CS5000: Foundations of Programming

Migon Kang, PhD
Computer Science, Kennesaw State University
Linked Lists

- Linked lists and arrays are similar since they both store collections of data.
- An array allocates memory for all its elements together as one block of memory.
- In contrast, a linked list allocates space for each element separately in its own block of memory called a "linked list element" or "node".
- The best and simplest example of a dynamic data structure that uses pointers for its implementation.
Linked Lists

- A set of dynamically allocated nodes
- The pointers points to the next member of the list
Linked Lists

- Traversing a Linked List
  - no “random” access
Linked Lists

- Adding a node in the specific position
Deleting a node in the specific position

```
DATA
NEXT POINTER

DATA
NEXT POINTER

DATA
NEXT POINTER

NULL
```
Linked Lists

- Class Node

  - Two components: an item and a reference for a next node

```java
public class Node {
    private Object item;
    private Node next;
    ...
}
public class LinkedList {
    private Node head;
    ...
}
```
Linked Lists

- Memory Allocation

```
DATA NEXT POINTER -> NULL
DATA NEXT POINTER -> DATA NEXT POINTER -> NULL
DATA NEXT POINTER -> DATA NEXT POINTER -> DATA NEXT POINTER -> NULL
```
Linked Lists

- Class LinkedList<E>
  - Check the manual

- Check LinkedListTest.java
Advantages

- Linked lists act like an array, but can grow and shrink as needed.
  - Items can be added or removed from the middle of the list
  - There is no need to define an initial size
  - Memory efficiency
Disadvantages

- There is no “random” access
- Have a much larger overhead over arrays
  - Items are dynamically allocated
  - Each item must store an additional reference
Stacks

- Ordered collection data, with two principal operations
  - Push: adds an element to the top of the collection
  - Pop: removes the last element from the collection
- LIFO (Last In, First Out)
Stacks

push()

pop()
Operations

- Initialize Stack
- Push
- Pop
- Status
  - IsEmpty(), IsFull(), getItemSize()
- Clear Stack
Implementations

- Arrays
  - Fixed Size

- Linked List
  - Dynamically increase/shrink stack size
Array representation

- Array of the data type
- For a static array, a current index of the top of the stack
Implementation with a Linked List

- The header of the linked list points to the top of the stack

- Push(): inserts an element to the front of the list
- Pop(): removes an element from the front of the list
Class Stack

- Class Stack in Java

- Check “StackTest.java”
Examples of Stacks

- Evaluation of Arithmetic Expression
- Infix to Postfix Conversion
Examples 1

- Evaluation of Arithmetic Expression
  - Infix and Postfix Expression
    - Infix expression
      - precedence and associativity of operators
      - Parenthesis
    - Postfix expression
      - No Parentheses
  - Examples:
    - Infix: \(4 * (2 + 4) - 3\)
    - Postfix: 424+*3-
Evaluation of Arithmetic Expression

- Rule for postfix expression
  - If an operand: push it on the stack
  - If an operator: pop 2 operands and perform the computation, and push the result
Queues

- Ordered collection data
- FIFO: First-In-First-Out
  - A new element is added at the end, the existing elements are deleted from the front (the other end)
Operations

- Enqueue: inserts an element in the queue
- Dequeue: delete an element from the queue
- IsEmpty: reports if the queue is empty or not
- IsFull: report if the queue is full or not
- Initialize: create a new empty queue
- Destroy: delete all contents of the queue
## A queue in an array

- The head index is not necessarily at index 0
- Circular array → Two indices for Head and Tail

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A queue in an array

```
5 8
8
8 7
8 7 3
8 7 3 1
```

Dequeue()
Dequeue()
Enqueue(7)
Enqueue(3)
Enqueue(1)
A queue in an array

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>8</th>
<th>7</th>
<th>3</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Enqueue(2)
Enqueue(10)

What are indices of Head and Tail for IsFull and IsEmpty?
A queue in a linked list

- Three variables:
  - pHead: a base pointer
  - pTail: a pointer of the tail node.
  - nNumItem: # of items

- Add an item into the tail of the list
- Remove an item from the front of the list
A queue in a linked list

Queue

pFront

pTail

nNumItems

Diagram showing a queue in a linked list with pointers pFront and pTail, and a variable nNumItems.
Queue Applications

- Waiting in line in real life
- Keyboard buffer in Computer Science
Hash Tables

- Mapping keys to values
- Components
  - Keys
  - Hash Function
  - Buckets
Motivation

- **Associative Array**
  - A collection of (Key, Value)

- **Examples**
  - **Student Grades (Name, Grade)**
    - ("John", 90), ("Tom", 92), ("Joe", 85)
    - `GradeOf("John")` returns 90
  - **Years when a war started (Year, War)**
    - (1776, "Revolutionary"), (1861, "Civil War"), (1939, "WW2")
    - `WarStarted(1861)` returns "Civil War"
    - `WarStarted(1990)` returns NULL
Motivation

- What if using Arrays? (for large volume of data)
  - Maximum size must be known in advance
  - The key can be a index of the array
    - The key might not be a number
    - Most of the array elements would be empty
Motivation

- Array of a structure
  - A structure that stores key and value
    - Adding elements: Add to the end
    - Removing elements: Find the element moving next elements after the element
    - Finding: search from the beginning of the array
Hash Tables

- Hash Table
  - Fixed-size array: each element points at a linked list
Hash Tables

- **Hash Function**
  - Mapping each key to an array index
  - E.g., for an integer key $h$
    - Hash function: $i = h \% \text{TABLESIZE}$
    - HashTable[$i$] is a head of a linked list
Array of size 5 with hash function “h % 5”

- “1776 % 5” is 1
- “1861 % 5” is 1
- “1939 % 5” is 4
Hash Table in Java

- Class Hashtable<K,V>
  - Java.util.*;

- Check "HashTableTest.java"