

Today's Content:

1. Issues with Iterative to compare
2. Asymptotic Analysis
 - a. Big O
 - b. Theta
 - c. Omega
3. How to calculate Big O
4. Issues with Big O
5. Why TLE occurs?
6. Idea from constraints

Note: Iterative ↑ Time ↑

Con: We use iterative to compare 2 Algo's

Q1: Sort arr[N] elements

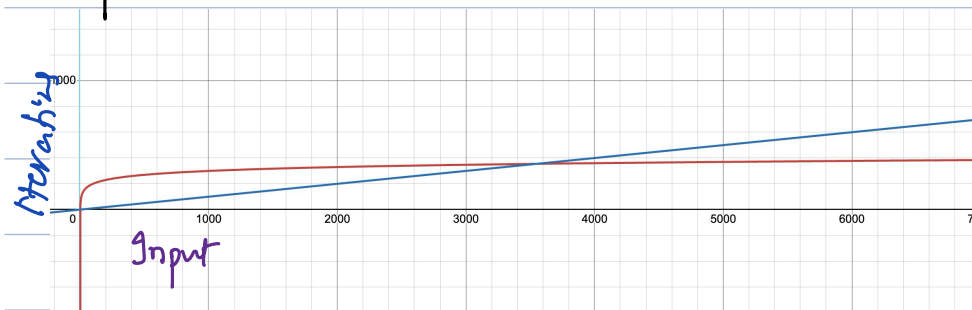
S_1 : Jsort

S_2 : Mersort

Code1: $100 \log_2 N$

Code2: $N/10$

Graph:



obs: Iterative Efficiency

$N < 3500$ Irfan

Nazim

$N > 3500$ Nazim

Irfan

} For large inputs we prefer Irfan's code

Issue: Comparing with iterative is hectic.

Asymptotic Analysis: Analysing performance of algo for very large inputs

1. Big O 2. Omega 3. Theta.

Big O: 1. What 2. Why 3. How.

How to Calculate

1. Calculate iterations based on Input

- ✓ 2. Neglect lower order terms or Consider only highest order term
- ✓ 3. Neglect constant coefficients

According to Big O

Code 1: $100 \log_2 N \rightarrow O(\log_2 N)$
Code 2: $N/10 \rightarrow O(N)$ } $O(\log_2 N)$ is more efficient

Higher Order Terms:

$\log_2 N < \sqrt{N} < N < N \log N < N\sqrt{N} < N^2 < N^3 < 2^N < N! < N^N$

$N=64$

\log_2^{64}	$\sqrt{64}$	64	$64 \times \log_2^{64}$	$64 \times \sqrt{64}$	64×64	64^3	2^{64}	<u>64!</u>	64^{64}
6	8		64×6	64×8					

\downarrow
 $(2^6)^3$
 2^{18}

\downarrow
 2^{64}

Q1 $F(N) = 4N + 3N \log N + 1 : O(N \log N)$

Q2 $F(N) = 4N \log N + 3N\sqrt{N} + 10^3 : O(N\sqrt{N})$

Q3 $F(N) = 4N^2 + 5N \log N + 10^2 : O(N^2)$

Why consider only higher order terms?

→ N: Input

Q. Say $f(n) = N^2 + 10N$ / iterations

Input Size:

Total iterations

% lower order terms

iterations in Total

$$N: 10 \rightarrow N^2 + 10N = 10^2 + 10 \cdot 10 = 200$$

$$\frac{100}{200} \times 100\% = 50\%$$

$$N: 10^2 \rightarrow N^2 + 10N = (10^2)^2 + 10 \cdot 10^2 = 10^4 + 10^3$$

$$\frac{10^3}{10^4 + 10^3} \times 100\% \approx 10\%$$

$$N: 10^4 \rightarrow N^2 + 10N = (10^4)^2 + 10 \cdot 10^4 = 10^8 + 10^5$$

$$\frac{10^5}{10^8 + 10^5} \times 100\% \approx 0.1\%$$

obs: We neglect lower order terms, because it's contribution is negligible

Neglect Constant Coefficient

Algo1

Algo2

For larger inputs
faster algo is

Q1

$10 \log_2 N$

N

Algo1

Q2

$100 \log_2 N$

N

Algo1

Q3

$10^2 \log_2 N$

$N/10$

Algo1

Q4

$10N$

$N^2/10$

Algo1

obs2: Constant coefficient won't affect your comparison, hence we neglect them.

Issue 1: Q: Algo1 Algo2

Iterations: $10N$ N^2

$N=5$: 50 25 : Algo2 is efficient

$N=10$: 100 100 : Same

$N=11$: 10×11 11×11 : Algo1 is efficient

$N=12$: 10×12 12×12 : Algo1 is efficient

BigO: $O(N)$ $O(N^2)$

As per BigO Algo1 is efficient than Algo2 for larger inputs.

Issue 2: If 2 Algo have same BigO, we compare iterations.

Q: Algo1 Algo2

Iterations: $10N^2 + N$ $3N^2 + 10N$ Algo2 is efficient

BigO: $O(N^2)$ $O(N^2)$

Note 1: While calculating BigO, we consider worst case iterations.

Q: Search k in arr[]

bool search(int arr[], int N, int k) { Iterations

```
for(int i=0; i<N; i++) {  
    if(arr[i]==k) return true;  
}  
return false;
```

Best Worst Avg:
1 N

Amortized Analysis

Note 2:

Q: Given an arr[] return max of first 4 elements in array

int max4(int arr[], int N) { // $N > 4$

```
int max = INT_MIN;
```

Iterations; $i = [0, 3] = 4 \Rightarrow$ Constant Iterations.

```
for(int i=0; i<4; i++) {
```

BigO = $O(1) \Rightarrow$ Constant Iterations.

```
    if(arr[i] > max) {
```

```
        max = arr[i];
```

```
    }  
return max;
```

TLE: Time limit Exceeded

Online Editor processing speed = 10^{13}
 $= 10^9 \text{ instructions/sec}$

```
int countfactor(int n){
```

```
    int c=0;  $\rightarrow +1$ 
```

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

$+1$

$+1$

$+1$

Total Instructions = $3 + 5N$

1 iteration = 5 instructions

$+1$

$+1$

```
        if(N%i==0){
```

```
            c++;  $\rightarrow +1$ 
```

```
        }
```

```
    }
```

```
    return c;  $\rightarrow +1$ 
```

Assumption: 1 iteration = 10 instructions.

Our Code = $10^9 \text{ instructions/sec}$

$= 10^8 \times 10 \text{ instructions/sec}$

$= 10^8 \text{ iterations/sec}$

Online server = $10^8 \text{ iterations/sec}$

Con: If code exceeds $> 10^8 \text{ iterations} = \text{TLE}$.