Chapter 3: Lisr, Stacks, and Queues

Yangtao Ge

June 24, 2019

Abstract

This section discusses about:

- Introduce Abstratc Data Type (ADT)
- how to efficiently perform operations on lists
- stack ADT
- queue ADT

1 Abstract Data Types (ADT)

Definition: *Abstract data type* is a set of objects together with a set of operations

The three data structures (Lists, Stacks, and Queues) are ADT examples

2 The List ADT

Some Feature of List:

- general form is: $A_0, A_1, A_2, ..., A_{N-1}$
- special list of size 0 is **empty list**
- A_i succeeds A_{i-1} & A_{i-1} precedes A_i

- First element is A_0 & Last element is A_{N-1}
- position of element A_i is i

2.1 Simple Array Implementation of Lists

Using plain array: Only use array accesses (i.e. *findkth* operation)

Ref: pp.58 - 59

2.2 simple Linked Lists

feature of linked list:

- consists of a series of nodes
- each node contains the '*element*' and '*next* link'
- the last cell's 'last link' is *null*

Some avaliable method definition:

- findkth: scan through the list and find the element on that position
- find: find the position that the element we specify
- remove: method can be executed in one *next* reference change
- insert: requires obtaining a new node from the system by using a *new* call and executing two reference maneuvers

When removing the last element: $\underline{\text{tricky}}$ $\rightarrow$ using double linked list

3 List in the Java Collection API

3.1 *Collection* Interface

Some feature of *Collection* Interface:

- package in *java.util*
- collection extends the *Iterable* Interface
- can use enhance for loop
- method in Collection Interface:

```
public interface Collection<AnyType> extends Iterable<AnyType>{
    int size();
    boolean isEmpty();
    void clear();
    boolean contains(AnyType x);
    boolean add(AnyType x);
    boolean remove(AnyType x);
    java.util.Iterator<AnyType> iterator();
}
```