Introduction to Data Science

(Lecture 21)

Dr. Mohammad Pourhomayoun

Assistant Professor

Computer Science Department

California State University, Los Angeles


ALSO
BIG
DATA
TOOLS
DISK
TARGET
APPLIED
SENSOR
SHARED
DEFINITION
CURRENT
MAY
MOVING
WITHIN
EXAMPLES
EVERY
RECONSIDER
PRACTITIONERS
CAPTURE
ZETTABYTES
THOUGHT
INTERNET
BUSINESS
SETS
GENOMICS
DISTRIBUTED
DESCRIBING
RADIO-FREQUENCY
COMPLEXITY
MANAGEMENT
TERABYTES
CASE
ABILITY
PETABYTES
INCLUDE
ELAPSED
SYSTEMS
TOLERABLE
SIZE
STORAGE
MPP
PARALLEL
SAN
QUALITIES
SINCE
MASSIVELY
GROW
CONNECTOMICS
ORGANIZATIONS
SET
USE
RELATIONAL
LARGER
CONTINUES
SOCIAL
INDEXING
TENS
COMPLEX
CITATION
ANALYTICS
NOW
BIOLOGICAL
PROCESSING
UBIQUITOUS
CAPACITY
BURIED
WORLD'S
HUNDREDS
DESKTOP
RECORDS
NETWORKS
DATABASES
SOLID
TYPES
CURRENTLY
GARTNER
WORKING
AMOUNT
OPPORTUNITIES
SEARCH
DIFFICULTY

Why Is Data Science So Important Now?

- Why is Data Science an important topic these days?

  (why didn't anyone talk about it 10 years ago?)

- Because now we have:
  1. New Sources of Data that did not exist before.
  2. New Capabilities to acquire, store, and process data.
  3. New Techniques in Data Analytics and Processing.

New Sources of Data

- Social Networks: Facebook, Twitter, …
- World Wide Web
- Online Activities: Amazon, ebay, …
- Smart Phone Activities
- Electrical Health Records (EHR)
- Body and wearable sensors
- …

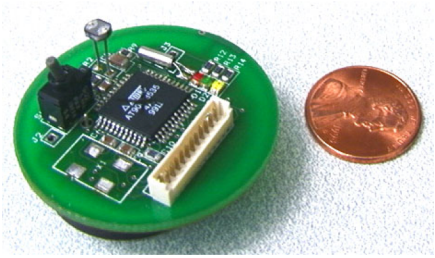
New Sources of Data

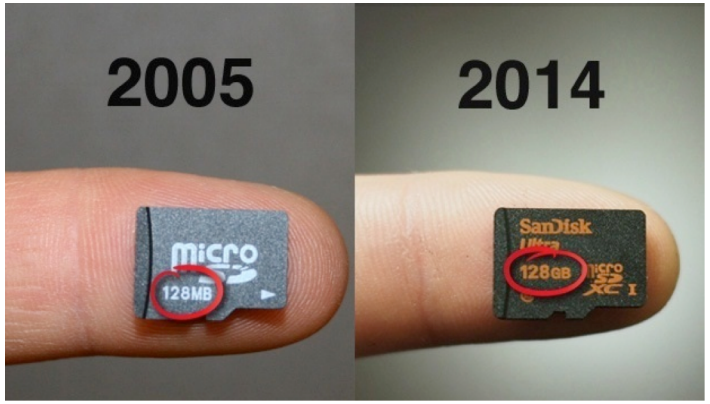
- "There was 5 exabytes of information ($5x10^{18}$ bytes) created between the **dawn of civilization through 2003**, but that much information is now created **every two days**."

- Eric Schmidt, Google, Alphabet

New Capabilities to Acquire & Store Data

- The cost of data acquisition and storing data has dropped incredibly fast thanks to advances in Electronics and other technologies.





New Capabilities to Acquire & Store Data

- New technology allows for designing and building small and cheap Sensors, Memory devices, Communication devices, and CPUs to collect, store, transmit, and process data.
  - Sensor Networks, Environmental Sensors, Body Sensors (Wearables), …
  - Life Sciences, Astronomy, Oceanography, …



New Challenge:

- The cost of analyzing data, extracting useful knowledge from it, and learn from it is the new big bottleneck.

What is Big Data?

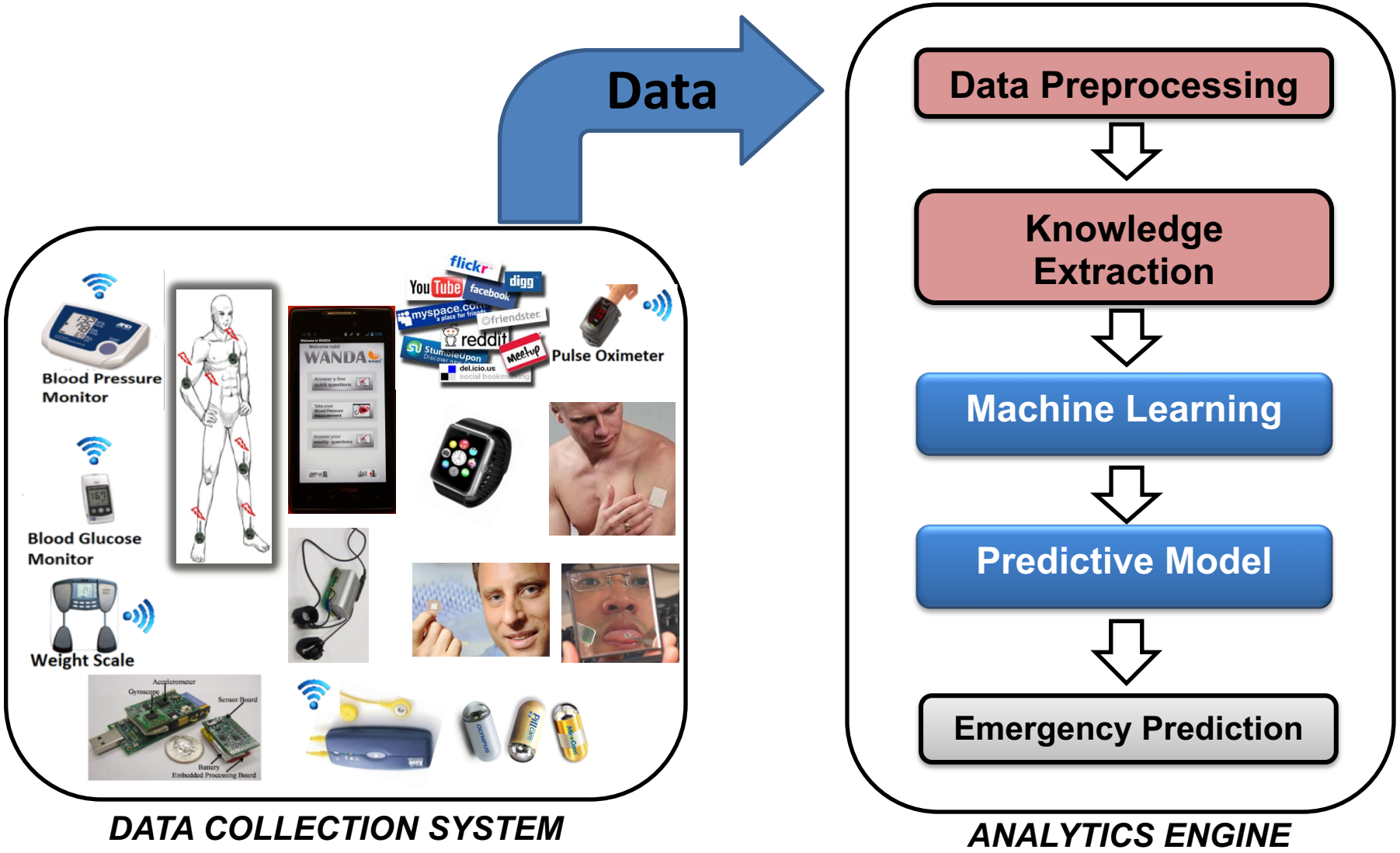
- **Big Data** is any data that is expensive to manage, store or transfer, or difficult to process and extract value from.

Big Data Characteristics

- The Three **V**'s of Big Data (3Vs):
  - **Volume**
    - The size of the data
  - **Velocity**
    - The latency of data processing relative to the growing demand for interactivity
  - **Variety**
    - The diversity of sources, formats, quality, structures

Example: Big Data Analytics for Stock Market Prediction

Example: Big Data Analytics for Healthcare



Example: Online Shopping

amazon

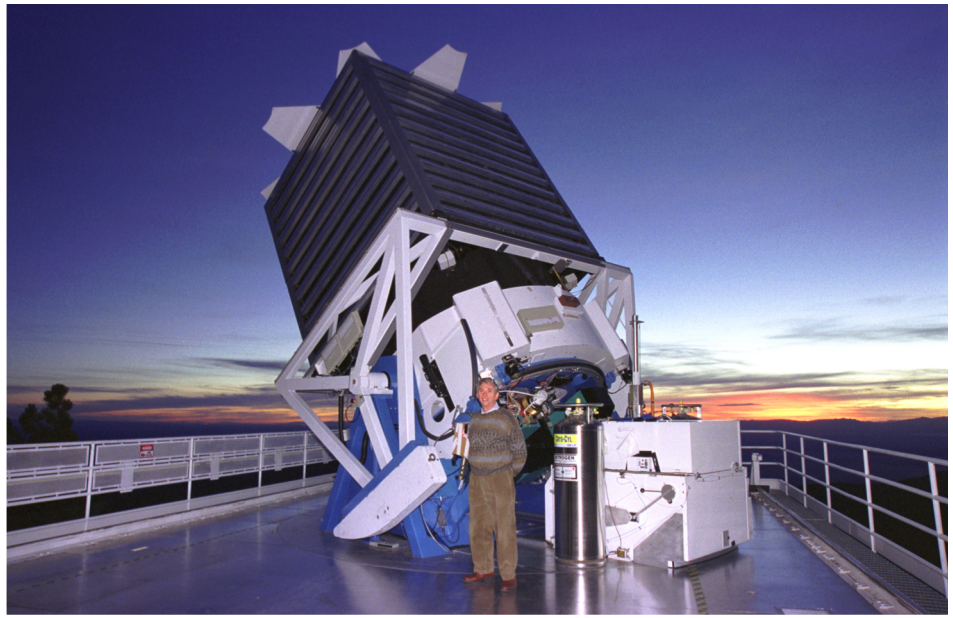
- **Customers Who Bought This Item Also Bought:**

+

+

Example: Astronomy

- Apache Point Telescope:
  - About 80TB (80x 10^{12} bytes) of raw image data.

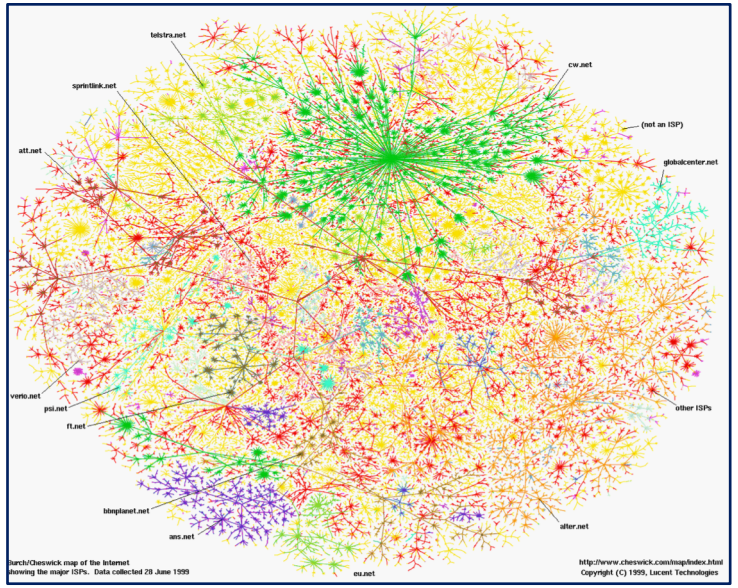


Example: Oceanography

- Cornell University, Bioacoustics Research Lab:
  - About 1 PB (1000 TB = 10^{15} bytes) audio data collected during a 10 year period from underwater sensors (hydrophones).

Example: WWW

- The Web:
  - More than 1 billion web pages
  - Google web index: 100 PB (100×10^{15} bytes)



Example: Social Networks

- Facebook:
  - The Facebook's **daily** logs: 60 TB

In only One Second …

In one second on the Internet there are...

http://www.internetlivestats.com/one-second/

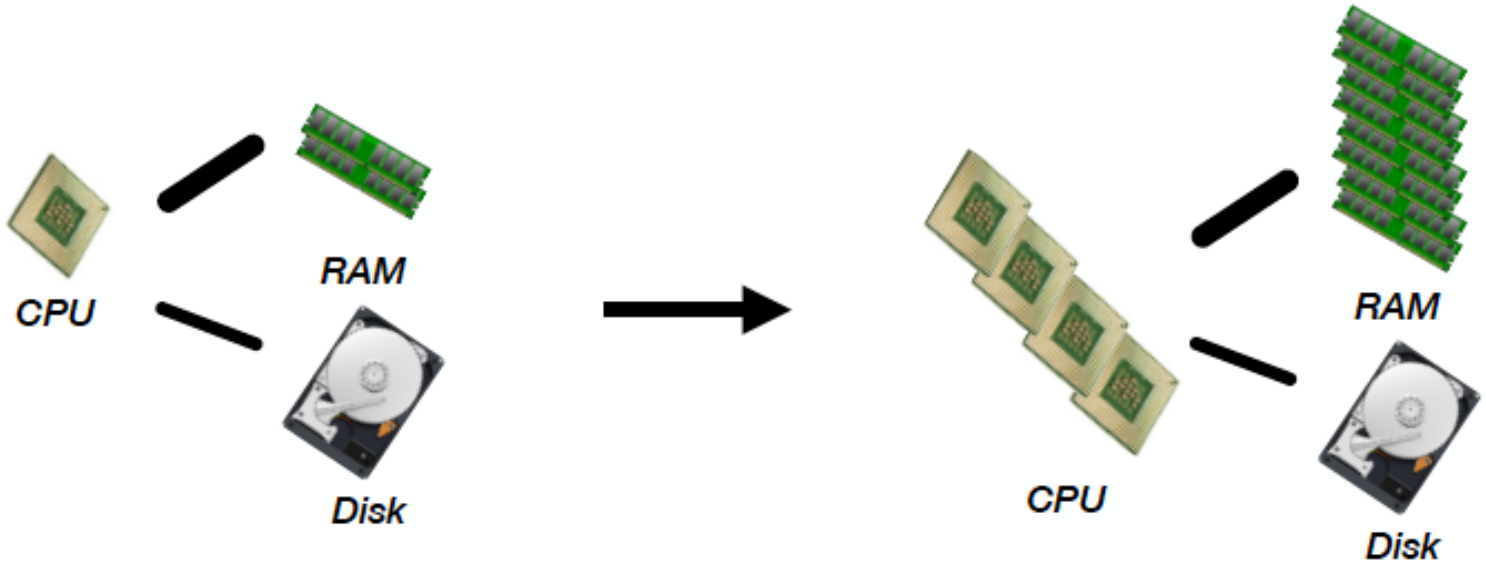
Scalability

- ***Scalability*** is the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged in order to accommodate that growth (wikipedia).

- Two main approaches for adding more resources for a particular application:
  - **Scale-Up**
  - **Scale-Out**

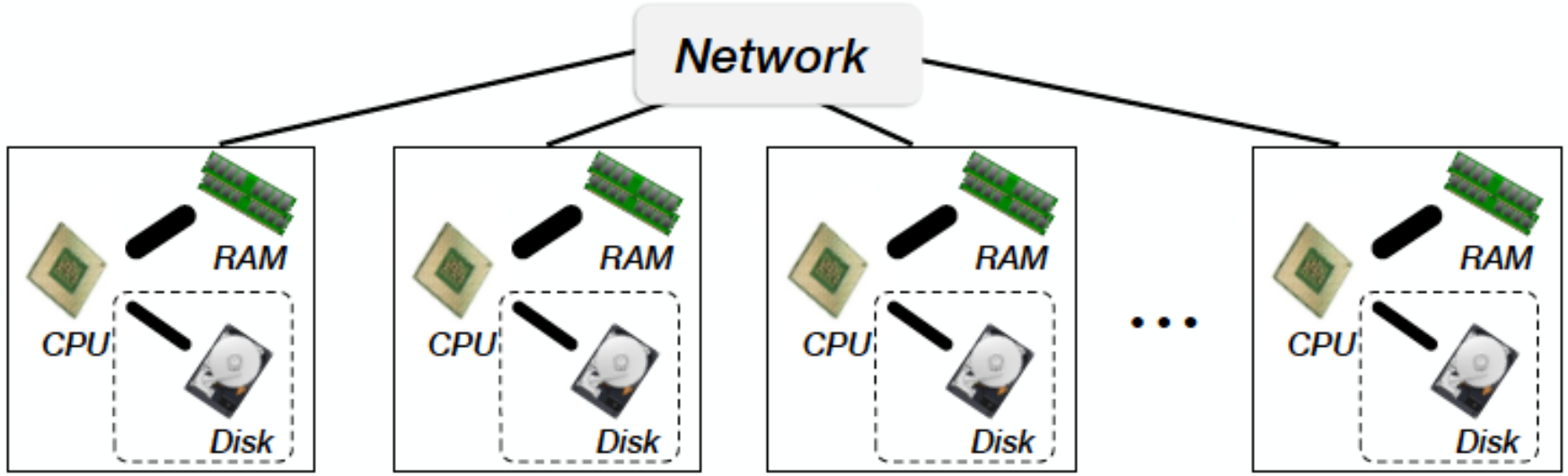
Scale-Up

- Scale-up (one big machine):
  - Can be very fast for medium scale problems
  - Expensive, specialized hardware
  - Eventually hit a wall



Scale-Out

- Scale-out (distributed, e.g., cloud-based):
  - Scalable to massive problems
  - Need to deal with network communication
  - Added software complexity



Dimensionality Reduction

- You may need to use "**dimensionality reduction**" methods to reduce feature redundancy and consequently, to reduce the number of features used in your prediction algorithm.

- **PCA (Principal Component Analysis)** is one of the most popular techniques for unsupervised dimensionality reduction.

- We provide more details about "dimensionality Reduction" algorithms and methods next semester in CS4662/CS5661.

Dimensionality Reduction (DR) based on PCA

- **from sklearn.decomposition import PCA**
- **n = 10** # (n is the number of components (new features)
  # after dimensionality reduction)
- **my_pca = PCA(n_components = n)**

  # (X_Train is feature matrix of training set before DR,
  # X_Train_New is feature matrix of training set after DR):
- **X_Train_new = my_pca. fit_transform(X_Train)**
- **X_Test_new = my_pca. transform(X_Test)**

Algorithm Complexity
(Optional)

Big-O Notation

- **Big-O** notation is used to denote the limit of growth of an algorithm by how they respond to changes in input size.
- E.g. If algorithm processing time grows linearly with the input set *n*, then we say the algorithm is Order n, or O(n).
- The idea behind the big-O notation is to establish an **upper boundary** for the growth of a function f(x) for large x.
- This boundary is specified by a function g(x) that is usually much simpler than f(x).

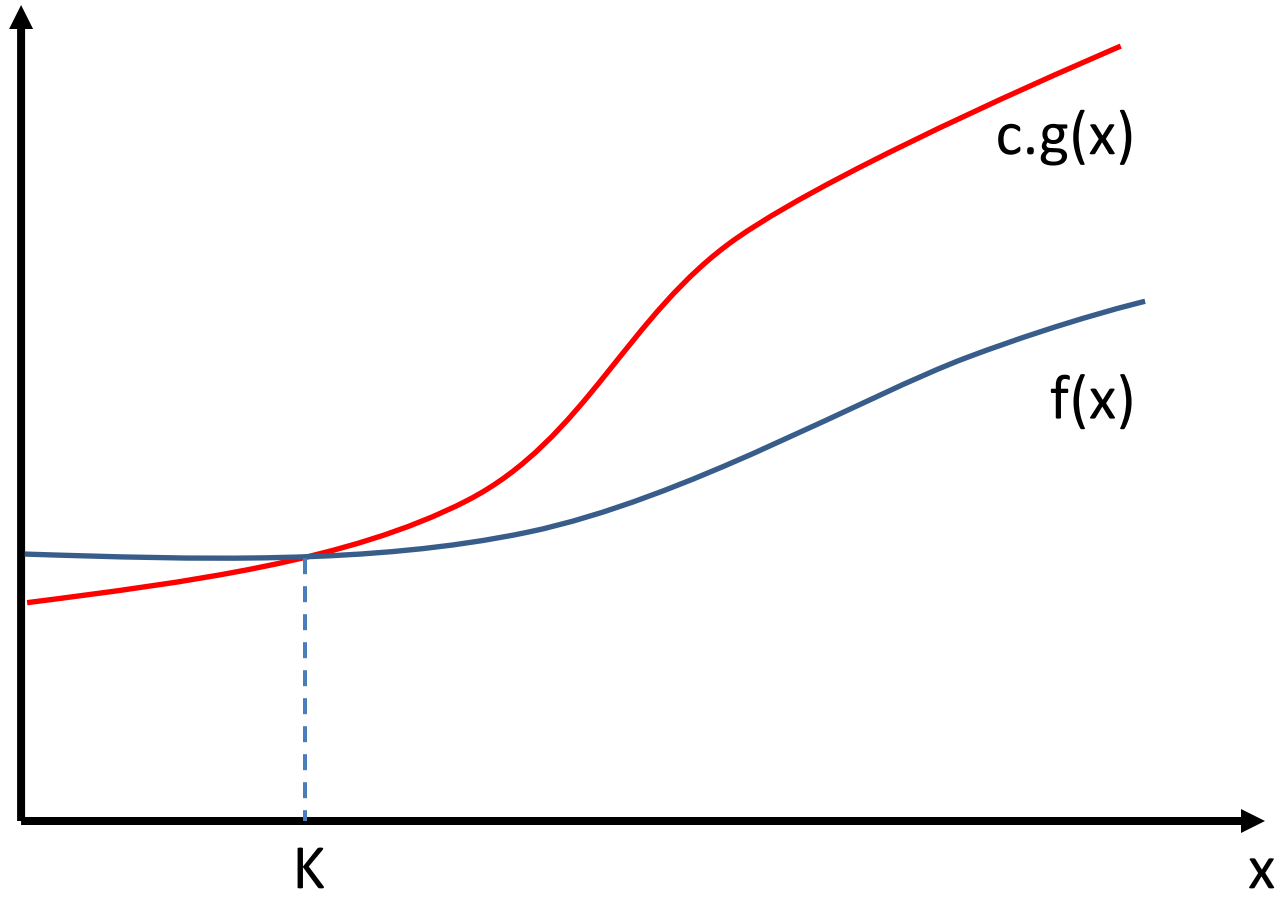
Big-O Notation

- **Definition:** Let *f* and g be functions from the integers or the real numbers to the real numbers.

- We say that **f(x) is O(g(x))** if there are some constants **C** and **k** such that

- **$|f(x)| \leq C|g(x)|$ for every $x > k$.**

Big-O Notation



Big-O Notation

- Note that when we analyze the growth of complexity functions, f(x) and g(x) are always positive.
- Therefore, we can simplify the big-O requirement to:
- **$f(x) \leq C \cdot g(x)$ for every $x > k$.**
- To show that **f(x) is O(g(x))**, we only need to find one pair (C, k) that satisfy the above statement.
- **Important Note:** We are only interested in large x, so it is OK if $f(x) > C \cdot g(x)$ for $x \leq k$.

Example

- Show that $f(x) = 3x+7$ is $O(x)$

- If $7 < x$:

  $3x+7 \leq 3x + x$

  $\Rightarrow 3x+7 \leq 4x = c.g(x)$

  $\Rightarrow$ Thus, $f(x)$ is $O(x)$ for all $x > k$ where $g(x) = x$, $k = 7$, $c = 4$

Example

- Show that $f(x) = x^2 + 10x + 1$ is $O(x^2)$.
- For $x > 1$ we have:

$$x^2 + 10x + 1 \leq x^2 + 10x^2 + 1x^2$$

$$\Rightarrow x^2 + 10x + 1 \leq 12x^2 = c.g(x)$$

Therefore, for $g(x) = x^2$, $C = 12$ and $k = 1$:

$f(x) \leq Cx^2$ whenever $x > k$.

$\Rightarrow f(x)$ is $O(x^2)$.

Function growth rates (for n=1000)

For input size n = 1000		
$O(1)$	1	Constant
$O(\log n)$	≈ 10	Logarithmic
$O(n)$	10^3	Linear
$O(n \log n)$	$\approx 10^4$	
$O(n^2)$	10^6	Quadratic
$O(n^3)$	10^9	
$O(n^4)$	10^{12}	
$O(n^c)$	10^{3*c}	
2^n	$\approx 10^{301}$	
$n!$	$\approx 10^{2568}$	
n^n	10^{3000}	

Other Bound Notations

- **Upper bound**: $T(x) = O(f(x))$ Big-O

  Exist constants c and k such that

  $T(x) \leq c\, f(x)$ for all $x \geq k$

- **Lower bound**: $T(x) = \Omega(g(x))$ Omega

  Exist constants c and k such that

  $T(x) \geq c\, g(x)$ for all $x \geq k$

- **Tight bound**: $T(x) = \Theta(f(x))$ Theta

  When both hold:

  $T(x) = O(f(x))$

  $T(x) = \Omega(f(x))$

Some Frequently Used Algorithms

Algorithm	Time Complexity			Space Complexity
	Best	**Average**	**Worst**	**Worst**
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	**O(n log(n))**	O(n)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Binary Search	Ω(1)	Θ(log(n))	**O(log(n))**	O(1)

Common Data Structure Operations

Data Structure	Time Complexity (Average)			
	Access	Search	Insertion	Deletion
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)
Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)
Hash Table (Dictionary)	-	Θ(1)	Θ(1)	Θ(1)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))

Thank You!

Questions?