UNIVERSITY OF CALGARY

CPSC 319
Data Structures, Algorithms, and Their Applications

Winter 2024

PA-2

- Read from a file:
  java.io.FileReader
  java.io.BufferedReader
- Using linked lists:
  java.util.LinkedList
- Write to a file:
  java.io.FileWriter
  java.io.BufferedWriter
- String manipulation:
  StringBuilder
- String comparison:
  firstString.compareTo(secondString)
  https://docs.oracle.com/javase/8/docs/api/java/lang/String.html#:~:text=equals(Object)-,compareTo,-public%C2%A0int%C2%A0compareTo

Array, Linked List, and Dynamic Array

- Definition:
  - Array:

    Contiguous block of memory elements with the same data type.
    Fixed size, allocated during declaration.

  - Linked List:

    Collection of nodes where each node points to the next node in the sequence.
    Dynamic in size, memory allocated as needed.

  - Dynamic Array:

    Resizable array that grows or shrinks in size as needed.
    Memory management is handled behind the scenes.

Array, Linked List, and Dynamic Array

- Access Time:
  - Array:

    Constant time O(1) for accessing elements using index.

  - Linked List:

    Linear time O(n) for accessing elements, as traversal is required.

  - Dynamic Array:

    Constant time O(1) for accessing elements using index.

Array, Linked List, and Dynamic Array

- Insertion/Deletion:
  - Array:

    Inefficient for insertions/deletions as it may require shifting elements.

    Time complexity for insertions/deletions is O(n).

  - Linked List:

    Efficient for insertions/deletions anywhere in the list (constant time).

    Memory allocation for new nodes can be a concern.

  - Dynamic Array:

    Efficient for append operations, but insertions/deletions may still incur occasional resizing overhead (amortized O(1)).

Array, Linked List, and Dynamic Array

- Memory Efficiency:
  - Array:

    More memory efficient as it uses a single block of memory.

  - Linked List:

    Less memory efficient due to additional pointers between nodes.

  - Dynamic Array:

    Balances memory efficiency with flexibility by resizing as necessary.

Array, Linked List, and Dynamic Array

- Usage:
  - Array:

    Ideal for situations where random access and fixed size are crucial.

  - Linked List:

    Suitable for scenarios with frequent insertions and deletions.

  - Dynamic Array:

    Suitable for scenarios with a variable and unpredictable number of elements.

Sample Question 3:

- Write a Java code that gets a positive integer n and then gets an array of n integers, a_0 a_1 … a_{n-1}. Print out the number of pairs, i and j, such that i <= j and min(a_i, a_{i+1}, …, a_j) = min(a_0, a_1, …, a_{n-1}).

- Test case:

  Input 1:
  5
  7 -2 3 0 5

  Output 1:
  8

  Input 2:
  13
  1 2 3 0 1 2 3 0 1 2 3 0 1

  Output 2:
  72