Close approximations

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

Same inputs

Nishant (nisort)

Execution time

15 sec (Macbook)

Hardwork

↓

Language

15 sec (python)

↓ c++

Physical factors

Antarcti.

10 sec.

↓

10 sec

Sourabh (sortify)

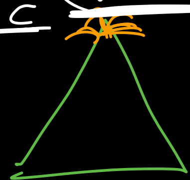
20 sec (Samsung phone)

↓ macbook.

12 sec (C++)

↓

12 sec (Volcano)



↓

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)

. . . .

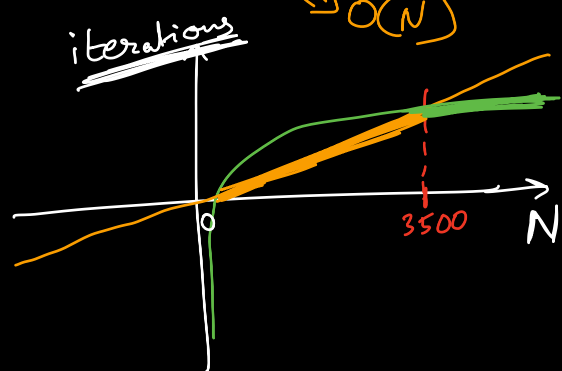
}

Iterations : Given N elements, sort the data.

N → input

Danyaal	Praveen
dsort	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:

$\Downarrow$

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 $\Rightarrow$ Super Sort

$N \rightarrow$ input.

iterations : $N^2 + 10N$

Total iteration ; % of 10N in total iteration.

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

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

0.00001%

$N = 10^{50}$

// sort N iteration

Gopika	aishwarya
G sort	aishu sort

Ex ①. $10 \log N$ $\quad$ N

$N = 10.$ → $10 * 3$ $\quad$ $10.$

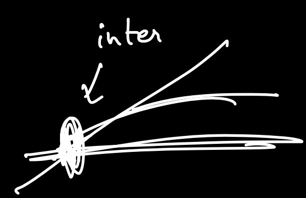
$N = 10^{10}$ $\quad$ $10*30$ <<< $10^{10}.$

Ex ②. $10^3 \log N.$ $\quad$ N

$\boxed{10^5} \log N.$

$\boxed{10^{10}} \log N$ $\quad$ N

$N = 10^{50}.$



Ex ③. $10^4 N + 10^6$ $\quad$ N^2

$N = 10^{50}$

$10^{54} + 10^6$ $\quad$ $\cancel{10^{100}}$

Ex ④ $10^4 N \log N$ $\quad$ N^2

Issues with Big(O).

⇒ Task to sort N element.

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

$\boxed{\text{Noufal's algo is better}}$

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

$\boxed{N > 100}$ ← Noufal's efficient.

Big(O) comparison holds after certain threshold!

(very large inputs)

// N inputs to sort.

Arjun.

(Krishn algo)

$10N^2 + 5N$

$\Downarrow$

Big(O) $\Rightarrow$ $O(N^2)$ $\longleftrightarrow$ $O(N^2)$

$5N \leq N^2 + 2N$

Arjun is better.

Swapnil.

(swap Algo).

$11N^2 + 2N$

$\Downarrow$

$10N^2 + N^2 + 2N$

Break: (1 :00 PM)

Time Complexity

Space Complexity

$\rightarrow$ 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

⇓

$\boxed{O(N)}$

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

bool → 1 byte

mat[N][N] → N^2 elements

Total memory

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

⇓

$\boxed{O(N^2)}$

// 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.

$\boxed{S.C}$: auxillary ⇒ Extra.

sum; & i ⇒ 4 + 4 ⇒ 8 bytes.

S.C ⇒ $\boxed{O(1)}$

4 bytes

$\boxed{T.C \Rightarrow O(N)}$

Doubts.

print.

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

4N bytes

// 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) opti ①→ TLE —(submit) opt ②→ $\boxed{AC}$

55 mins are over.

② ________

10^8 iteration → $\boxed{\sim 1 \text{ sec}}$

$N = 10^6$

for (0 – N)

0 – N

$O(N^2)$

$N\sqrt{N}$

10^{12}.

$10^6 * 10^3$

$\boxed{10^9}$

$N \log N \Rightarrow 10^6 * 20 \Rightarrow 2\times10^7$

1 Advanced

// power => TC : O(1)

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

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 : $[1 \cdots 1^K]$	1^K
2	j : $[1 \cdots 2^K]$	2^K
3	j : $[1 \cdots 3^K]$	3^K
⋮	⋮	⋮
N	j : $[1 \cdots N^K]$	N^K

Total iterations :

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

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

$K = 3$

$\dots \quad \frac{N(N)}{5} + \dots$

$K \quad : \quad \{1^K + 2^K + \dots + N^K\} : \quad \Rightarrow \frac{N^{K+1}}{K+1} \quad \cancel{O(N^{K+1})}$

Only the highest order term.

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

inputs : N, K

$K \to$ large

$n \to$ small

N^K

$O(N^2) \approx O(N^3)$ $\quad N^K + N$

Doubts

prin( ) $\Rightarrow$ 0 0 1 0

G.P. $\quad a, ar, ar^2, ar^3 \dots ar^{n-1}$

iterations : $\left[N + \frac{N}{2} + \frac{N}{4} + \frac{N}{8} + \dots \frac{N}{2^{\log N}}\right] \Rightarrow \frac{N}{N} =$

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

$i = N \to N$ +

$i = N/2 \to \frac{N}{2}$

+

$\vdots$

$i = 1 \quad\quad 1$

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

$$\frac{1/4}{1/2} = \frac{1}{2} \qquad \frac{1/8}{1/4} = \frac{1}{2} \quad \cdots \quad \boxed{G.P}$$

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

$\boxed{T.C}$ for recursion

$$\left(N + \frac{N}{2} + \cdots\right)$$

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

$\boxed{H.W}$ $\boxed{\text{Binomial theorem}}$

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

$$(N+1)^K - \cancel{N^K} =$$

$$+ \cancel{(N)}^K = \cancel{(N-1)}^K$$

$$\infty \longrightarrow N$$

$\boxed{O(1)}$.

$O(N)$

$C_1 * N + C_0$