Asymptotic Analysis

Input → ∞

Looking behaviour or your algorithm

100 users
→ 1000 users
→ 10000 users

Time and Space

$n \to \infty$

$T(n) = k_1 n^2 + k_2$

$T(n) = k_1 n^2 + k_2 n + k_3$
$= k_1 n^2 + m + k_3$

10^{10}
$(10^{10})^2$ (10^{10})
$(\to \infty)$

Rules

1) Drop the sign insignificant terms
2) Drop the constants

n^{10000}

$T(n, m) = n^2 + m + 4$ $\quad 2^n, \; n!$

$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2)$
$< O(n^3) \cdots < O(2^n) < O(n!) \cdots$

defined on upper bound

$\log.o$: Upper Bound
$\log.\Omega$: Lower Bound
$\log.\theta$: Exact bound

$T(n) = O(n^2)$, $O(n^3)$, $O(2^n)$, $O(n!)$

$\Omega(n^2)$, $\Omega(n^3)$, $\Omega(1)$, $\Omega(\log n)$ -

$\Theta(n^2)$

Bracket definition of $\log.o$
=
Theoretical definition of $\log.\theta$

$i = 0$
$j = 0 \to n-1$
$i = 1$
$j = 0 \to n-1$
⋮
$i = n-1$
$j = 0 \to n-1$

$T(1) = O(1)$
$= O(1)$
$T(10^2) = O(10)$
$= O(1)$

Print an integer: $O(1)$

SE

Print a string of size k: $O(k)$

"abc" : $O(3)$
" " : $O(10^2)$

abc
$\neq$ 1

abd
$\neq$ 1

Part-1: Sort each string individual

Sorting 1 string: $O(n \log n)$
" " " : $O(n \log n)$

Part-2: Sort a list of strings

Sorting a list of size: $= O(n \log n)$
Comparing 2 size $= O(1)$
No. of comparisons $= x = O(n \log n)$

&

Comparing 2 strings $= O(n)$
No. of comparisons $= O(n \log n)$
$\Rightarrow$ Sorting a list of strings $= O(n n \log n)$

$\Rightarrow$ Overall TC $= O(n n \log n + n n \log n)$

```
Input Number: 10
For List: 1 2 3 4 5 6 7 8 9 10
Output: 1 2 3 4 5 6 7 8 9 10
Input Number: 5
For List: 1 2 3 4 5
Output: 1 2 3 4 5
```

$i = 0$: $n-1$
$j: 1 \to n-1$

$i = 1$: $n-2$
$j: 2 \to n-1$

$i = 2$: $n-3$
$j: 3 \to n-1$
⋮

$i = n-1$: 0
$j: n \to n-1$

$0 + 1 + 2 + 3 + \cdots + (n-3) + (n-2) + (n-1)$

AP

Sum of first n natural nos. $= \frac{n(n+1)}{2}$

$T(n) = \frac{(n-1)(n)}{2} = \frac{n^2 - n}{2}$

$T(n) = O(\cancel{2} \times n^2 - \cancel{1} \times \cancel{n})$
$= O(n^2)$

GP

First term: a
multiplier: r
no. of terms: n

Sum: $\frac{a(r^n - 1)}{r - 1}$

Linear Search

Best-case: First element: $= O(1)$

Worst-case: Every not present $= O(n)$

Avg-case:

0th index $= 1$
1 index $= 2$
2 index $= 3$

$n-1$ index $= n$
not present $= n+1$

Sum $= (1 + 2 + 3 + \cdots + (n+1))$
$= \frac{(n+1)(n+2)}{2}$

No. of cases $= (n+1)$

Avg $= \frac{\frac{(n+1)(n+2)}{2}}{(n+1)} = \frac{(n+2)}{2}$
$= O(n)$

Quick Sort

Avg-case $= O(n \log n)$
Worst-case $= O(n^2)$

- Space-Complexity = I/P space + Aux space + O/P space
- Aux-space = Extra space needed to solve an algo

```
Vector<int> fun(vector<int a1>> vec) {//func
    vector<int> temp;
    vector<int>* ptr;
    return temp;
}
```

AS $= O(\text{fun(temp)}) = O(1)$

SC $=$ I/P space $= O(n \times m)$
Aux space $= O(1)$
O/P space $= O(1)$

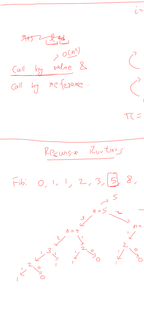
$= O(n \times m + 1 + 1)$

$O(n)$

Call by value &
Call by reference.

$i = n$
⋮
$i = 2$
$i = 1$
$i = 0$

TC $= O(n)$

Recursion Runtimes

Fib. 0, 1, 1, 2, 3, 5, 8, 13, 21, 34 ...