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}

Execution:

Algo1: TamSort		Algo2: NewSrt	
15sec	{Windows XP}	10sec	{MacBook}
↓	Moved to MacBook	↓	
7sec	{C++}	10sec	{python}
↓		↓	Moved to C++
7sec	{Volcano Hot}	6sec	{Cool Temp}
↓	Moved to cool temp	↓	
6sec		6sec	

Issues with using Execution to compare 2 Algo

1. Lot of external factors effect execution time
   Software/Hardware/Language/Surroundings

Good Factor: Iterations

```
void funC(int N){
    int N=ar.length;
    int S=0;
    for(int i=0;i<N;i++){ // i=0..N-1, iterations = N. final
        S=S+ar[i]
    }
    print(S);
}
```

Con: To compare 2 algo we need to calculate it's iterations

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:

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

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

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

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

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

$\log_2^{10}$ → $2^4 = 16$, $2^3 = 8$ → $2^{3.14..} = 10$, ans = 3

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

Few Formula:

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

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

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

$N = 2^k$

Apply $\log_2$ n both side

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

$\log_2^N = k$

Few basic maths

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

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

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

   Note: [corner included (corner excluded

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

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

$$S_N = 1 + 2 + 3 + \dots + N-2 + N-1 + N$$
$$S_N = N + N-1 + N-2 + \dots + 3 + 2 + 1$$
$$2S_N = N+1 + N+1 + N+1 \dots\dots N+1 + N+1 + N+1$$

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

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

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);
}
```

a b b-a+1

i : [1...100] 100-1+1 = 100

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

a b b-a+1

i : [3...50] 50-3+1 = 48

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

a b b-a+1

i : [1...N] N-1+1 = N.

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

a b b-a+1

i : [0...N-1] N-0+1 = N.

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

Ans: N>0

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

$i = N > 1$ —After 1→ $N/2 > 1$ —After 2→ $N/4 > 1$ —After 3→ $N/8 > 1$ —After 4→ $N/16$

$i = N/2^2$ $i = N/2^3$ $i = N/2^4$

Ans: After k iterating code stops. // After $\log_2^N$ iterating code stops

$i = 1 = \frac{N}{2^k} \Rightarrow 1 = \frac{N}{2^k} \Rightarrow 2^k = N \Rightarrow k = \log_2^N$

Code stops

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

Q7
```
void fun(int N){ // N>0 : ∞ iterating
    for(int i=0; i<N; i=i*2){
        print("i");
    }
}
```

Q8
```
void fun(int N){
    for(int i=1; i<N; i=i*2){ // i=N once & stops
        print("i");
    }
}
```

$i=1<N \xrightarrow{1} 2<N \xrightarrow{2} 2^2<N \xrightarrow{3} 2^3<N \xrightarrow{4} 2^4<N$

Assume after k iteration stops

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

Nested loops:

```
void fun(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 time Hello2
2 Hello1	j : [1..2]	2 time Hello2
3 Hello1	j : [1..3]	3 time Hello2
4 Hello1	j : [1..4]	4 time Hello2
5	Stop;	

Outerloop = 4 times Hello1

Innerloop = 10 times Hello2

Total = 14 times

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

```
i=1
for(int j=1; j<=1; j++){
    printf("Hello2");
}

i=2
for(int j=1; j<=2; j++){
    printf("Hello2");
}

i=3
for(int j=1; j<=3; j++){
    printf("Hello2");
}

i=4
for(int j=1; j<=4; j++){
    printf("Hello2");
}
```

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

Nested loops

i	j : [1..N]	
1 Hello1	j : [1..N]	N time Hello2
2 Hello1	j : [1..N]	N time Hello2
3 Hello1	j : [1..N]	N time Hello2
⋮		
10 Hello1	j : [1..N]	N time Hello2
11 stop		

Outerloop = 10 times Hello1

Innerloop = 10N times Hello2

Total = 10 + 10N.

Nested loops

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

i	j : [1..i]
1	j : [1...1] : 1 iter
2	j : [1..2] : 2 iter
3	j : [1..3] : 3 iter
⋮	
N	j : [1..N] : N iter
N+1 : stop	

Outer loop = N iterat

Inner loop = $1+2+3+..N = \frac{(N)(N+1)}{2}$

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

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

i	j : [ ] iterations

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

i	j : [ ] iterations

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

i	j	iterations