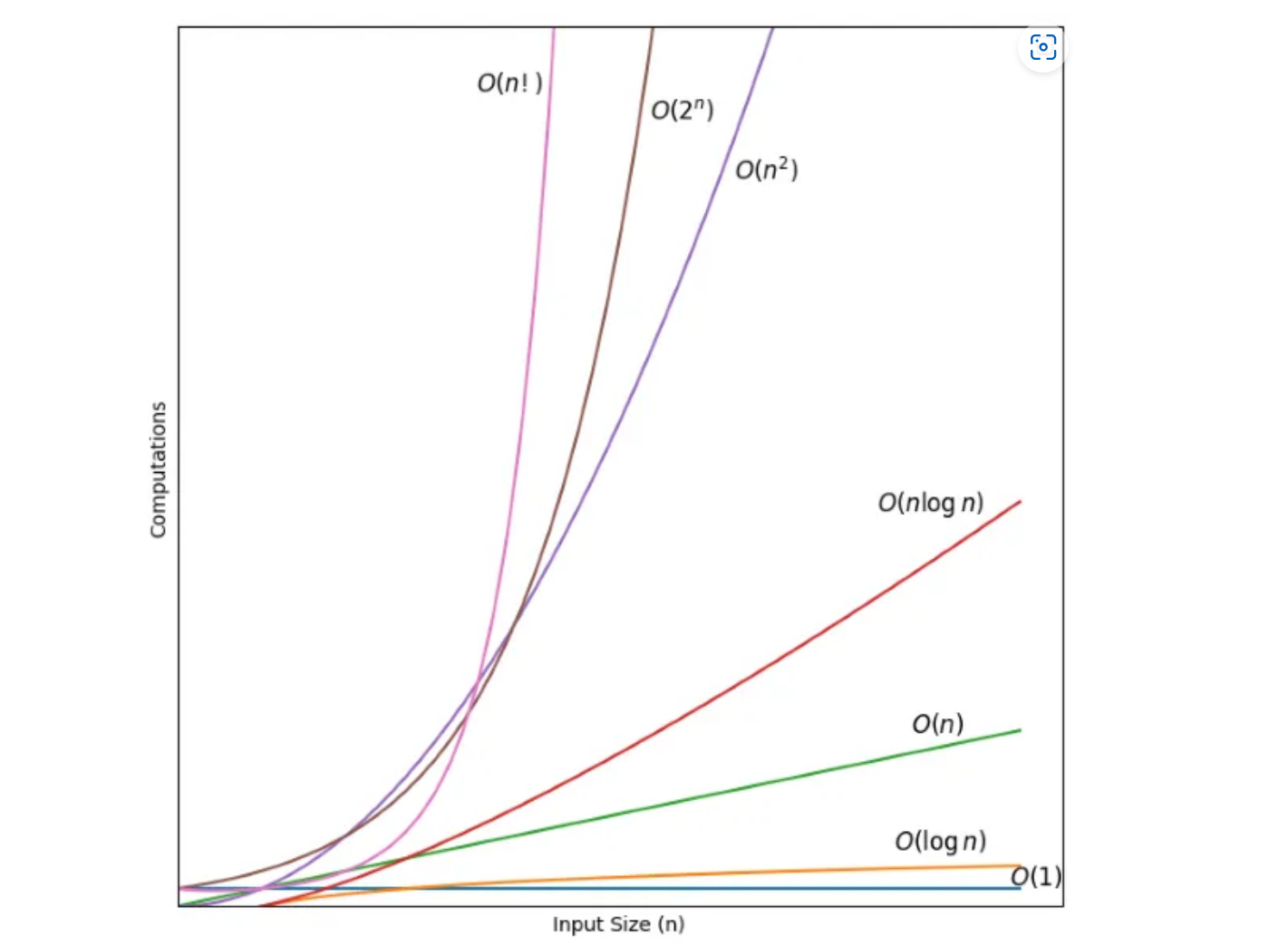
**---------------------------------------------- Compare Class of Time Functions -----------------------------------------**

**1<log n<<n<n logn<n^2<n^3<…………………. <2^n<3^n<……………. <n^n//**

**Example:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **log n** | **N** | **N^2** | **2^n** |
| 0 | 1 | 1 | 2 |
| **1** | **2** | **4** | **4** |
| **2** | **4** | **16** | **16** |
| **3** | **8** | **64** | **256** |
| **3.1** | **9** | **81** | **512** |
|  | **15** | **225** | **32,768** |

**So, you can see how the growth is 2^n grows very fast hence exponential then n^2 hence quadratic then N hence linear and then log n and hence logarithmic//**

****

**Fig:- Comparison of various time Functions/Classes**

**We did not mention about n! it has the highest growth,**

**O(n!) — Factorial Time**

**Time complexity= O(n!)//**