

Today's content

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

How to compare two Algorithms

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

$ar[5] = \{3, 2, 6, 8, 1\} \rightarrow \{1, 2, 3, 6, 8\}$

Execution: Algo1: Abhisort

15 sec {Windows XP}

↓ → Macbook

7 sec {C++}

↓

7 sec {Volcano hot}

↓ → Himalayas

5 sec

Algo2: Manna Sort

10 sec {Macbook M4}

↓

10 sec {Python}

↓

→ C++

5 sec {Himalayas cool}

↓

5 sec

Issues with Execution Time: External factor will affect it.

Hence not a good factor to use to compare 2 Algos.

Good Factor: Iterations → 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, iterations = N
```

```
        s = s + ar[i]
```

```
    }  
    print(s);
```

```
}
```

Con: To compare 2 algo we need iterative of program

Basics of logarithm.

\log : logarithm is the inverse of exponential functional

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

\log basics:

$$\log_b^a = n \quad // \quad b^n = a.$$

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

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

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

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

$$\log_2^{10} \left\{ \begin{array}{l} 2^3 = 8 \\ 2^4 = 16 \end{array} \right\} \quad 2^{3.7} = \text{Ans: } 3$$

$$\log_2^{40} \left\{ \begin{array}{l} 2^5 = 32 \\ 2^6 = 64 \end{array} \right\} \quad 2^{5.7} = \text{Ans: } 5$$

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

Apply \log_2 on 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 = $b-a+1$.

Ex: $[3..7] = 3\ 4\ 5\ 6\ 7 = 7-3+1 = 5$

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

Note: $[$ corner considered $)$ corner not considered.

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

3. Sum of 1st N Natural Numbers =

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

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

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

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

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

Calculating iterations: No. of times loop runs

```
void fun(int n) {
```

```
    int s=1;    101
```

```
    for(int i=1; i<=100; i++) {    a    b    b-a+1  
        i: [1..100]    100-1+1 = 100 iterations
```

```
        s=s+i;
```

```
    }  
    print(s);
```

```
}
```

```
void fun(int n) {
```

```
    int s=1;
```

```
    for(int i=3; i<=50; i++) {    a    b    b-a+1  
        i: [3...50]    50-3+1 = 48
```

```
        s=s+i;
```

```
    }  
    print(s);
```

```
}
```

```
void fun(int n) {
```

```
    int s=1;
```

```
    for(int i=1; i<=N; i++) {    a    b    b-a+1  
        i: [1..N]    N-1+1 = N
```

```
        print("Hello");
```

```
    }  
    print("Ty");
```

```
}
```

```
void fun(int n) {
```

```
    int s=1;
```

```
    for(int i=0; i<N; i++) {    a    b    b-a+1  
        i: [0..N-1]    N-1-0+1 = N iterations
```

```
        print("Hello");
```

```
    }  
    print("Ty");
```

```
}
```

Q5 void fun(int n) {

int sz1;

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: $i = N \xrightarrow{1^{st}} i = N/2 > 1 \xrightarrow{2^{nd}} i = N/4 > 1 \xrightarrow{3^{rd}} N/8 > 1 \xrightarrow{4^{th}} N/16 \rightarrow \dots \rightarrow \frac{N}{2^k}$
 $i = N/2^1 \quad i = N/2^2 \quad i = N/2^3 \quad i = N/2^4 \quad i = N/2^k$

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

$i = 1$ & $i = N/2^k$ { a = b & a = c }
 obs1 \uparrow obs2 \uparrow

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

Q7 #N>0

void fun(int N) {

for(int i=0; i<N; i=i*2) { i = 0 < N $\xrightarrow{1^{st}}$ 0 < N $\xrightarrow{2^{nd}}$ i = 0 < N $\xrightarrow{3^{rd}}$ i = 0 < N $\rightarrow \dots \rightarrow \infty$ loop
 } print("i");

}

}

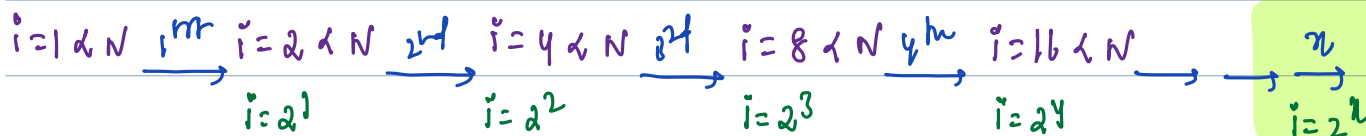
N > 0

```

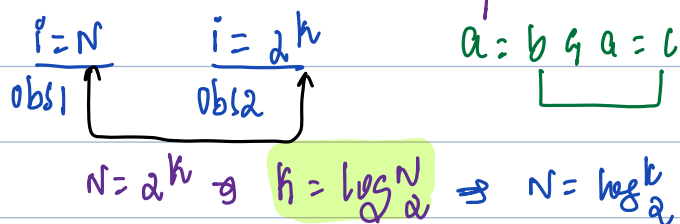
Q8 void fun(int N) {
    for (int i = 1; i <= N; i = i * 2) {
        print("i");
    }
}

```

obs 2:



After k iterations code stops



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 Hello2
2 Hello1	j: [1..2] : 2 Hello2
3 Hello1	j: [1..3] : 3 Hello2
4 Hello1	j: [1..4] : 4 Hello2
5 Stop	

Outer loop = 4 iter

Inner loop = 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

Nested loops

```
void fun(int N) { TODO
```

```
    for(int i=1; i<=N; i++) {
```

```
        print("Hello");
```

```
        for(int j=1; j<=N; j++) {
```

```
            print("Hello2");
```

```
        }
```

```
    }
```

```
}
```

i	j: [1..i]
	j: []

Outerloop =

Innerloop =

Total =

```
void fun(int N) {
```

```
    for(int i=1; i<=N; i++) {
```

i=N

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

Nested loops

```
void fun(int N) { TODO
```

```
    int d = 1;
```

```
    for (int i = 1; i <= N; i++) {
```

```
        printf("Hello");
```

```
        while (d <= 10) {
```

```
            printf("Yellow");
```

```
            d++;
```

```
    }
```

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

	d: []
--	---------

Outer loop =

Inner loop =

Total =

