Time Complexity 2

Close approximations

Q: Given some input, to write an algorithm to sort the data.

Same input

Nishant (nisort)

Sourabh (sortify)

Execution time

Hardwork

15 sec (Macbook)

20 sec (Samsung phone)

macbook.

15 sec (python)

12 sec (C++)

Language

C++

physical factors

Antarct.

10 sec.

12 sec (Volcano)

10 sec

9 sec

Execution time is not a good factor.

S/W + H/W + outside factors

(i=0 ; i<N ; i++){ → i [0, N-1] : (N iterations)

. . . .

}

Iterations : Given N elements, sort the data.

N → input

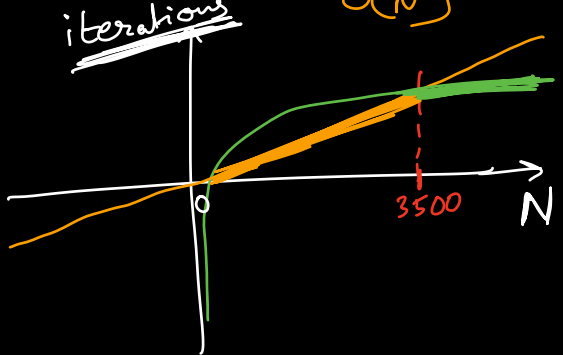
Danyaal
dSort

Praveen :
Psort

iterations :

(100 log N)
↓
O(log N)

(N/10)
↓
O(N)



Till (N <= 3500),
Psort is better

After N > 3500,
Danyaal's algo is better

Hotstar :
$3 * 10^7$

Despacito
$7 * 10^9$

Google :
7, 579. . . . .

Asymptotic analysis:

↓

performance of your algorithm for very large inputs

↓

Big(O) Notation for an algorithm

↳ ① Calculate iteration. ✓

② Neglect lower order terms ✓

③ Neglect constant coefficients.

Kiran ⇒ SuperSort

N → input.

iterations : $N^2 + 10N$

Total iteration ; % of 10N in total iteration.

$N = 100$: $10^4 + 10^3$; $\frac{10^3}{10^4+10^3} * 100 \simeq 10\%$.

$N = 10^5$: $10^{10} + 10^6$; $\frac{10^6}{10^{10}+10^6} * 100 \simeq 0.01\%$.

0.00001%

$N = 10^{50}$

// Sort N iteration

Gopika
G Sort

aishwarya
aishu sort

Ex 1.

$10 \log N$

$10 * 3$

$N = 10$.
$N = 10^{10}$.

$10 * 30$ <<<

N

10.
10^{10}.

Ex 2.

$10^3 \log N$.

$10^5 \log N$.

$10^{10} \log N$

N

$N = 10^{50}$.

inter

Ex 3.

$10^4 N + 10^6$

$N = 10^{50}$

$10^{54} + 10^6$

N^2

100
10

Ex 4

$10^4 N \log N$

N^2

Issues with Big(O).

⇒ Task to sort N element.

	Noufal (NoSort)	Praveen (quicker sort)
Iterations:	$100N$	N^2
	⇓	⇓
	$O(N)$	$O(N^2)$.

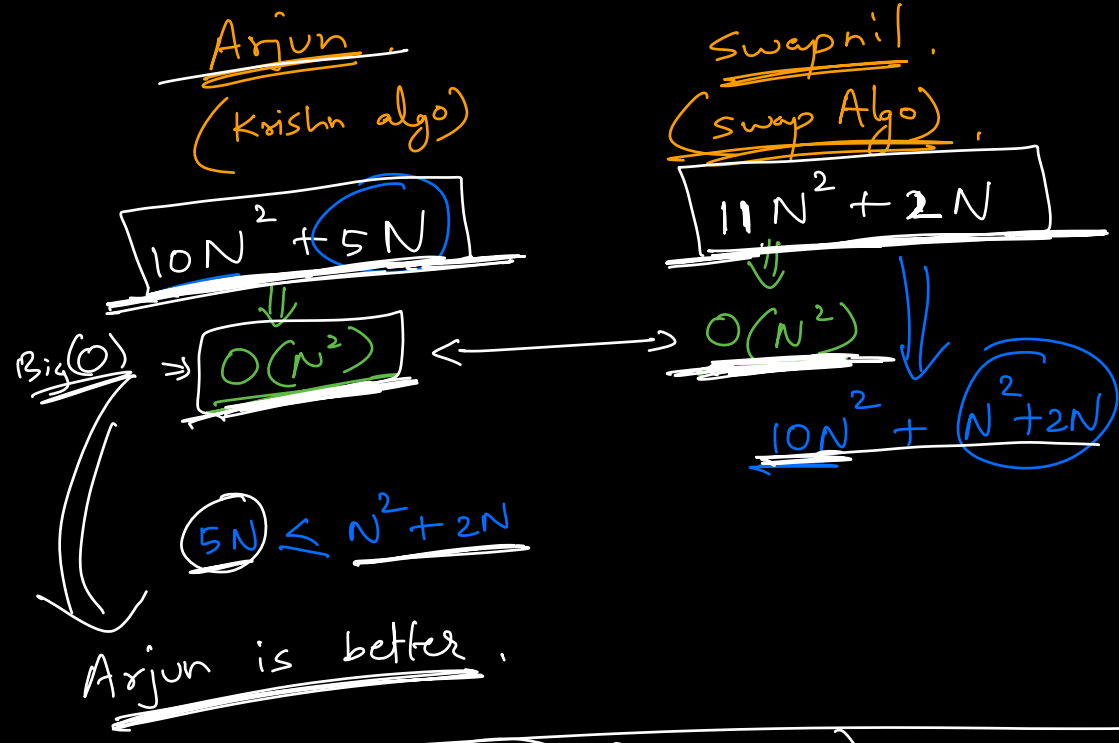
Noufal's algo is better.

N:			
10 ⇒	10^3.	10^2.	Praveen wins.
99 ⇒	9900	9801.	Praveen wins
100 ⇒	10^4	10^4	⟹
101 ⇒	$101*100$	$101*101$	

$N > 100$ ← Noufal's efficient.

Big(O) comparison holds after certain threshold!
(very large inputs)

// N inputs to sort.



Break: (1 :00 PM)

Time Complexity

Space Complexity → Big(O).

Space Complexity

Write a function, & calc.

```
func (int N)
{
    int    x = N
    int    y = x^2
    long   z = x+y
    double pi = 3.14
    ...
}
```

int → 4 byte
long → 8 byte
double → 8 bytes.

Total memory = 24 bytes.

⇒ Constant space

$O(1)$ space

```
func (int N)
{
    int    x = N
    int    y = x^2
    long   z = x+y
    double pi = 3.14
    int arr[N]
}
```

⇒ 24 bytes

int arr[N] → N integers → 4N bytes.

Total memory = (24 + 4N) bytes

$O(N)$

```
{ func (int N)
    ...
    int arr[N]
    bool mat[N][N]
}
```

mat[N][N] → N^2 elements → 1 byte

Total memory = $(24 + 4N + N^2)$ bytes

$O(N^2)$

// Given array of N elements, calculate the sum of all elements.

```
SumA( int arr[ ], int N)
{  sum = 0;
   (i=0; i<N; i++){
    sum = sum + arr[i]
}
return sum
}
```

O(N) to store the array 4N bytes.

int N → 4 bytes.

Input : 4 + 4N bytes

S.C : auxillary ⇒ Extra.

sum, i ⇒ 4 + 4 ⇒ 8 bytes.

S.C ⇒ O(1)

4 bytes

T.C ⇒ O(N)

```
SumA (arr[ ], N)
{
   int temp[N];
   for (i = 0 → N)
      temp[i] = arr[i]
```

4N bytes

Doubts.

Print.

// Space Complexity : Amount of Extra Space taken by your algorithm other than input space.

Example :

```
func ( int arr[], int N, int K)
{
    for (i=0; i<N; i++)
    {
        if (arr[i] == K){
            return True
        }
    }
    return False
}
```

iterations.

best: 1

worst: N

T.C : O(N)

S.C : 4 bytes
O(1)

TLE ⇒ time limit exceeded } code is taking a lot of time.

Sidarth

↳ Test Link : 60 mins

① ——— (submit) TLE → (submit) TLE → (submit) AC
opt① opt②

55 mins are over.

② ———

10^8 iteration → ~1 sec

$N = 10^6$.

for (i = 0 → N) $O(N^2)$

$N\sqrt{N}$ → 10^{12}.

$10^6 * 10^3$ → 10^9

$N \log N$ ⇒ $10^6 * 20$ ⇒ 2×10^7

1 Advanced

// power => TC : O(1)

```
func(N, K)
{
    for(i=1 ; i<=N ; i++){
        p = power(i,K)
        for(j=1 ; j<=p ; j++)
        {
            print()
        }
    }
}
```

$\Rightarrow (i)^K$

① $O(N^K)$
② $O(N * 2^N)$
③ $O(N * K)$
④ $O(N^{K+1})$
⑤ $O(N^3)$

// Table.

i	j	iterations.
1	j : [1 ... 1^K]	1^K
2	j : [1 ... 2^K]	2^K
3	j : [1 .... 3^K]	3^K
⋮	⋮	⋮
N	j : [1 .... N^K]	N^K

Total iterations :

$1^K + 2^K + 3^K + \cdots + N^K$

		Total iter	Big(O)
if K=1	: $[1 + 2 + 3 + \cdots + N]$	: $\frac{N(N+1)}{2} \Rightarrow \frac{N^2}{2}$ (coefficient of highest term)	$O(N^2)$
K=2	: $[1^2 + 2^2 + 3^2 + \cdots + N^2]$	: $\frac{N(N+1)(2N+1)}{6} \Rightarrow \frac{N^3}{3}$	$O(N^3)$
K=3	: $[1^3 + 2^3 + 3^3 + \cdots + N^3]$	: $\left[\frac{N(N+1)}{2}\right]^2 \Rightarrow \frac{N^4}{4}$	$O(N^4)$

$K = 3$

K : $[1^K + 2^K + \cdots + N^K]$: $\Rightarrow \frac{N^{K+1}}{K+1}$ ~~$O(N^{K+1})$~~

Only the highest order term.

inputs : N, K

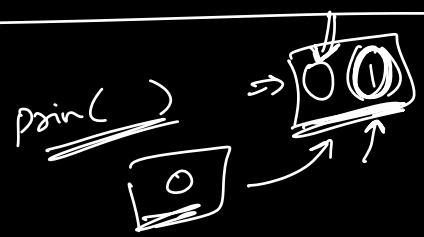
$K \rightarrow$ large

$n \rightarrow$ small

$O\left(\frac{N^{K+1}}{K+1}\right) \rightarrow N^K$

$O(N^2) \simeq O(N^3)$ $\quad N^K \not\simeq N$

Doubts.



GP. $a, ar, ar^2, ar^3 \ldots ar^{n-1}$

iterations : $N + \frac{N}{2} + \frac{N}{4} + \frac{N}{8} + \cdots \frac{N}{2^{\log N}} \rightarrow \frac{N}{N} =$

```
for( i = N; i > 0; i = i/2)
{
    for (j = 0 -> i)

}
```

$i = N \rightarrow N$ +

$i = N/2 \rightarrow \frac{N}{2}$ +

$i =)$ $\quad$:

$$N + N\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots \frac{1}{2^{\log N}}\right)$$

$\frac{1}{4} = \frac{1}{2}$ $\frac{1}{8} = \frac{1}{2}$ ... G.P

$$\frac{a(1-r^N)}{(1-r)}$$

T.C

Asymptotic recursion

$(N + \frac{N}{2} + \cdots$

ne $\log(\log N + 1)$

H.W

Binomial theorem

$$(N+1)^K = {}^KC_0 N^K + {}^KC_1 N^{K-1} + \cdots {}^KC_K 1$$

$(N+1)^K - N^K =$

$+ (N)^K - (N-1)^K$

$N \rightarrow$ O(1)

$O(N)$

$C_1 * N + C_0$