CSC263 Tutorial — Jan 12, 2018

- Topic: running time

- $T(n)$ - worst-case running time; max number of steps the algorithm takes on an input of size n
- $t(x)$ - returns # of steps taken by algo on input x
- $T(n) = \max\{t(x) \mid |x| = n\}$

- O, Ω, Θ ; mathematical concepts that quantify relations between functions
  - $O(n^2) = \{1, 2, n^2, 2n^2, n^2+1 \ldots etc\}$ = less than, within constant factor
  - $\Omega(n^2) = \{n^2, 2n^2, 2n^3 \ldots etc\}$ = all functions $\geq n^2$ within constant factor
  - $\Theta(n^2) = O(n^2) \cap \Omega(n^2)$
  - adopted by computer scientists to compare algos with details abstracted

- $T(n) \in O(g(n)) \iff$ for some constant c, for <u>every</u> input of size n, the algo takes <u>at most</u> $c \cdot g(n)$ steps
- $T(n) \in \Omega(g(n)) \iff$ for some constant c, there is <u>some</u> input of size n for which the algo takes <u>at least</u> $c \cdot g(n)$ steps

Ex 1] input: $A[1 \ldots n]$

```
for i=1 to n:
    for j=1 to n:
        if A[i]*1 STOP
```

→ $O(n^2)$ = can only

$\Omega(n^2)$ = for all n, always an array of all 1's will cause runtime n^2

loop invariant: at the end of ith iteration of while loop $A[last+1 \cdots n]$ contains largest elements in A, sorted

Ex 2] BubbleSort($A[1 \ldots n]$)

```
- last=n, sorted=false
- while not sorted:
    sorted=true
    for j=1 to last ind-1
        if A[j]>A[j+1]
            swap A[j], A[j+1]
            sorted=false
    last=last-1
```

$T(n) \in O(n^2)$

→ while loop executes at most $n-1$ times (each loop reduces 'last' by 1 & when 'last'=1 loop is not executed)

→ inner for loop executes at most $n-1$ times

$T(n) \in \Omega(n^2)$ consider reverse sorted A, after 1 iteration of while loop, $A[n]$ largest (via LI) & took $n-1$ swaps....

Total swaps $= \left[\sum_{i=1}^{n-1} n-i\right] = \frac{(n-2)(n-1)}{2} \propto n^2$