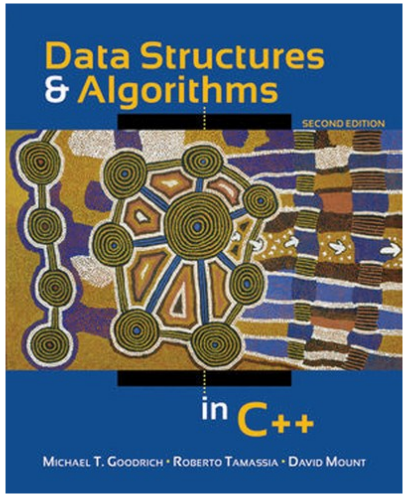
Data Structure

Linked List

Shin Hong

7 Mar 2023


Data Structures & Algorithms
SECOND EDITION
in C++
Michael T. Goodrich · Roberto Tamassia · David Mount

DS&A. Chapter 3

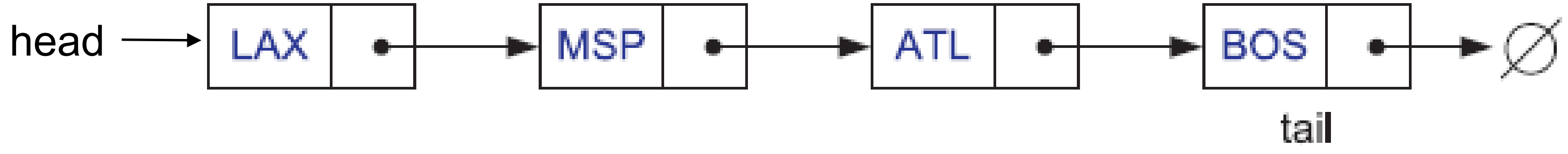
Foundation of Computer Science http://infolab.stanford.edu/~ullman/focs.html
- Ch. 6. The List Data Model

Motivation

- allocate new memory space on demand
- together with the space for an element, allocate a memory space for storing a pointer
    - store the pointer to the (i+1)-th node in the i-th node

Singly Linked List

- A linked list is a collection of nodes that form a linear ordering
  - allocate a memory space for each element together with a pointer
    - a node is a pair of element and next pointer
  - the next pointer inside a node is a link to the next node, or null when the node is terminal

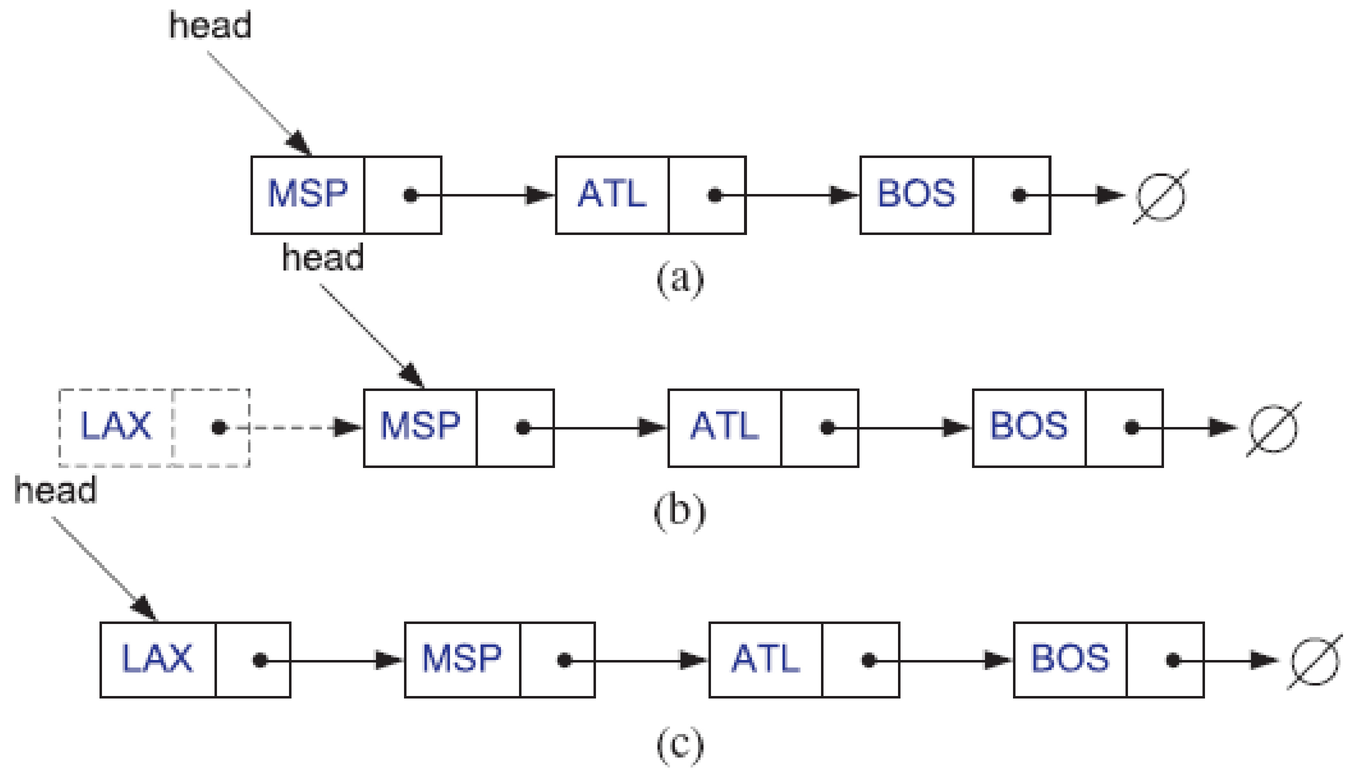


Linked List Structure

- Node
  - a pair of a data element and a Node pointer
- Linked List
  - head: a Node pointer to the first Node object
  - tail: a Node pointer to the last Node object
    - optional

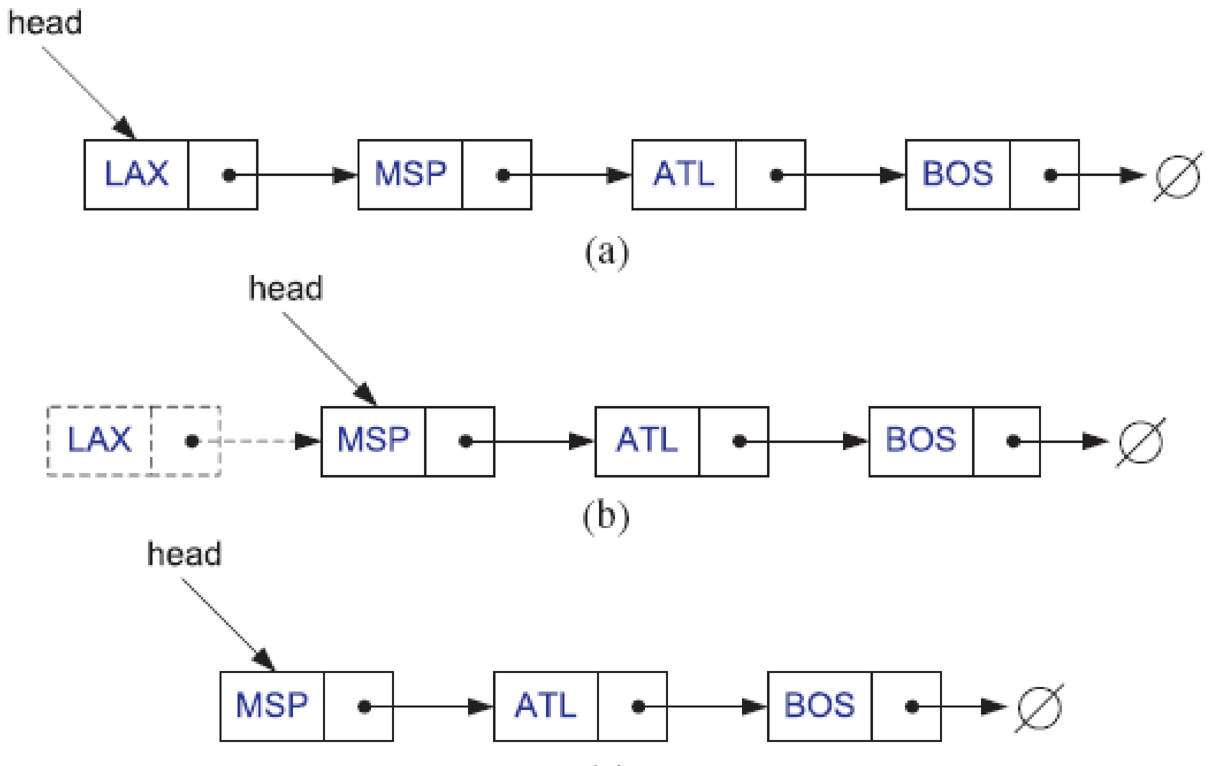
Insertion to the List Front

- Create a new node, and then set its next link to point to the current head node



Removal from the List Front

- Save the pointer of the current head node
- Update the head node as the next pointer of the current head node



(a)

(b)

(c)

Array list vs. Linked list

- Time performance
  - insertion
  - removal
  - element access
  - search

- Memory performance

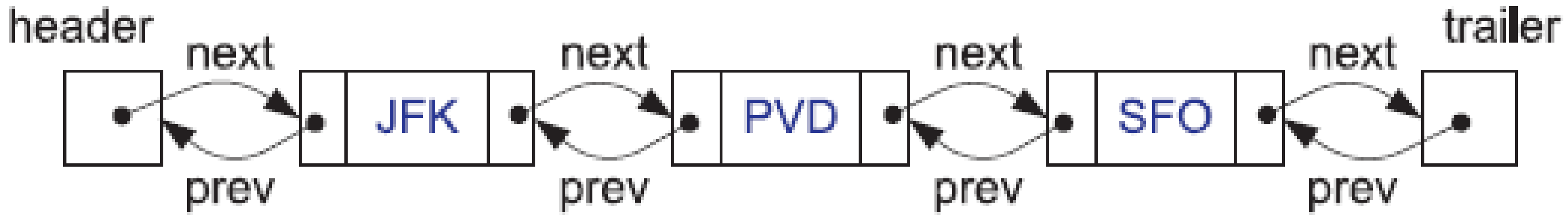
Doubly Linked Lists

- Limitation of singly linked list
  - making a quick update while iterating a list is not easy for there is no access to a predecessor from the current node

- Doubly linked lists
  - a node stores two pointers, a next link and a prev link
  - allow a great variety of quick insertion, removal, and other updates

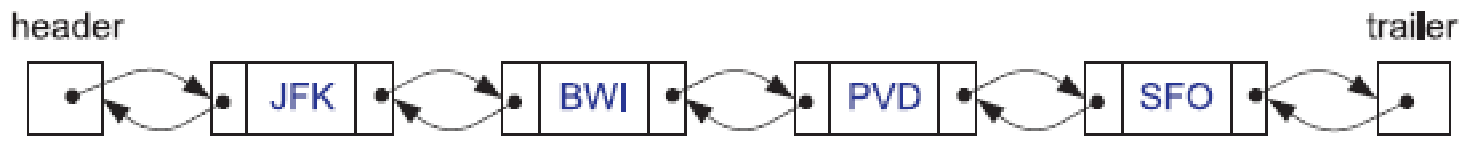
Doubly Linked List - Structure

- Header and sentinel nodes
  - dummies for indicating the front and the end of a list

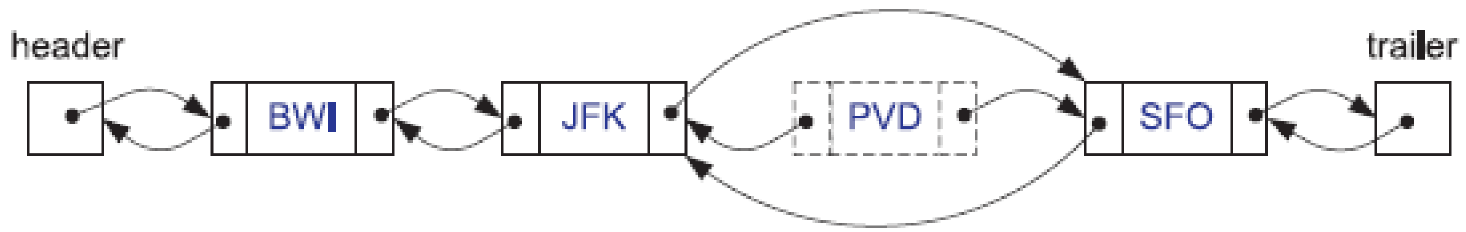


Doubly Linked List - Removal

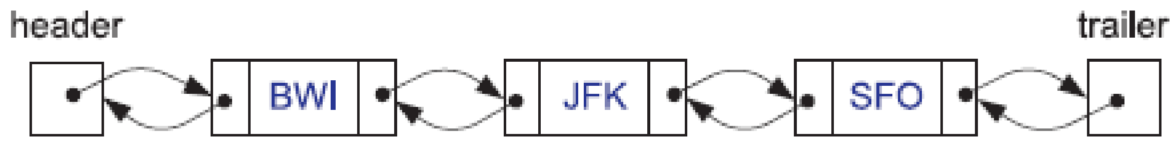
- Remove a node c
  - `c->prev->next = c->next`
  - `c->next->prev = c->prev`
  - `delete c`



(a)

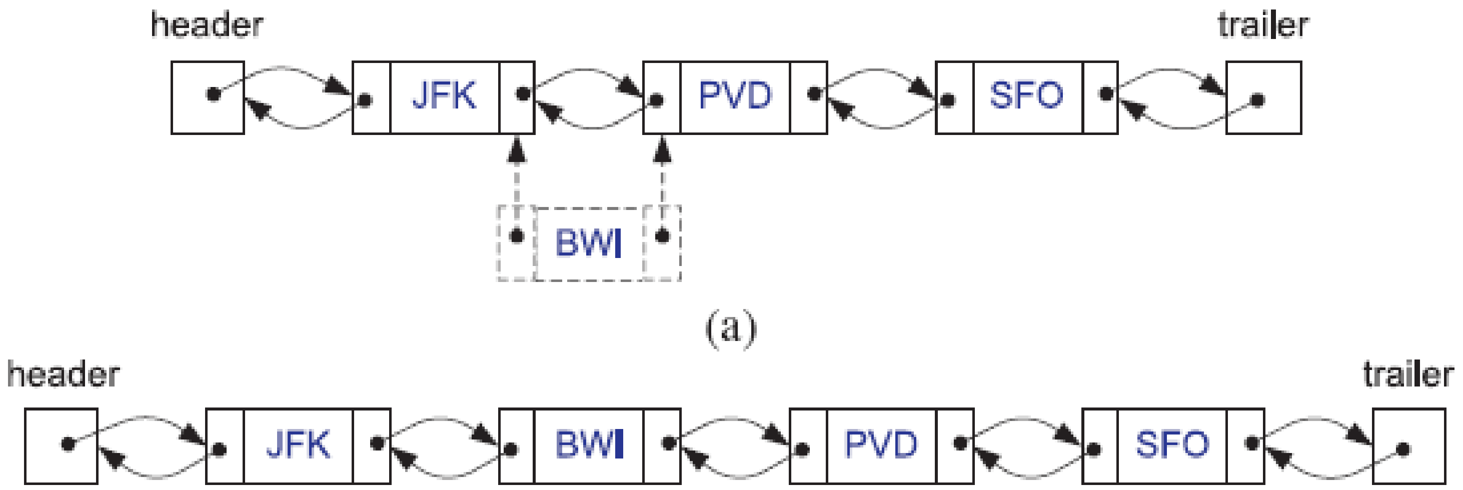


(b)



Doubly Linked List - Insertion

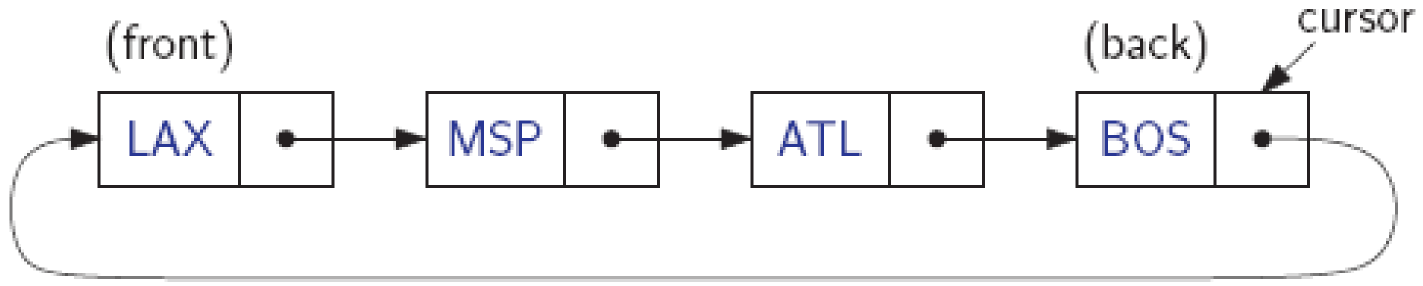
- Insert a new node n immediately after the current node c
  - `n->prev = c`
  - `n->next = c->next`
  - `c->next->prev = n`
  - `c->next = n`



Circularly Linked List

- Extend a singly linked list to form a cycle
  - following next pointers, we can visit all nodes and cycle back to the starting node

- Structure
  - cursor: a pointer of the current node
  - front: the node currently pointed by cursor
  - back: the node immediately following the front node


(front)
(back)
cursor
LAX
MSP
ATL
BOS