# Asymptotic Analysis

- Runtime Analysis
- Big Oh - $O(\cdot)$
- Complexity Classes and Curse of Exponential Time
- $\Omega(\cdot)$, $\Theta(\cdot)$, $o(\cdot)$, $\omega(\cdot)$ - Relational properties

Imdadullah Khan

# Analysis of Algorithms

- Algorithm analysis is the theoretical study of performance and resource utilization of algorithms
- How "goodness" of algorithms can be measured?
  - Time consumption
  - Space and memory consumption
  - Bandwidth consumption or number of messages passed
  - Energy consumption
  - $\vdots$

# How to measure runtime?

- Clock-time of algorithm execution is not a suitable measure
  - Depends on machine/hardware, operating systems, other concurrent programs, implementation language and style etc.
  - We want platform independent and implementation Language independent
- Number of operations is the right framework
  - Time complexity is measured in terms of number of elementary operations
  - Assuming computation of each elementary operation takes fixed amount of time
  - Important to decide which operations are counted as elementary

# Runtime as a function of input size

We want a consistent mechanism to measure efficiency that is

- Platform independent
- Language independent
- Has predictive value with respect to increasing input sizes
- We measure runtime by number of elementary operations as a function of size of input
- Size of input: usually number of bits needed to encode the input instance, can be length of an array, number of nodes in a graph etc.

# Best/Worst/Average Case

- For inputs of fixed size ($n$) there could be different runtimes depending on different instances

- Recall the parity test of Odd/Even integers

# Parity Test: Odd/Even integer

**Input:** An integer $A$
**Output:** True if $A$ is even, else False

If $A$ is given in an array

| 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| 4 | 6 | 9 | 2 | 7 | 5 | 8 |

$A =$ (array above)

**if** $A[0] \in \{0, 2, 4, 6, 8\}$ **then**
  **return** true

**if** $A[0] = 0$ **then return** true
**else if** $A[0] = 2$ **then return** true
**else if** $A[0] = 4$ **then return** true
⋮
**else** **return** false

Number of comparisons is different for $A[0] = 0$ and $A[0] = 8$

# Best/Worst/Average Case

- For inputs of fixed size ($n$) there could be different runtimes depending on different instances
- Let $T(I)$ be the time, algorithm takes on instance $I$

Best case runtime: $t_{best}(n) = \text{MIN}_{I:|I|=n} \left\{ T(I) \right\}$

Worst case runtime: $t_{worst}(n) = \text{MAX}_{I:|I|=n} \left\{ T(I) \right\}$

Average case runtime: $t_{av}(n) = \text{AVERAGE}_{I:|I|=n} \left\{ T(I) \right\}$

In general, we consider the worst case runtime

# Asymptotic Notation

- We use asymptotic analysis of functions for running time
- Characterize running time for all inputs instances of a certain size (so worst-case) with just one runtime function
- Small inputs are not much of a problem, we want to learn behavior of an algorithm on large inputs

# Asymptotic Notation

Our foremost goals in analysis of algorithms are

Determine running time of algorithms on inputs of **large size**

Determine how the runtime **grows with increasing inputs**

How the runtime changes when input size is doubled/tripled?