

## Today's Content

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

How to compare two Algorithms

IQ 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: Abhishek

15sec {Windows XP}

↓ → MacBook

7sec {C++}

↓

7sec {Volcano trip}

↓ → Himalayas

5sec

Algo2: Meenu

10sec {Macbook M4}

↓

10sec {Python}

↓ → C++

5sec {Himalayas cool}

↓

5sec

Issues with Execution Time: External factor will effect it.

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

Good Factor: Iterations  $\rightarrow$  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 iterations 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:

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

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

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

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

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

$$\log_2^{10} \left. \begin{array}{l} \rightarrow 2^3 = 8 \\ \rightarrow 2^4 = 16 \end{array} \right\} 2^n = \text{ans} : 3$$

$$\log_2^{40} \left. \begin{array}{l} \rightarrow 2^5 = 32 \\ \rightarrow 2^6 = 64 \end{array} \right\} 2^n = \text{ans} : 5$$

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 =  $b-a+1$ .

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

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

Note: [ corner considered ( corner not considered.

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

3. Sum of  $1^{\text{st}}$   $N$  Natural Numbers =

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

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

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

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

Calculating Iterations: No. of times loop runs

void fnc(int n){

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

void fnc(int n){

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

void fnc(int n){

    int s=1;      for                            a      b      b-a+1  
    for(int i=1; i<=n; i++) {      i: [1..N]    N-1+1 = N  
        print("Hello");  
    }  
    print("Type");

void fnc(int n){

    int s=1;      for                            a      b      b-a+1  
    for(int i=0; i<n; i++) {      i: [0..N-1]    N-1-0+1 = N iterations  
        print("Hello");  
    }  
    print("Type");

Q5 void fun(int N){

    int s2l;

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

    } printf("Hello");

} Non-iterative

b-a+1

for(int i=1; i<=M; i++) { i : [1..N] = N-1+1 = N }

    } printf("Hello");

    printf("Ty");

}

#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^n} i=N/2 \geq 1 \xrightarrow{2^n} i=N/2^2 \geq 1 \xrightarrow{3^n} i=N/2^3 \geq 1 \xrightarrow{4^n} i=N/2^4 \geq 1 \xrightarrow{5^n} i=N/2^5 \geq 1 \xrightarrow{6^n} i=N/2^6 \geq 1 \xrightarrow{7^n} i=N/2^7 \geq 1 \xrightarrow{8^n} i=N/2^8 \geq 1 \xrightarrow{9^n} i=N/2^9 \geq 1 \xrightarrow{10^n} i=N/2^{10} \geq 1 \xrightarrow{11^n} i=N/2^{11} \geq 1 \xrightarrow{12^n} i=N/2^{12} \geq 1 \xrightarrow{13^n} i=N/2^{13} \geq 1 \xrightarrow{14^n} i=N/2^{14} \geq 1 \xrightarrow{15^n} i=N/2^{15} \geq 1 \xrightarrow{16^n} i=N/2^{16} \geq 1 \xrightarrow{17^n} i=N/2^{17} \geq 1 \xrightarrow{18^n} i=N/2^{18} \geq 1 \xrightarrow{19^n} i=N/2^{19} \geq 1 \xrightarrow{20^n} i=N/2^{20} \geq 1 \xrightarrow{21^n} i=N/2^{21} \geq 1 \xrightarrow{22^n} i=N/2^{22} \geq 1 \xrightarrow{23^n} i=N/2^{23} \geq 1 \xrightarrow{24^n} i=N/2^{24} \geq 1 \xrightarrow{25^n} i=N/2^{25} \geq 1 \xrightarrow{26^n} i=N/2^{26} \geq 1 \xrightarrow{27^n} i=N/2^{27} \geq 1 \xrightarrow{28^n} i=N/2^{28} \geq 1 \xrightarrow{29^n} i=N/2^{29} \geq 1 \xrightarrow{30^n} i=N/2^{30} \geq 1 \xrightarrow{31^n} i=N/2^{31} \geq 1 \xrightarrow{32^n} i=N/2^{32} \geq 1 \xrightarrow{33^n} i=N/2^{33} \geq 1 \xrightarrow{34^n} i=N/2^{34} \geq 1 \xrightarrow{35^n} i=N/2^{35} \geq 1 \xrightarrow{36^n} i=N/2^{36} \geq 1 \xrightarrow{37^n} i=N/2^{37} \geq 1 \xrightarrow{38^n} i=N/2^{38} \geq 1 \xrightarrow{39^n} i=N/2^{39} \geq 1 \xrightarrow{40^n} i=N/2^{40} \geq 1 \xrightarrow{41^n} i=N/2^{41} \geq 1 \xrightarrow{42^n} i=N/2^{42} \geq 1 \xrightarrow{43^n} i=N/2^{43} \geq 1 \xrightarrow{44^n} i=N/2^{44} \geq 1 \xrightarrow{45^n} i=N/2^{45} \geq 1 \xrightarrow{46^n} i=N/2^{46} \geq 1 \xrightarrow{47^n} i=N/2^{47} \geq 1 \xrightarrow{48^n} i=N/2^{48} \geq 1 \xrightarrow{49^n} i=N/2^{49} \geq 1 \xrightarrow{50^n} i=N/2^{50} \geq 1 \xrightarrow{51^n} i=N/2^{51} \geq 1 \xrightarrow{52^n} i=N/2^{52} \geq 1 \xrightarrow{53^n} i=N/2^{53} \geq 1 \xrightarrow{54^n} i=N/2^{54} \geq 1 \xrightarrow{55^n} i=N/2^{55} \geq 1 \xrightarrow{56^n} i=N/2^{56} \geq 1 \xrightarrow{57^n} i=N/2^{57} \geq 1 \xrightarrow{58^n} i=N/2^{58} \geq 1 \xrightarrow{59^n} i=N/2^{59} \geq 1 \xrightarrow{60^n} i=N/2^{60} \geq 1 \xrightarrow{61^n} i=N/2^{61} \geq 1 \xrightarrow{62^n} i=N/2^{62} \geq 1 \xrightarrow{63^n} i=N/2^{63} \geq 1 \xrightarrow{64^n} i=N/2^{64} \geq 1 \xrightarrow{65^n} i=N/2^{65} \geq 1 \xrightarrow{66^n} i=N/2^{66} \geq 1 \xrightarrow{67^n} i=N/2^{67} \geq 1 \xrightarrow{68^n} i=N/2^{68} \geq 1 \xrightarrow{69^n} i=N/2^{69} \geq 1 \xrightarrow{70^n} i=N/2^{70} \geq 1 \xrightarrow{71^n} i=N/2^{71} \geq 1 \xrightarrow{72^n} i=N/2^{72} \geq 1 \xrightarrow{73^n} i=N/2^{73} \geq 1 \xrightarrow{74^n} i=N/2^{74} \geq 1 \xrightarrow{75^n} i=N/2^{75} \geq 1 \xrightarrow{76^n} i=N/2^{76} \geq 1 \xrightarrow{77^n} i=N/2^{77} \geq 1 \xrightarrow{78^n} i=N/2^{78} \geq 1 \xrightarrow{79^n} i=N/2^{79} \geq 1 \xrightarrow{80^n} i=N/2^{80} \geq 1 \xrightarrow{81^n} i=N/2^{81} \geq 1 \xrightarrow{82^n} i=N/2^{82} \geq 1 \xrightarrow{83^n} i=N/2^{83} \geq 1 \xrightarrow{84^n} i=N/2^{84} \geq 1 \xrightarrow{85^n} i=N/2^{85} \geq 1 \xrightarrow{86^n} i=N/2^{86} \geq 1 \xrightarrow{87^n} i=N/2^{87} \geq 1 \xrightarrow{88^n} i=N/2^{88} \geq 1 \xrightarrow{89^n} i=N/2^{89} \geq 1 \xrightarrow{90^n} i=N/2^{90} \geq 1 \xrightarrow{91^n} i=N/2^{91} \geq 1 \xrightarrow{92^n} i=N/2^{92} \geq 1 \xrightarrow{93^n} i=N/2^{93} \geq 1 \xrightarrow{94^n} i=N/2^{94} \geq 1 \xrightarrow{95^n} i=N/2^{95} \geq 1 \xrightarrow{96^n} i=N/2^{96} \geq 1 \xrightarrow{97^n} i=N/2^{97} \geq 1 \xrightarrow{98^n} i=N/2^{98} \geq 1 \xrightarrow{99^n} i=N/2^{99} \geq 1 \xrightarrow{100^n} i=N/2^{100} \geq 1$

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

$i=1 \xrightarrow{1^n} i=N/2^k \geq 1$  {  $a=b \wedge a=c$  }

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

Q7 #N>0

void fun(int N){

    for(int i=0; i<N; i=i\*2) { i = 0 & N  $\xrightarrow{1^n} 0 \times N \xrightarrow{2^n} i=0 \times N \xrightarrow{3^n} i=0 \times N \xrightarrow{4^n} i=0 \times N \xrightarrow{5^n} \dots \text{loop}$

    } printf("%i");

}

#N>0

Q8 void fun(int N){

    for(int i=1; i<N; i=i\*2){  
        obs1: i==N & stop  
        print("i")  
    }

}

obs2:

$i=1 < N \xrightarrow{m} i=2 < N \xrightarrow{m} i=4 < N \xrightarrow{m} i=8 < N \xrightarrow{m} i=16 < N \xrightarrow{m} i=32 < N \xrightarrow{m} \dots$

After k iterations code stops

$i=N$        $i=2^k$        $a=b \wedge a=c$   
obs1      obs2       $\boxed{a=b \wedge a=c}$   
 $N=2^k \Rightarrow k=\log_2 N \Rightarrow N=2^k$

Nested loops:

void fun(int N){

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

Nested loops

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

Outerloop = 4 iter

Innerloop = 10 iter

Total = 40 iter

## Nested loops

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

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	

Outerloop = 10

Innerloop = 10N

Total = 10 + 10N

### Nested loops

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

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

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

Outwloop =

Innloop =

Total =

```
void fun(int N) {
    for(int i=1; i<=N; i++) {
        for(int j=1; j<=i; j++) {
            printf("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] = <u>N</u> ite

Outwloop = N

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

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]
	j : [ ]

Outerloop =  
Innerloop =  
Total =

