Linked List

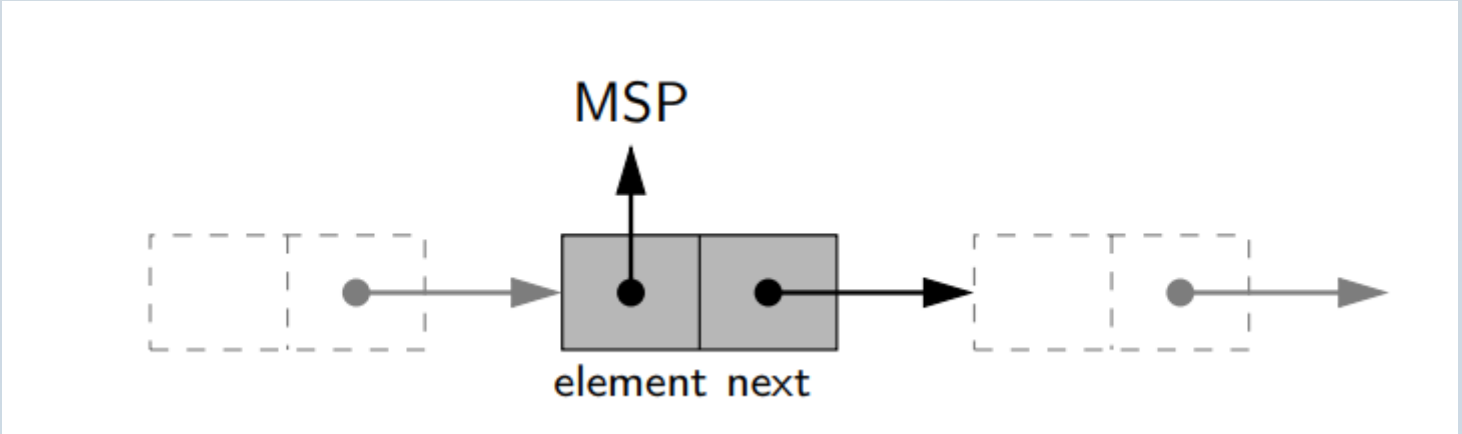
TUAN NGUYEN

Outline

- Linked List
- Double Linked List

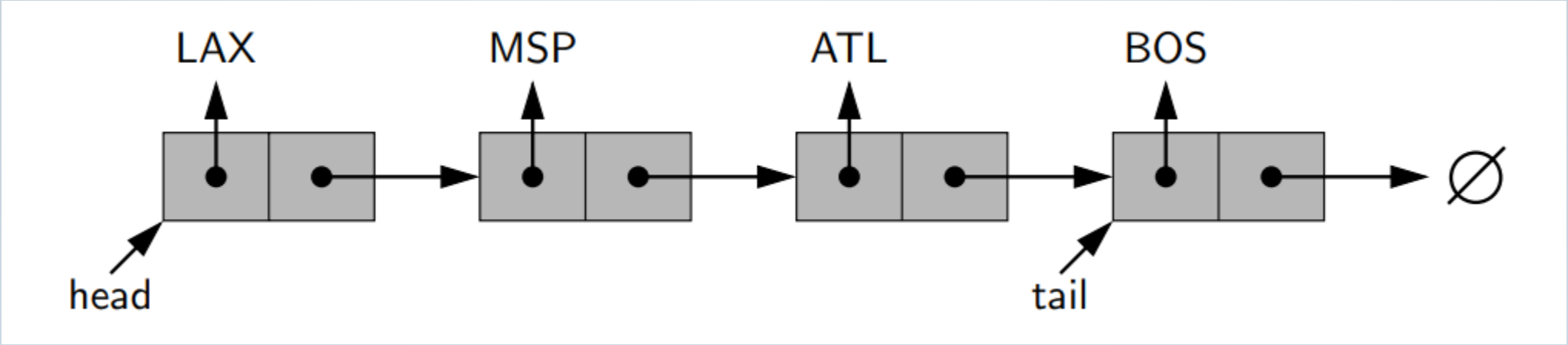
Singly Linked Lists

- A singly linked list is a collection of nodes that collectively form a linear sequence.
- Each node stores a reference to an object that is an element of the sequence, as well as a reference to the next node of the list.

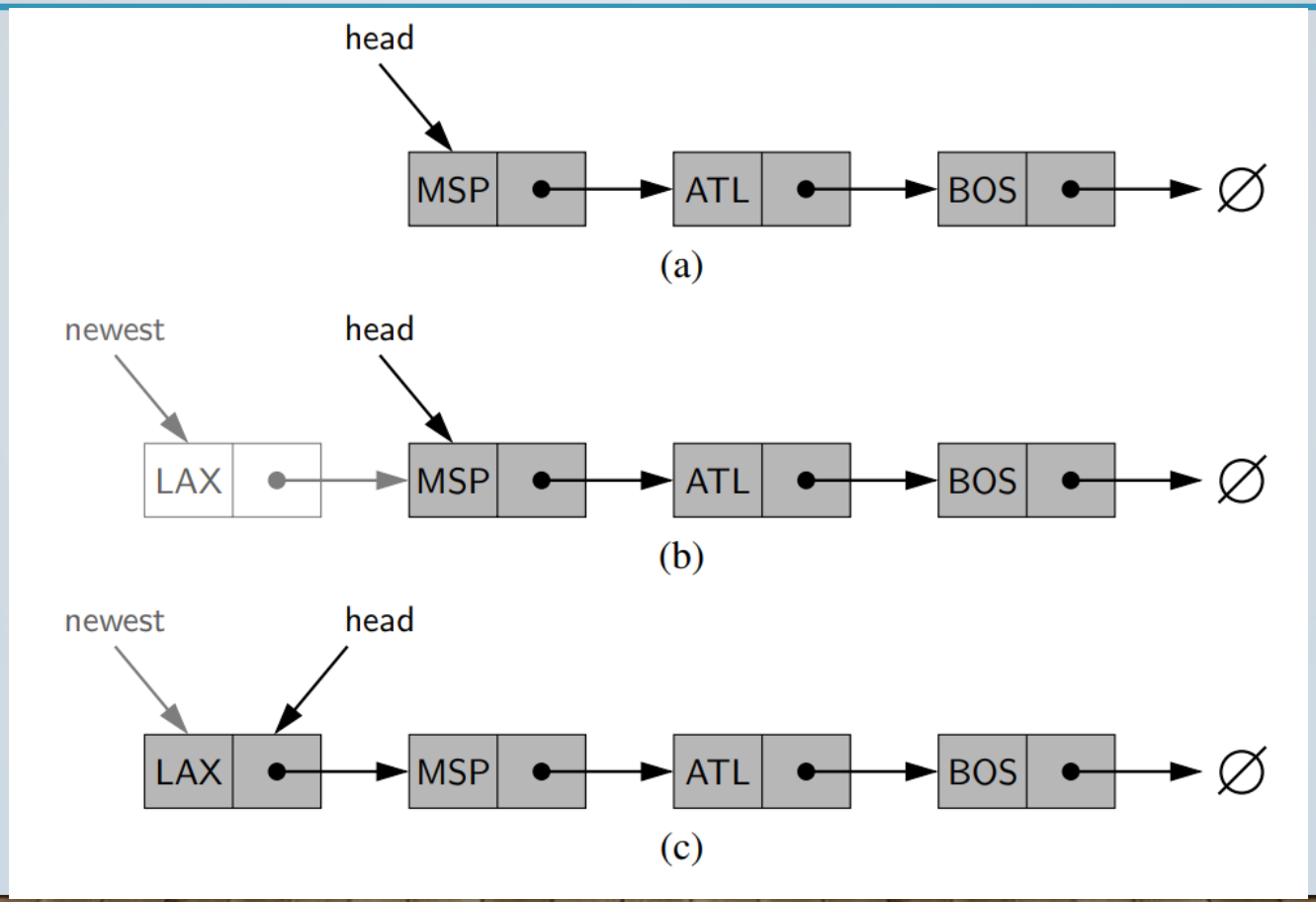


Singly Linked List

- The first and last node of a linked list are known as the head and tail of the list, respectively.
- The tail as the node having None as its next reference.



Inserting an Element at the Head

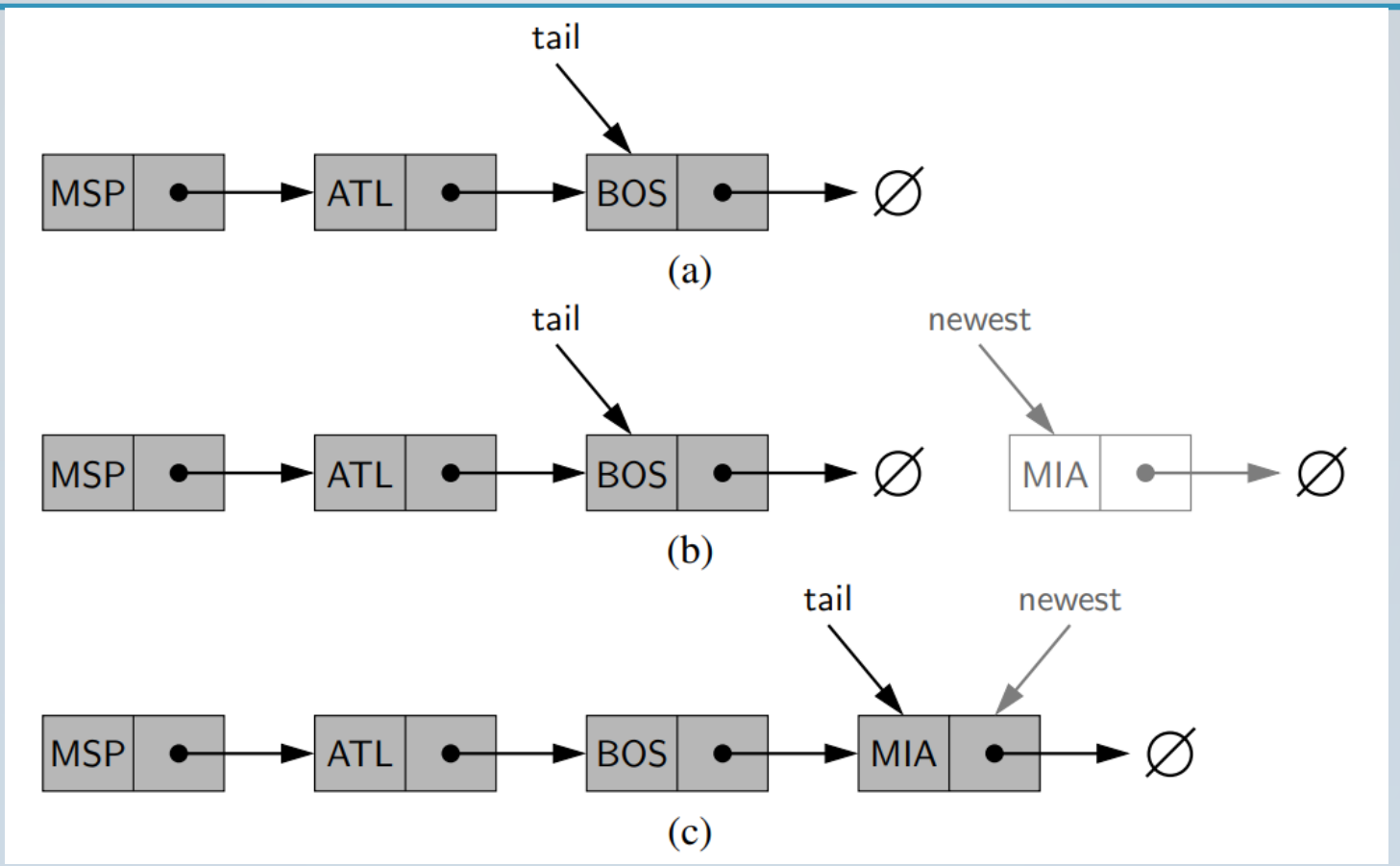


Inserting an Element at the Head (1)

Algorithm add_first(L, e):

```
newest = Node(e)  {create new node instance storing reference to element e}
newest.next = L.head  {set new node's next to reference the old head node}
L.head = newest  {set variable head to reference the new node}
L.size = L.size + 1  {increment the node count}
```

Inserting an Element at the Tail

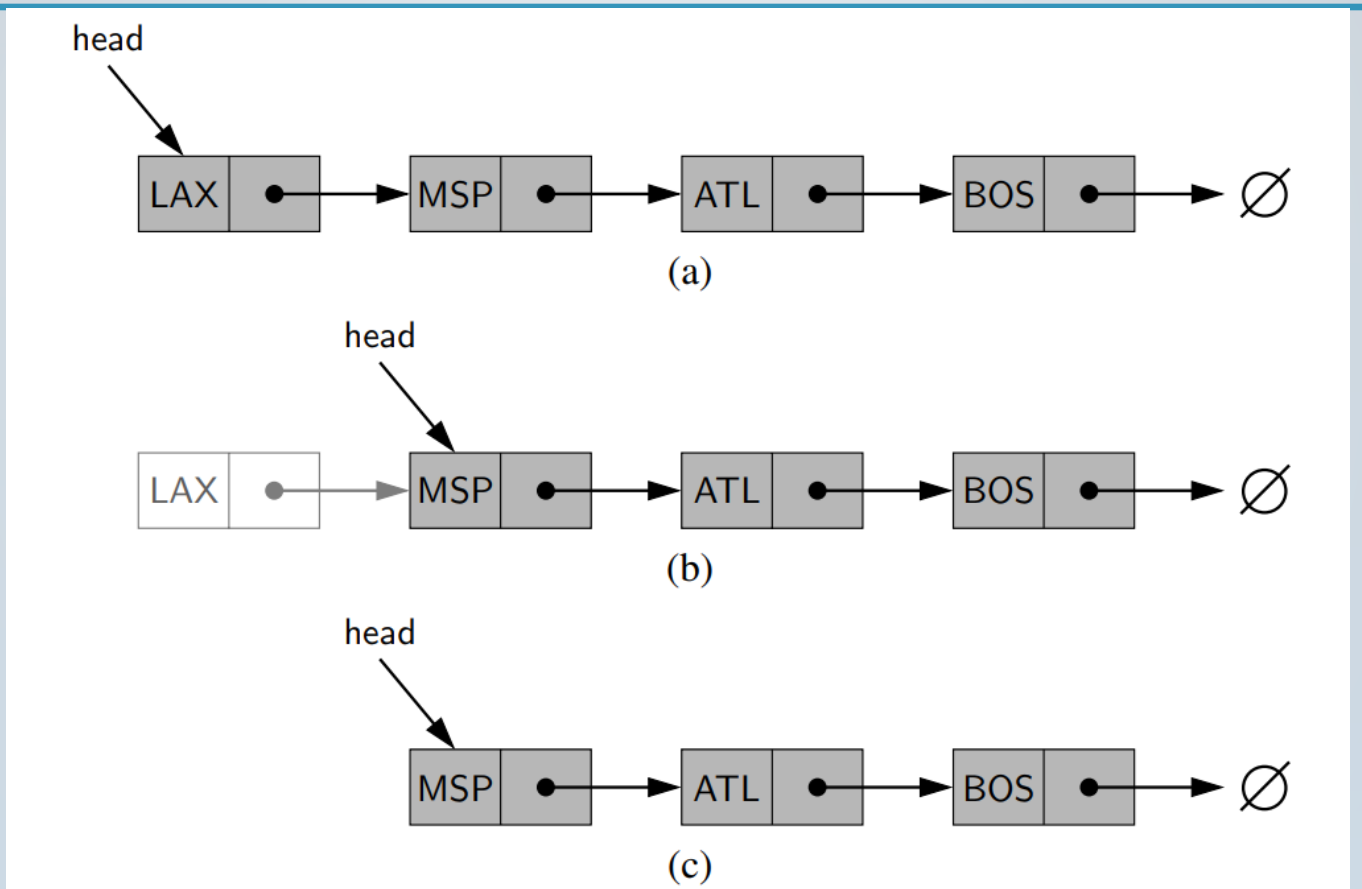


Inserting an Element at the Tail (1)

Algorithm add_last(L, e):

```
newest = Node(e)   {create new node instance storing reference to element e}
newest.next = None   {set new node's next to reference the None object}
L.tail.next = newest   {make old tail node point to new node}
L.tail = newest   {set variable tail to reference the new node}
L.size = L.size + 1   {increment the node count}
```

Removing an Element



Removing an Element (1)

```
Algorithm remove_first(L):
    if L.head is None then
        Indicate an error: the list is empty.
    L.head = L.head.next          {make head point to next node (or None)}
    L.size = L.size − 1                          {decrement the node count}
```

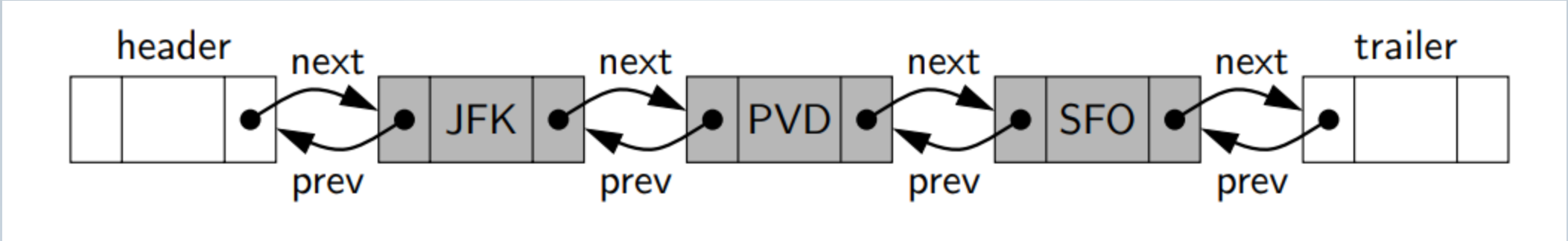
Running time

Operation	Running Time
S.push(e)	$O(1)$
S.pop( )	$O(1)$
S.top( )	$O(1)$
len(S)	$O(1)$
S.is_empty( )	$O(1)$

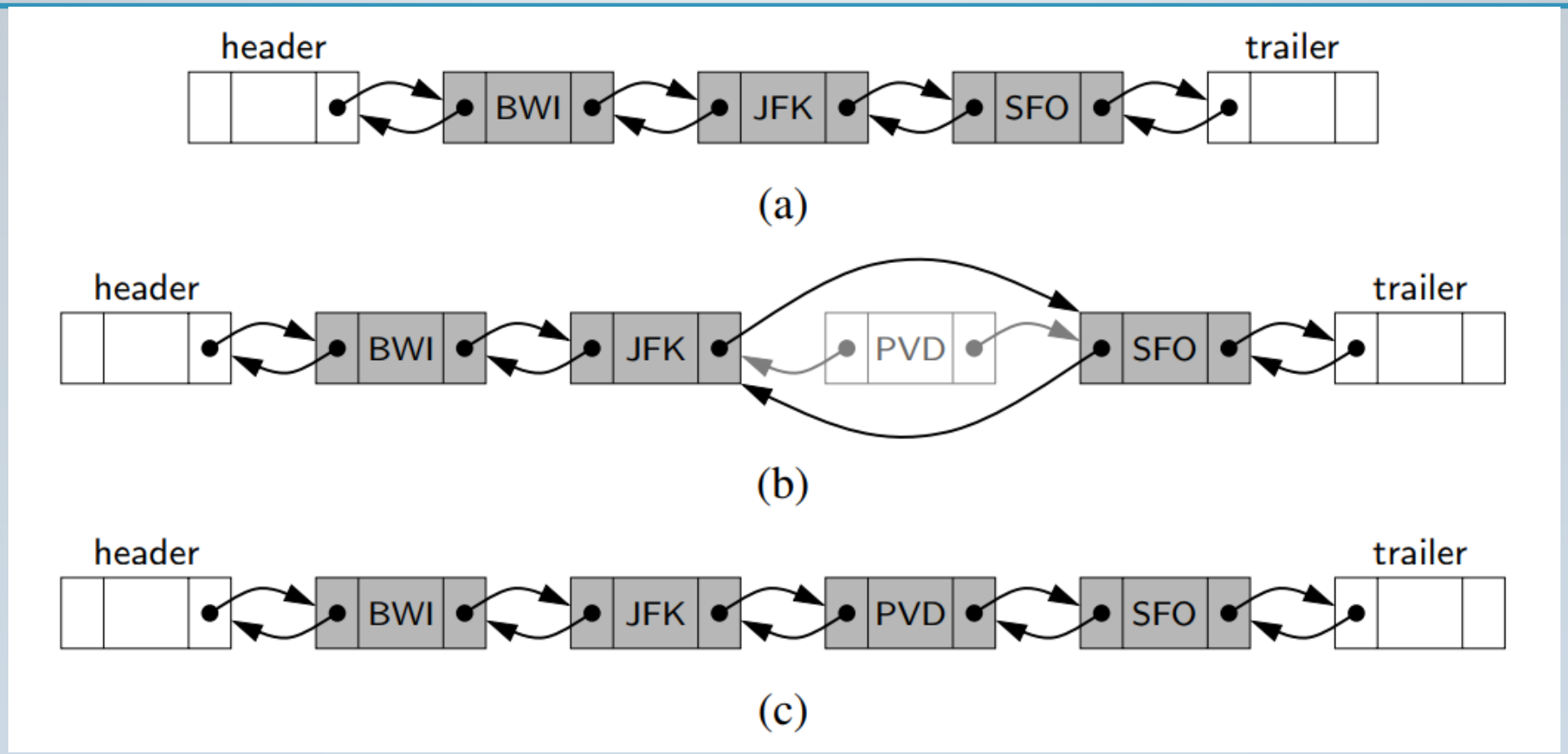
Array vs Linked list

Advantage	Disadvantage
• Size of linked lists is not fixed, they can expand and shrink during run time. • Insertion and Deletion Operations are fast and easier in Linked Lists. • Memory allocation is done during run-time (no need to allocate any fixed memory).	• Memory consumption is more in Linked Lists when compared to arrays. Because each node contains a pointer in linked list and it requires extra memory. • Elements cannot be accessed at random in linked lists. • Traversing from reverse is not possible in singly linked lists.

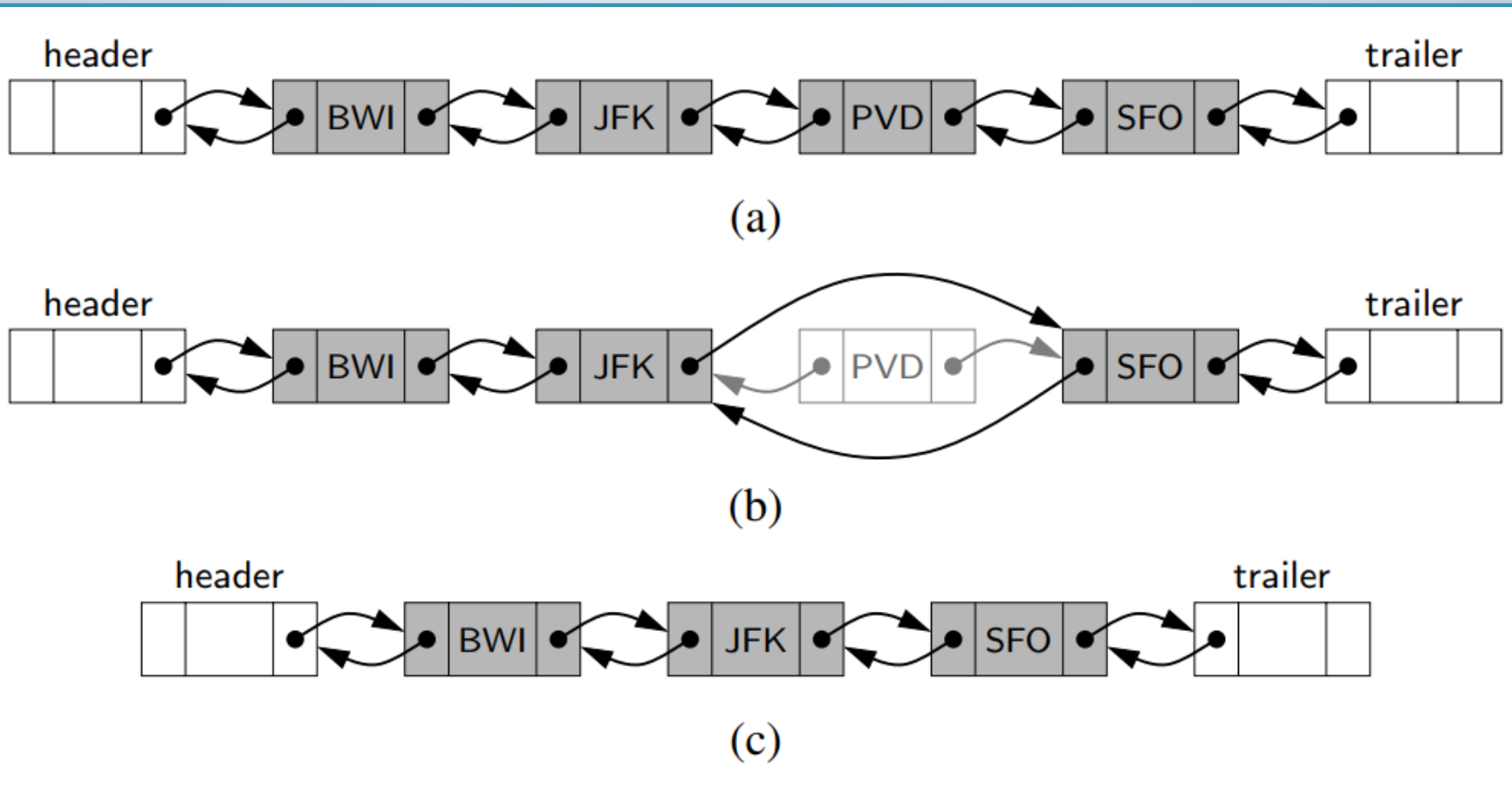
Doubly Linked Lists



Inserting with a Doubly Linked List



Deleting with a Doubly Linked List



Linked List vs Double Linked List

Comparison	Linked List	Doubly Linked List
Access	The singly linked list can be traversed only in the forward direction.	The doubly linked list can be accessed in both directions.
List pointer	It requires only one list pointer variable, i.e., the head pointer pointing to the first node.	It requires two list pointer variables, head and last. The head pointer points to the first node, and the last pointer points to the last node of the list.
Memory space	It utilizes less memory space.	It utilizes more memory space.
Efficiency	It is less efficient as compared to a doubly-linked list.	It is more efficient.
Complexity	In a singly linked list, the time complexity for inserting and deleting an element from the list is O(n).	In a doubly-linked list, the time complexity for inserting and deleting an element is O(1).