Lecture 01: Syllabus, Course Overview and Introduction

CS 0445: Data Structures

Constantinos Costa

http://db.cs.pitt.edu/courses/cs0445/current.term/

Sep 3, 2019, 8:00-9:15
University of Pittsburgh, Pittsburgh, PA

Meta-notes

- These notes are intended for use by students in CS0445 at the University of Pittsburgh. They are provided free of charge and may not be sold in any shape or form.
- These notes are incomplete and NOT a substitute for material covered during course lectures. If you miss a lecture, you should definitely obtain both these notes and notes written by a student who attended the lecture.
- Material from these notes is obtained from various sources, including, but not limited to, the following:
  - Data Structures and Abstractions with Java (5th Edition).Frank M. Carrano and Timothy M.
  - Various online resources (see notes for specifics)

Instructor Info

- Constantinos Costa (costa.c@cs.pitt.edu)
  Office: 5425 Sennott Square

- All other info appears on the class website
  - E.g., office hours, TA contact info/office hours

A note about email

- Prefix all email subjects with [CS0445]
- Address all emails to both the instructor and the TA

Piazza

- Piazza:
  - for all clarifications to lectures, recitations and assignments
- Assignments:
  - To be submitted electronically
  - No Piazza or email clarifications 4 hours prior a deadline

Piazza Guidelines

- Remember that **everything you post** can been seen from everyone.
- Please do not post any assignment code on Piazza, even as a private message. Visit office hours for any code related questions.
- Keep posts on Piazza course-related.
- Please read all questions and responses that are on Piazza before asking a question. Utilize the excellent Piazza search facilities.
- Use a meaningful subject heading.
- Make sure your questions/posts are in the appropriate folder (e.g., hw0, hw1, hw2, ...).
- Tag your post with all the applicable tags.
- Please don't post things to the group that give no useful information.
- Please keep complaints about the course out of the newsgroup. If you have a concern about anything to do with the course, please talk to the instructor.
- **Please be respectful of your peers and others in your posts.**

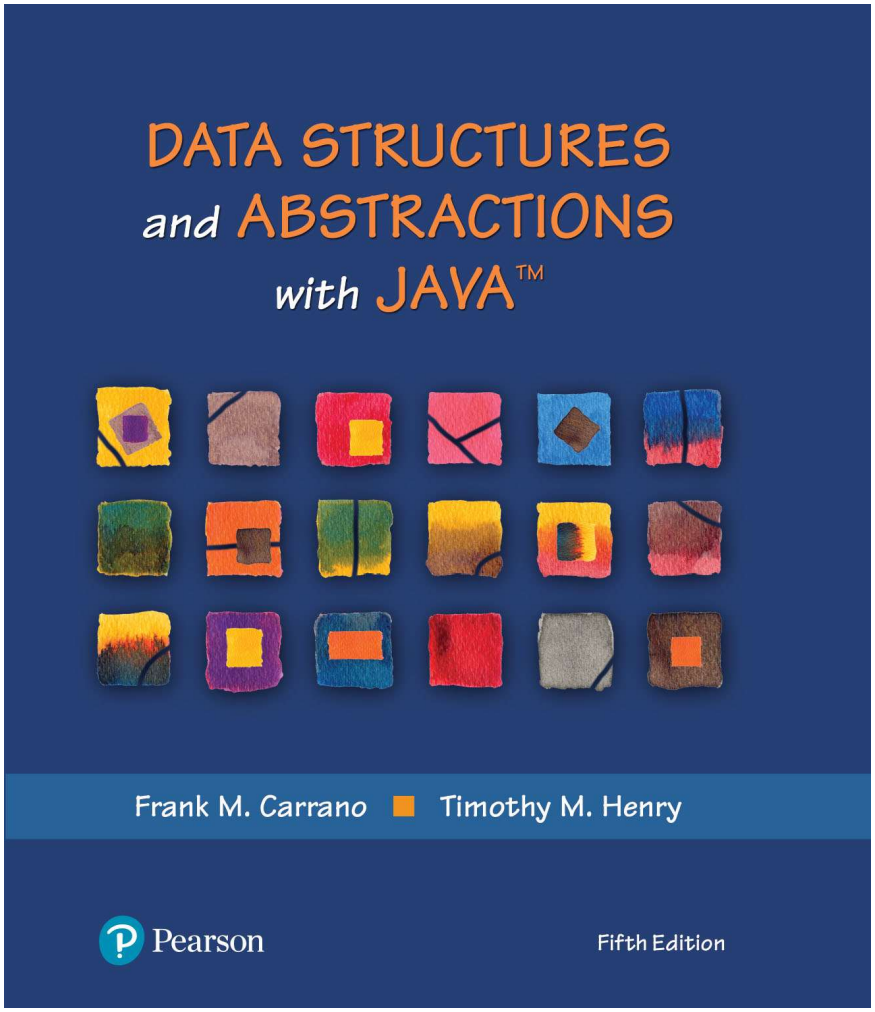
Course Info

- Website:
  - http://db.cs.pitt.edu/courses/cs0445/current.term/
  - Username: cs0445
  - Password: cs0445_fall19_key
- **Review the Course Information and Policies**
- **Assignments will not be accepted after the deadline**
  - No late assignment submissions
    - If you do not submit an assignment by the deadline, you will receive a **0** for that assignment

Assessments - Grading

Assessment	Percentage	Dates
Homeworks	20%	
2 x Quizzes	20%	
Midterm Exam	25%	Oct 22 (tentative)
Final Exam	30%	TBD
Participation (Class & Recitations)	5%	

Data Structures and Abstractions with Java™



- Data Structures and Abstractions with Java (5th Edition).Frank M. Carrano and Timothy M. Henry, Pearson (c) 2016, 5th Edition (ISBN-10: 0-13-483169-1)

Data

- **What is *Data*?**
- Abstract view of reality.
- From the (processed) data the information is derived.
- The data we use **does not always have the same degree of complexity**, e.g., "Employee Details" vs. "Age"
- Some can be decomposed into simpler components, while others may not. Data that cannot be decomposed is called **primitive data types (π.χ. int, char)**
- Problem data is usually not an amorphous data collection. Most of the time they can be expressed by known mathematical structures, e.g.
  - as a simple set (discrete elements): {2, 3, 5, 7, 11, 13}
  - as a vector (ordered elements): (2, 3, 5, 7, 11, 13)
  - as an array: (2D vector) $\begin{pmatrix} 2 & 3 & 5 \\ 7 & 11 & 13 \end{pmatrix}$

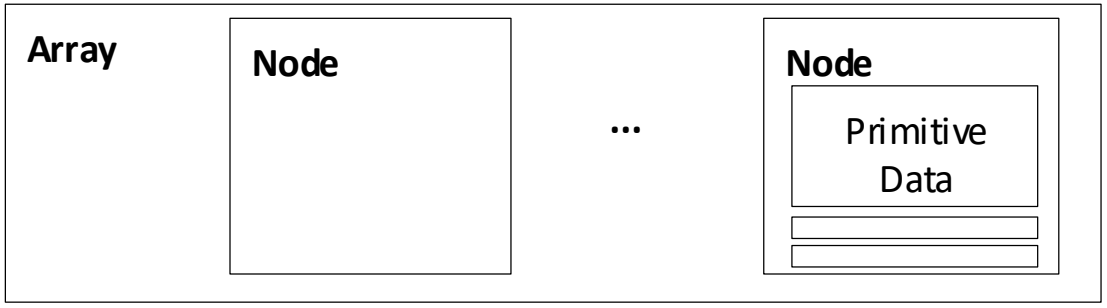
So we need the right structures to organize the data into memory.

Data

- ***Data structures*** are collections of primitive data, which are combined to form more complex data.

- We will use the term ***array*** to describe a collection of elements that we will call ***nodes***.

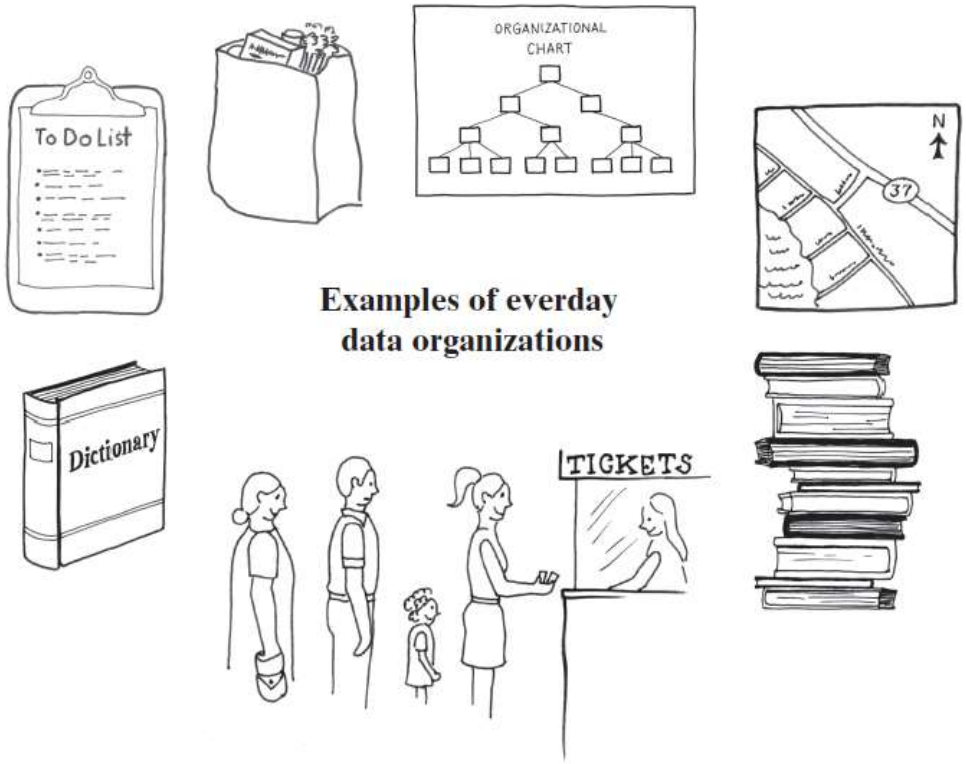
- A node can be ***simple***, that is, it consists of a single primitive data, or it is ***complex***, in which it consists of two or more ***fields***. The fields of a node can represent primitive data of various types.




```
public class Node{
    char sex; // 8 bits
    int age; // 32 bits
    boolean owns_car; // 1 bit
    int idcard; // 32 bits
};
```

Data Organization in Life

- Standing in a line
- Stack of books
- To-Do list
- Dictionary
- Folders, directories c
- Road map



Computer Data Organization

- Abstract Data Type: ADT
- Data Structure
- Collection
- Examples of containers

  - **Bag**
  - **List**
  - **Stack**
  - **Queue**
  - **Dictionary**
  - **Tree**
  - **Graph**

32/64-bit Architectures

- A few words about data types with x86 και x64 architectures:
  - **I**(ntegers) L(ong) P(ointer) 32 => x86 Model
  - L(ong) P(ointer) 64 => x64 Model

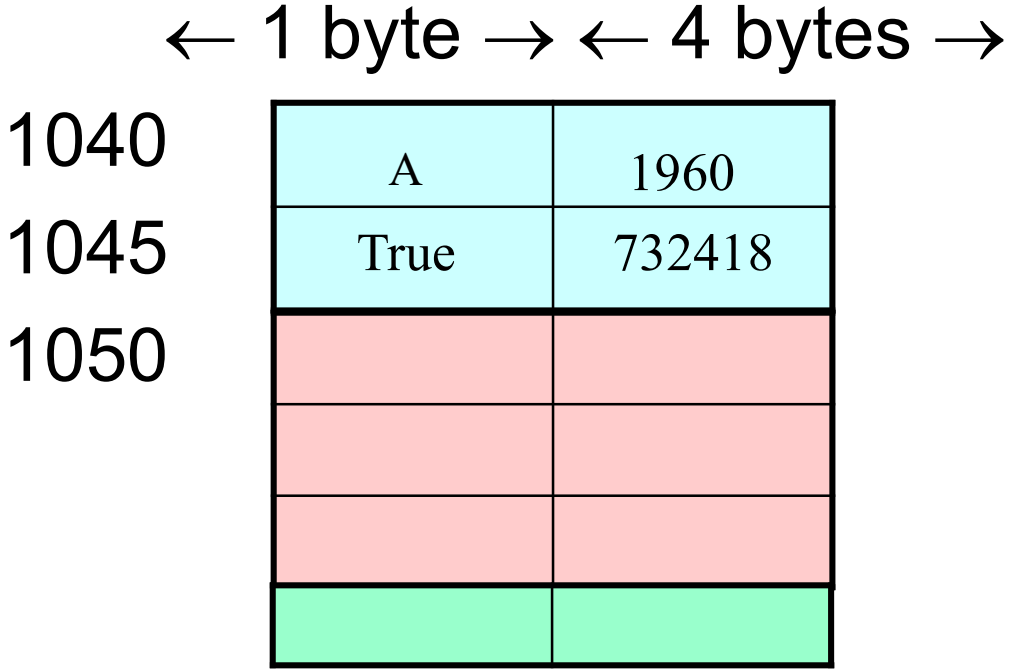
Datatype	**ILP32 Model**	**LP64 Model**
char	8	8
short	16	16
int	32	32
long	**32 (4 bytes)**	**64 (8 bytes)**
pointer	**32 (4 bytes)**	**64 (8 bytes)**

The memory can ONLY have up to 2^32 = ~4x10^9 (δηλ., 4GB) addresses ☹!

The memory can have up to 16 Exa (x10^{18}) addresses☺!

Data

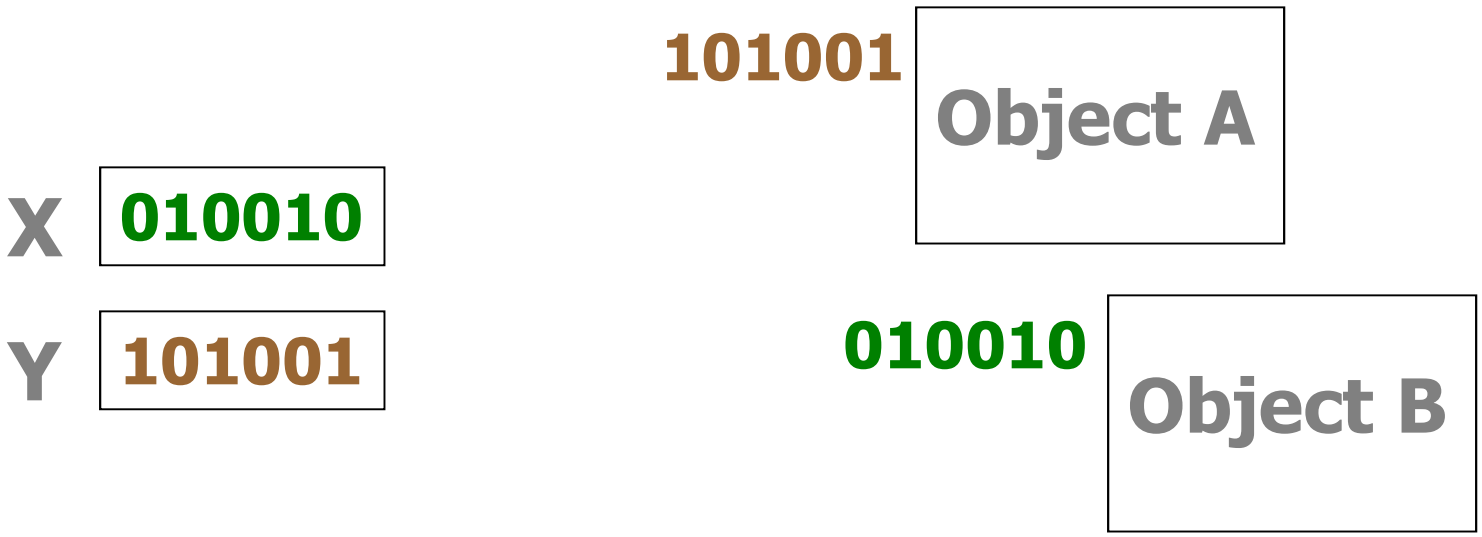
- A field is characterized by its ***address*** and its ***length***. The address of a field is the address of its first memory cell and its length is the number of cells it consists of.



The address of the first field, the first entry, and the whole table is 1040. The address of the second entry 1050, and so on.

References, Pointers and Memory

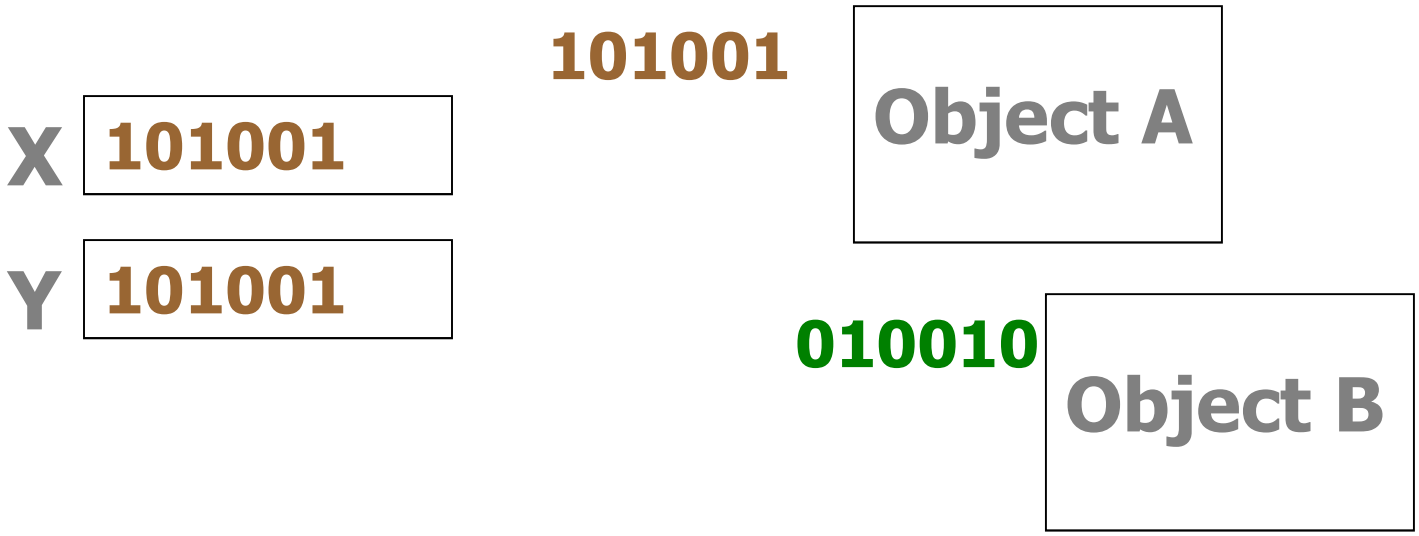
- How do references and pointers relate?
  - Many programming languages (ex: C, C++, Pascal) use pointer variables
    - Pointers are variables that store addresses of other memory locations
    - Pointers allow indirect access of the data in objects



References, Pointers and Memory

- So the value stored in a pointer is an address
- However, if you dereference a pointer, you gain access to the object it "points to"

```
X = Y;  // Changes what X points to
        // X no longer has access to
        // Object B
```



Algorithms

What is an algorithm?

Abu Jafar Mohammed ibn Musa Al-Khowarizmi (790-840)

An **algorithm** is a finite sequence of commands, strictly defined and executable in a finite (measurable) time, which, if followed, achieves a desired result.

Necessary criteria:

- There is input and output
- Defined commands (not ambiguities).
- Deterministic (to accomplish the goal).

Performance: not necessary but desirable

That is, given a **problem**, an algorithm provides the **instructions** where the **data** is **transformed** and **combined** to **solve the problem**.

Algorithms

- When executing an algorithm, the structure of the data plays a very important role.

- Equation Wirth

  Algorithms + Data = Programs

The main objective of the course is to study data structures, their representation in the memory of a computer, and algorithms that create and edit them.

Selection problem

- **Problem: Suppose we have n numbers and want to determine the k-th largest.**

  e.g. Let the numbers {5, 72, 3, 4, 1, 9, 65}
  (72,65,9,5,4,3,1)

  The 2^{nd} larger is 65, the 6^{th} larger is 3, …

- This problem is known as **the selection problem.** There are several 'easy' ways to solve it:

  1. **Sort-Based:** We 'read' n numbers in a list, a. We sort the list from largest to smallest based on some sorting algorithm. Return the (k-1) th of the list, i.e. a [k-1].

Selection problem

2. **Buffer-based:** We ‘read' the first k numbers in a list a. We sort the list from largest to smallest so for k = 3 we have a = {72,5,3}. Next, we edit the remaining n-k numbers as follows:
   if an element is smaller than a [k-1] we ignore it, otherwise we place it in the correct position in the list. When this procedure is complete, we return the k-th element of the list, i.e. a [k-1].

- The methodology for evaluating the performance of individual solutions will be the subject of this course.

Algorithms

- An important conclusion is that writing **a correct program is not enough**. In particular, when the initial set of data is large, **the execution time** of a program should be first class citizen.

- In this lesson we will learn to i) **calculate the execution time of algorithms** and ii) compare the efficiency of different algorithms **before implementing them**. We will also study methods to improve program speed.