input $\rightarrow \infty$

1) How will the code will perform for larger inputs?

2) How to compare 2 pieces of code meant for the same purpose?

---

i: 0 → n-1
print (i)

$O(n)$

i: 0 → n-1
print(n-i)

$O(n+n) = O(2n)$
$= O(n)$

i: 0 → 5
print (i)

$O(\cancel{2}n + \cancel{5}) = O(n)$

$O(n^2 + \cancel{n}) = O(n^2)$

32- bit

10^2 - 1000 - 5 - 0 - 1000000

printing int → $O(32) = O(1)$

'a' 'c'

"abc" "- - - 10^6 - - - "

print string → $O(len)$

[[abc, def, hello, ·]
[-]
[- -]
[·]]

$O(n \times m \times (\text{maxlen of any str. in the matrix}))$

a c
ab ac
ab bc

"abcdef" "abcdff"

Comparing 2 strings: $O(len)$

I/P: n [abd, dfg, bca, ddb]. string lens = m.

Step-1: [abd, dfg, abc, bdd].

Step-2: [abc, abd, bdd, dfg].

TC: $O((n \cancel{\log n})^2)$

step-1: $O(n\, m \log m)$ ✓

step-2: $O(m\, n \log n)$ | $O(n \log m)$ ✗

Overall = $O(nm \log m + nm \log n) = O(nm \log(nm))$.

Sort a list of ints: $O(n \log n)$.

Sorting is a sequence of comparisons.

Overall: No. of comparisons $= k$
Comparing 2 ints $= O(1)$

Sorting a list of ints $= O(n \log n)$

No. of comparisons $= O(n \log n)$

```cpp
void func(int n) {
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            cout << i + j << endl;
        }
    }
}
```

i=0 : j = 1 → n-1 → (n-1)
i=1 : j = 2 → n-1 → (n-2)
i=2 : j = 3 → n-1 → (n-3)
⋮
i=n-2 : j = (n-1) → (n-1) → 1
i=n-1 : 0 → 0

Overall: $1 + 2 + 3 + 4 - - - - + (n-1)$
$= \frac{(n-1)(n)}{2}$ no. of iterations.

$O\left(\frac{n^2}{\cancel{2}} - \frac{\cancel{n}}{\cancel{2}}\right) = O(n^2)$.

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

Sum of GP ~~AP~~: $\frac{a(r^n - 1)}{r - 1}$

Sum of AP: TODO.

Big-O: Gives an upper bound.

i: 0 → n-1
print (i)

$O(n)$ / $O(n^2)$ / $O(2^n)$ - - ✓

$O(\log n)$ / $O(1)$ ✗

Big-Omega Ω → Gives a lower-bound.

$\Omega(\log n)$ / $\Omega(1)$ ✓
$\Omega(n^2)$ / $\Omega(2^n)$ - - ✗

Big-Theta Θ → Exact bound.

$\Theta(n)$ ✓

$fib(n) = fib(n-1) + fib(n-2)$.