http://algs4.cs.princeton.edu

1.2 Abstract Data Types

- *Author* Fulvio Corno
- *Edited by* Tolga Çapın

Introduction

We know C/Java primitive data types: int, float, double, char…

We need to write programs at a higher level of abstraction

Abstract Data Type

ADTs store data and allow various operations on the data to access and change it

ADTs are mathematical models

ADTs are useful when designing complex algorithms

ADTs are not classes (well, not quite)

Abstract Data Types

ADTs are "abstract" because they specify the operations of the data structure and leave *implementation details* to later

More similar to "abstract classes" or "interfaces" (C++ supports abstract classes, Java supports interfaces).

Note: not all implementation details can be deferred.

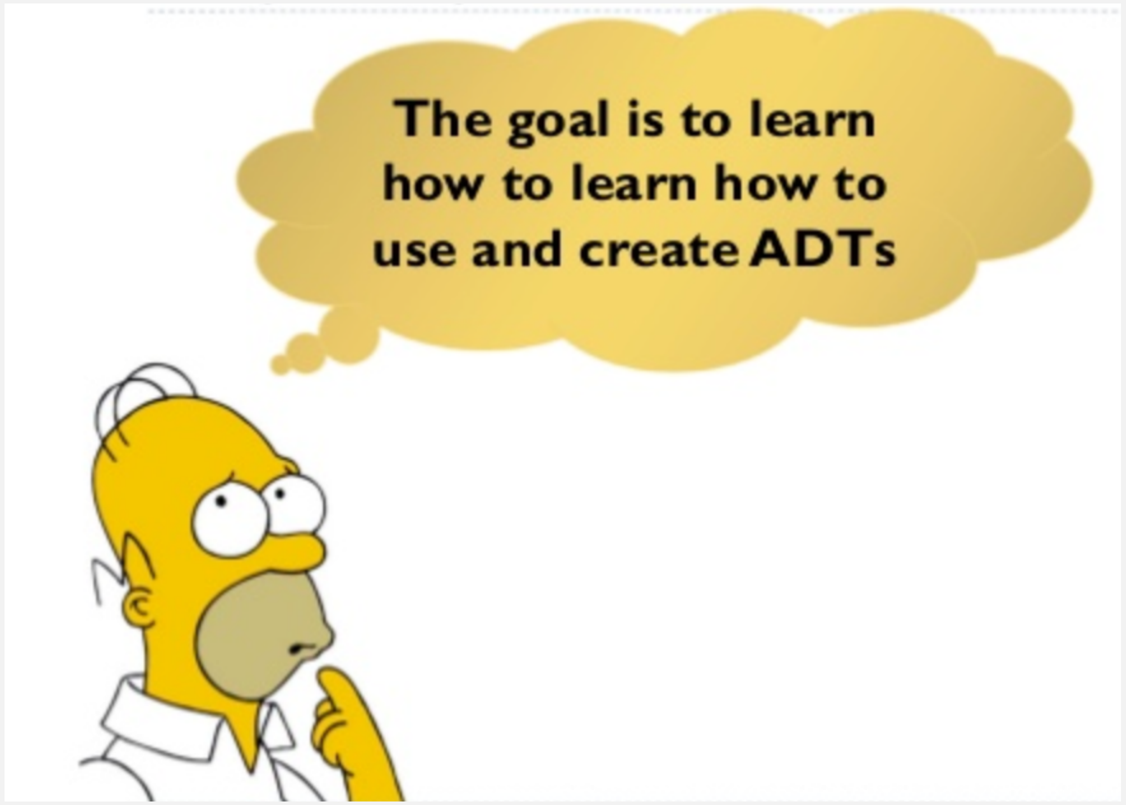
Why study ADTs?

“Get your data structures correct first, and the rest of the program will write itself.”

David S. Johnson

(winner of Knuth’s Prize in 2010)

Why study ADTs?


The goal is to learn how to learn how to use and create ADTs

Built-in ADTs

High level languages often provide built-in ADTs.

For example:

Standard Template Library (STL) (C++)

Java Collections Framework (Java)

Common Properties

Most ADTs provide a way to

- add an item
- remove an item
- find, retrieve, or access an item

Most Collection ADTs provide more possibilities

- check whether the collection is empty
- make the collection empty
- give me a subset of the collection
- …

A very simple ADT: Santa’s stack

Santa's sack operations

insertToy(toy)

- Insert a toy in the sack
- Duplicates are — obviously — allowed

extractToy(toy)

- Remove the given toy from the sack

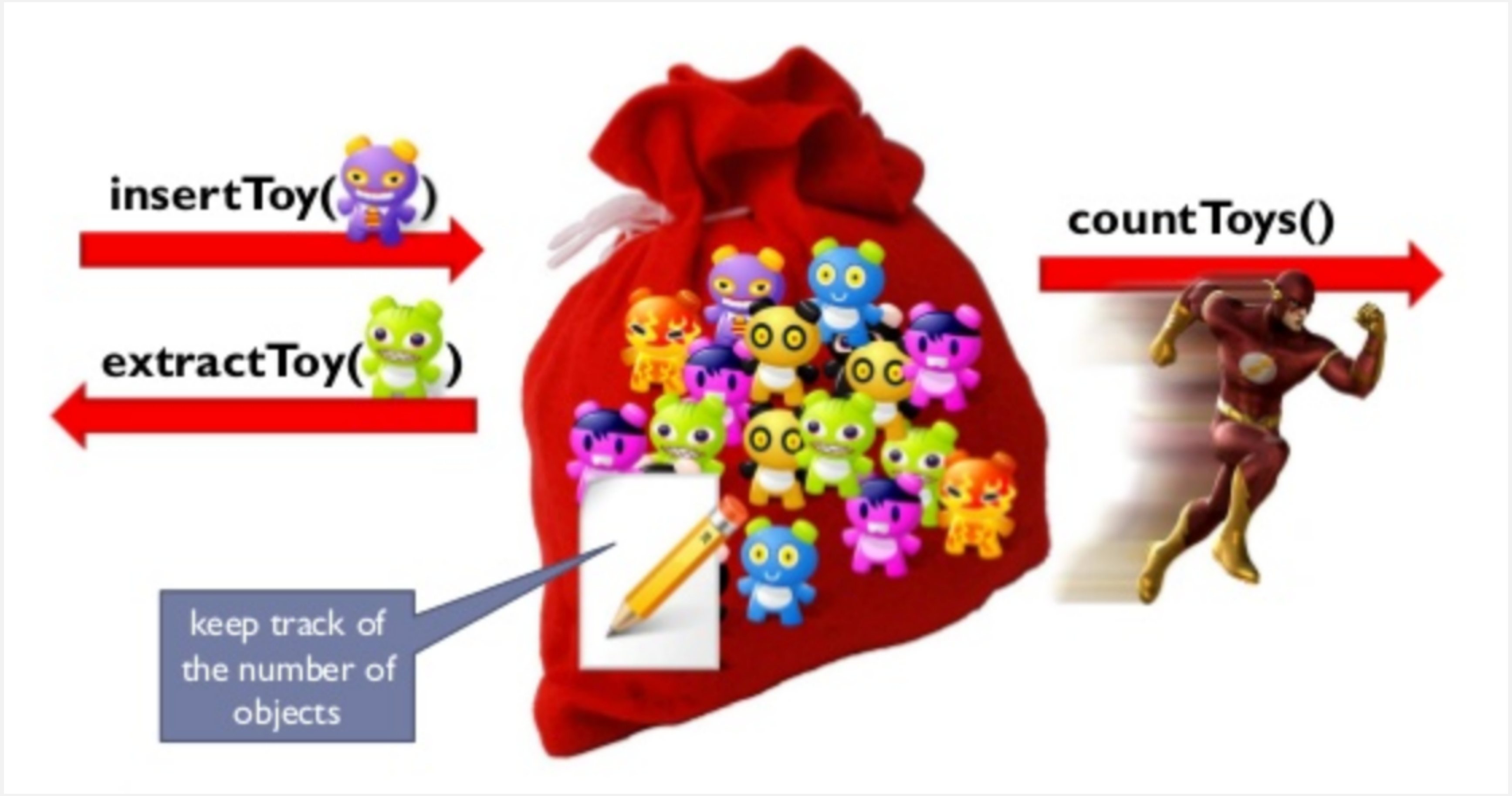
countToys()

- Count how many toys are actually stored in the sack

Santa's sack (inefficient version)

insertToy( )
extractToy( )
countToys()

Santa's sack (more efficient version)



The lesson

ADTs do not specify ***the details*** of the implementation.

BUT

It is essential to choose the right ADT for the algorithms.

Complexity is a major issue for large problems.