Todays content

1. How to compare two algorithms
2. Calculate iterations
3. Comparing 2 Algo's

How to compare two Algorithms

If Given $N = 10^4$ elements sort them in increasing order

ar[5] = {3 2 6 8 1} → {1 2 3 6 8}

Erection:

Algo1: AbhiSort

15sec {Windows XP}

↓ → Macbook

7sec {C++}

↓

7sec {Volcano hot}

↓ → Himalayas

Sec

Algo2: MaheshSort

10sec {Macbook My}

↓

10sec {Python}

↓ → C++

5sec {Himalayas cool}

↓

Sec

Issues with Execution Time: External factor will effect it.

Hence not a good factor to use to compare 2 Algo's.

Good Factor: Iterating → Independent of external factors.

```
void func(int N){
    int N=ar.length;
    int s=0;
    for(int i=0; i<N; i++){ // i = 0..N-1, iterating = N.
        s=s+ar[i]
    }
    print(s);
}
```

Con: To compare 2 algo we need iterating of program

# Basics of Logaritum.

Log : Logaritum is the inverse of exponential functional

$\log_b^a$ : To what value we need to raise b to get a.

## log Basics:

$\log_b^a = x \;//\; b^x = a$.

$\log_2^{64} = 6 \;//\; 2^6 = 64$

$\log_5^{25} = 2 \;//\; 5^2 = 25$

$\log_3^{27} = 3 \;//\; 3^3 = 27$

$\log_2^{32} = 5 \;//\; 2^5 = 32$

$\log_2^{10}$ → $2^3 = 8$, $2^4 = 16$ } $2^{3.x} = any\ :3$

$\log_2^{40}$ → $2^5 = 32$, $2^6 = 64$ } $2^{5.x} = any\ :5$

## Few Formula:

1. $\log_a^{a^N} = N$

$\log_5^{5^3} = 3$ $\log_2^{2^{10}} = 10$ $\log_7^{7^3} = 3$

2. $N = 2^k \Rightarrow k = \log_2^N$

Apply $\log_2$ m both sides

$\log_2^N = \log_2^{2^k}$

$\log_2^N = k$

# Few basic maths

1. No: of elements from [a..b] both included : $b-a+1$.

   Eg: [3 7] = 3 4 5 6 7 = $7-3+1=5$

   [6 10] = $10-6+1=5$.

   Note: [ corner considered ( corner not considered.

2. No: of elements from [a..b) both included : $b-a$

3. Sum of $1^{st}$ N Natural Numbers :

$$S_N = \boxed{1} + \boxed{2} + \boxed{3} + \dots + \boxed{N-2} + \boxed{N-1} + \boxed{N}$$

$$S_N = \boxed{N} + \boxed{N-1} + \boxed{N-2} + \dots + \boxed{3} + \boxed{2} + \boxed{1}$$

$$2S_N = N+1 + N+1 + N+1 + \dots + N+1 + N+1 + N+1$$

$$2S_N = (N)(N+1)$$

$$S_N = \frac{(N)(N+1)}{2}$$

Calculating Iterations: No: of times loop runs

```
void fun(int N){
    int s=1;
    for(int i=1; i<=100; i++){
        s=s+i;
    }
    print(s);
}
```

i : [1..100] a b b-a+1 → 100-1+1 = 100 iterations

```
void fun(int N){
    int s=1;
    for(int i=3; i<=50; i++){
        s=s+i;
    }
    print(s);
}
```

i : [3...50] a b b-a+1 → 50-3+1 = 48

```
void fun(int N){
    int s=1;
    for(int i=1; i<=N; i++){
        print("Hello");
    }
    print("Tyu");
}
```

i : [1..N] a b b-a+1 → N-1+1 = N

```
void fun(int N){
    int s=1;
    for(int i=0; i<N; i++){
        print("Hello");
    }
    print("Tyu");
}
```

i : [0..N-1] a b b-a+1 → N-1-0+1 = N iterations

Q5
```
void fun(int N){
    int i,j;
    for(int i=1; i<=N; i++){ i : [1..N] = N-1+1 = N
        print("Hello");
    }
    for(int i=1; i<=M; i++){ i : [1..M] = M-1+1 = M
        print("Hello");
    }
    print("Ty");
}
```

N+M iterations (b-a+1)

#N>0

Q6
```
void fun(int N){
    int i = N;
    while(i>1){ //Obs1 Code stop: i=1
        i=i/2
    }
}
```

Obs2: $i=N \xrightarrow{1^{st}} i=N/2>1 \xrightarrow{2^{nd}} i=N/4>1 \xrightarrow{3^{rd}} N/8>1 \xrightarrow{4^{th}} N/16 \longrightarrow \xrightarrow{k^{th}} i=N/2^k$

Ans: Let's say after k iteration code should stop

$i=1$ & $i=N/2^k$ { $a=b$ & $a=c$ }

Obs1 Obs2

$1=N/2^k \Rightarrow 2^k=N \Rightarrow k=\log_2 N$

Q7 #N>0
```
void fun(int N){
    for(int i=0; i<N; i=i*2){ i=0<N 1st→ 0<N 2nd→ i=0<N 3rd→ i=0<N →...∞loop
        print("i");
    }
}
```

#N>0

Q8
```
void fw(int N){
    for(int i=1; i<N; i=i*2){ obs1: i==N stop
        print("i");
    }
}
```

obs2:

$i=1<N \xrightarrow{1^{st}} i=2<N \xrightarrow{2^{nd}} i=4<N \xrightarrow{3^{rd}} i=8<N \xrightarrow{4^{th}} i=16<N \rightarrow \rightarrow \xrightarrow{n}$

$i=2^1 \quad i=2^2 \quad i=2^3 \quad i=2^4 \quad i=2^n$

After k iterations code stops

$i=N$ (obs1) ↔ $i=2^k$ (obs2)

$a=b$ & $a=c$

$N=2^k \Rightarrow k=\log_2 N \Rightarrow N=\log_2 k$

Nested loops:

```
void fw(int N){
    for(int i=1; i<=4; i++){
        print("Hello1");
        for(int j=1; j<=i; j++){
            print("Hello2");
        }
    }
}
```

Nested loops

| i | j : [1..i] |
|---|---|
| 1 Hello1 | j : [1..1] : 1 Hello2 |
| 2 Hello1 | j : [1..2] : 2 Hello2 |
| 3 Hello1 | j : [1..3] : 3 Hello2 |
| 4 Hello1 | j : [1..4] : 4 Hello2 |
| 5 stop | |

Outerloop = 4 iter

Innerloop = 10 iter

Total = 14 iter

```
void fw(int N){
    for(int i=1; i<=10; i++){
        print("Hello1");
        for(int j=1; j<=N; j++){
            print("Hello2");
        }
    }
}
```

## Nested loops

| i | j : [1..N] |
|---|---|
| 1 Hello1 | j : [1..N] = N Hello2 |
| 2 Hello1 | j : [1..N] = N Hello2 |
| 3 Hello1 | j : [1..N] = N Hello2 |
| ⋮ | |
| 10 Hello1 | j : [1..N] = N Hello2 |
| 11 : Break | |

Outer loop = 10

Inner loop = 10N

Total = 10 + 10N

```
void fun(int N){ TODO
    for(int i=1; i<=N; i++){
        print("hello1");
        for(int j=1; j<=N; j++){
            print("hello2");
        }
    }
}
```

Nested loops

| i | j : [1..i] |
|---|---|
| | j : [ ] |

Outerloop =

Innerloop =

Total =

```
void fun(int N){
    for(int i=1; i<=N; i++){
        for(int j=1; j<=i; j++){
            print("hi");
        }
    }
}
```

Nested loops

| i | j : [1..i] | |
|---|---|---|
| 1 | j : [1..1] = | 1 ite |
| 2 | j : [1..2] = | 2 ite |
| 3 | j : [1..3] = | 3 ite |
| ⋮ | | |
| N | j : [1..N] = | N ite |
| | | = |

Outerloop = N

Innerloop = $\frac{(N)(N+1)}{2}$

Total = $N + \frac{N*(N+1)}{2}$

```
void fn(int N){ TODO
    int k=1;
    for(int i=1; i<=N; i++){
        print("Hello");
        while(k<=10){
            print("Yellow");
            k++;
        }
    }
}
```

## Nested loops

| i | j : [1..i] |
|---|---|
| | j : [ ] |

Outerloop =

Innerloop =

Total =