# Review of Chapters 3-6

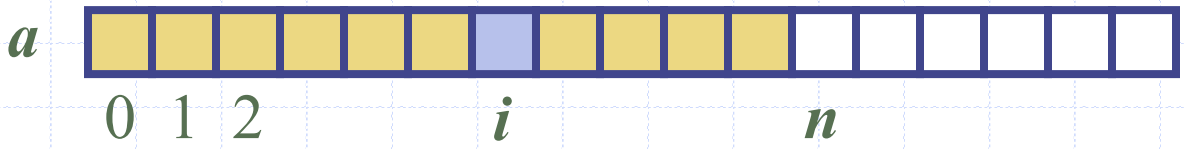
Feb. 14, 2018

# Chapter 3 Fundamental Data Structures

- Arrays
- Linked lists
  - Singly linked lists
  - Doubly linked lists
  - Circularly Linked List

# Arrays

- What is array?
- Capacity of array
- Array operations (access, add, delete elements) and efficiency.
- When to use array?



# Sorting an Array

- The Insertion-Sort Algorithm

**Algorithm** InsertionSort($A$):

***Input:*** An array $A$ of $n$ comparable elements

***Output:*** The array $A$ with elements rearranged in nondecreasing order

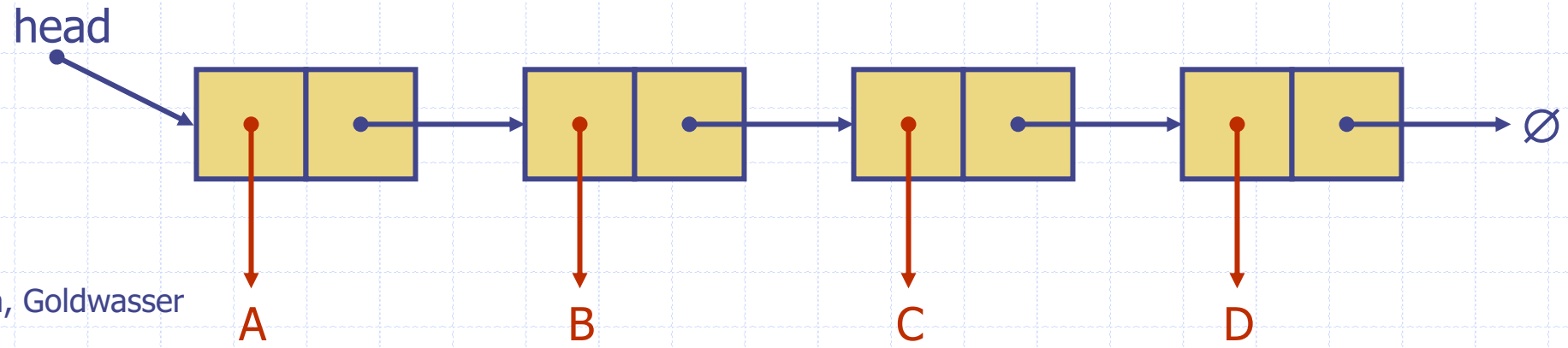
**for** $k$ from 1 to $n-1$ **do**

Insert $A[k]$ at its proper location within $A[0], A[1], \ldots, A[k]$.

**Code Fragment 3.5:** High-level description of the insertion-sort algorithm.

# Singly Linked Lists

- What is singly linked list?
- Capacity of a singly linked list
- Linked list operations and efficiency.
  - Find an element in the list
  - Inserting/removing at the Head / tail
- When to use singly linked list?



# Other Linked Lists

- What is doubly linked list ?
- Circularly Linked List

CS 2233-01 Data Structures and Algorithms

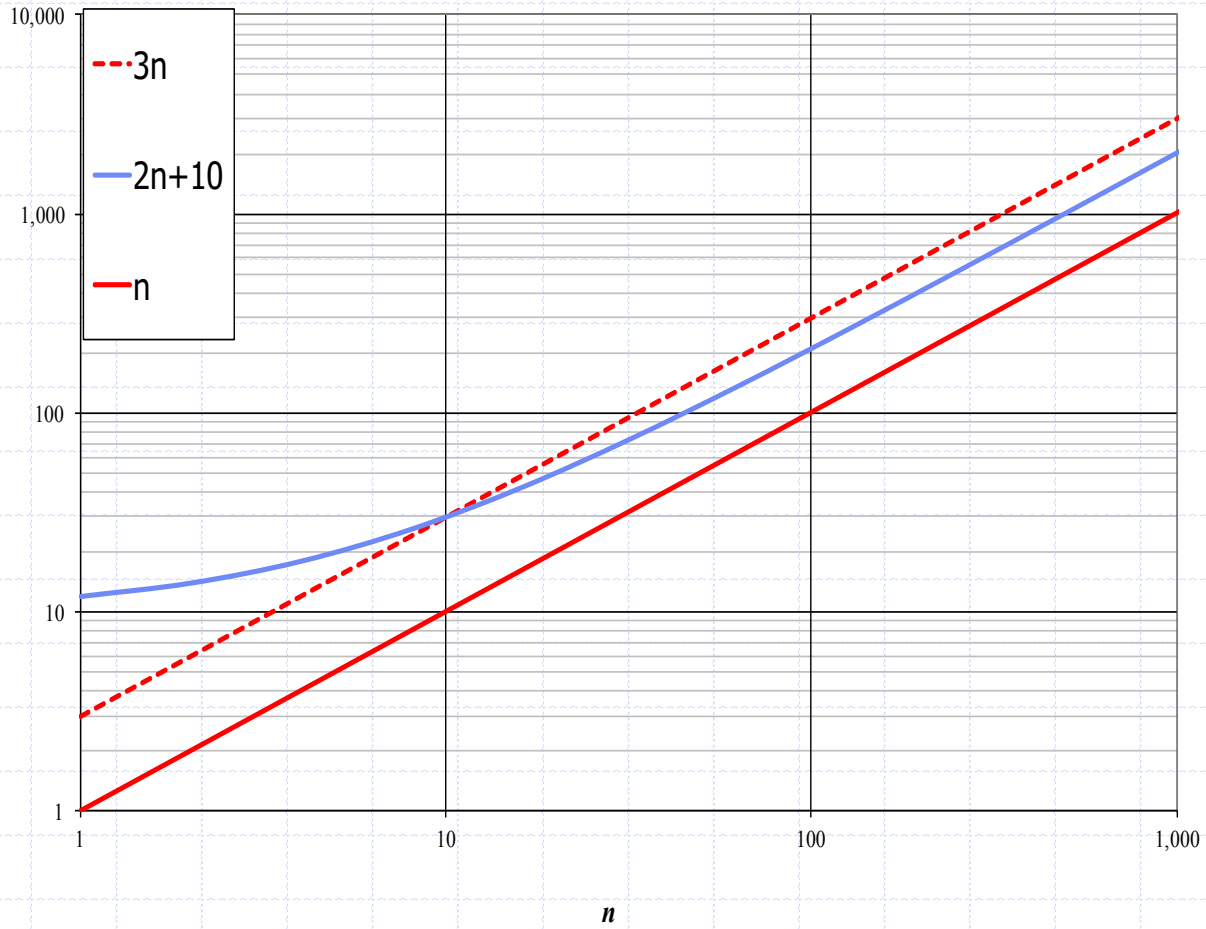
# Chapter 4 Algorithm Analysis

# Asymptotic Notation (Big-Oh Notation)

- **Algorithm analysis:** determine how running time increases as the *size* of the problem increases.

- **Big Oh-notation:** Given functions $f(n)$ and $g(n)$, we say that $f(n)$ is $O(g(n))$ if there are positive constants $c$ and $n_0$ such that

  $f(n) \leq cg(n)$ for $n \geq n_0$



# Basic asymptotic efficiency classes

| | |
|---|---|
| 1 | constant |
| $\log n$ | logarithmic |
| $n$ | linear |
| $n \log n$ | $n$-log-$n$ |
| $n^2$ | quadratic |
| $n^3$ | cubic |
| $2^n$ | exponential |
| $n!$ | factorial |

# Big-Oh Rules

- If $f(n)$ is a polynomial of degree $d$, then $f(n)$ is $O(n^d)$, i.e.,
  1. Drop lower-order terms
  2. Drop constant factors, e.g. $n^4 + 100n^2 + 10n + 50$
- Use the smallest possible class of functions
  - Say "$2n$ is $O(n)$" instead of "$2n$ is $O(n^2)$"
- Use the simplest expression of the class
  - Say "$3n + 5$ is $O(n)$" instead of "$3n + 5$ is $O(3n)$"

# Relatives of Big-Oh

## big-Omega

- f(n) is $\Omega(g(n))$ if there is a constant $c > 0$ and an integer constant $n_0 \geq 1$ such that

$$f(n) \geq c\, g(n) \text{ for } n \geq n_0$$

## big-Theta

- f(n) is $\Theta(g(n))$ if there are constants $c_1 > 0$ and $c_2 > 0$ and an integer constant $n_0 \geq 1$ such that

$$c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for } n \geq n_0$$

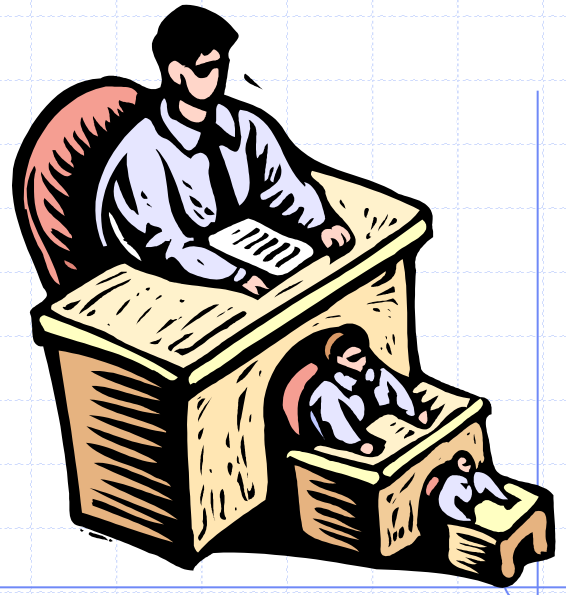
# Questions in Mid-term exam

- Compare different efficiency classes
- Estimate running time of a piece of code (in big-Oh term)
- Choose data structures/algorithms to solve a problem, under constraints of time or space

CS 2233-01

# Data Structures and Algorithms

## Chapter 5 Recursion



# Content of a Recursive Method

- Base case(s)
  - Values of the input variables for which we perform no recursive calls are called base cases (there should be at least one base case).
  - Every possible chain of recursive calls must eventually reach a base case.
- Recursive calls
  - Calls to the current method.
  - Each recursive call should be defined so that it makes progress towards a base case.

# A Classic Recursive Algorithm: Binary Search

- A Classic Recursive Algorithm: ***Binary search***
- What is **Linear Recursion** (Binary Recursion, Multiple Recursion)?
- What is **Tail Recursion**?
- Convert Tail recursion to non-recursive methods?

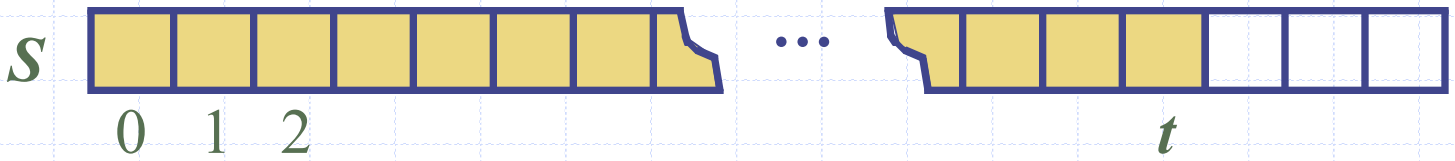
# Chapter 6 Stacks, Queues, and Deques

- Presentation for use with the textbook Data Structures and Algorithms in Java, 6th edition, by M. T. Goodrich, R. Tamassia, and M. H. Goldwasser, Wiley, 2014

# Stacks

- Definition & ADT
- Array-based stacks
- Performance and Limitations of Array-based Stacks
- Linked list-based stacks
- Applications of stacks

```
public interface Stack<E> {
  int size();
  boolean isEmpty();
  E top();
  void push(E element);
  E pop();
}
```



# Applications of stacks

- Text parsing
- Parentheses matching
- Evaluating Arithmetic Expressions
- Evaluate postfix notation

# Queues

- Definition & ADT
- Queues V.S. Stacks
- A Circular Queue

```
public interface Queue<E> {
  int size();
  boolean isEmpty();
  E first();
  void enqueue(E e);
  E dequeue();
}
```

# Deque (Double-Ended Queues) ADT

addFirst($e$): Insert a new element $e$ at the front of the deque.

addLast($e$): Insert a new element $e$ at the back of the deque.

removeFirst( ): Remove and return the first element of the deque (or null if the deque is empty).

removeLast( ): Remove and return the last element of the deque (or null if the deque is empty).

# Deque ADT (cont.)

first(): Returns the first element of the deque, without removing it (or null if the deque is empty).

last(): Returns the last element of the deque, without removing it (or null if the deque is empty).

size(): Returns the number of elements in the deque.

isEmpty(): Returns a boolean indicating whether the deque is empty.

# Implementation

- Circular Array
- Doubly Linked List
- Performance of the Deque Operations

| Method | Running Time |
|---|---|
| size, isEmpty | $O(1)$ |
| first, last | $O(1)$ |
| addFirst, addLast | $O(1)$ |
| removeFirst, removeLast | $O(1)$ |

# Midterm will be on Friday, Feb. 16.