Linked Lists

Contents

- What are Linked Lists?
- Singly linked vs doubly linked
- Creating a Linked List
  - Sentinal values
- Adding elements.
- Removing elements

What are Linked Lists

- Linked lists are another structure for storing data.
- Linked lists are built on the idea of a node that stores each element of data.
- Each node contains the data for that node and a pointer to the next and previous nodes in the list.
- Each individual node can be allocated anywhere.
  - Unlike in an array, where each element is sequential in memory

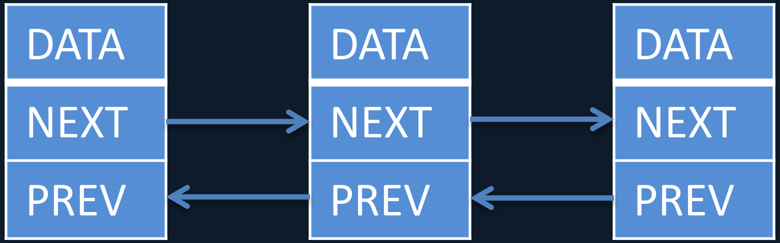
Canberra Institute of Technology

aie SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

What are Linked Lists

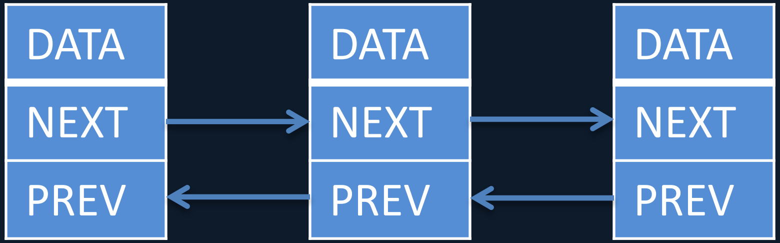
- A linked list is made up of **nodes**
- A node has two things in it.
  - The data that node is storing. This is all an array has.
  - Pointers to the next and the previous node

- To access each element in the list, you start at the first node and 'follow' the pointers to each subsequent node.



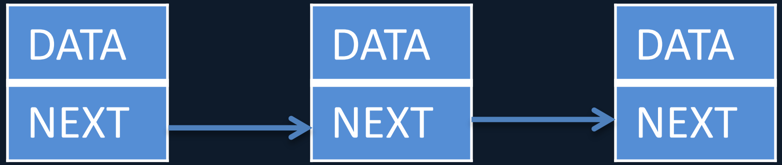
Doubly Linked Lists

- The diagram shown below is a doubly linked list.
- This means that there are pointers to both the next and previous nodes in the list.



Singly linked lists

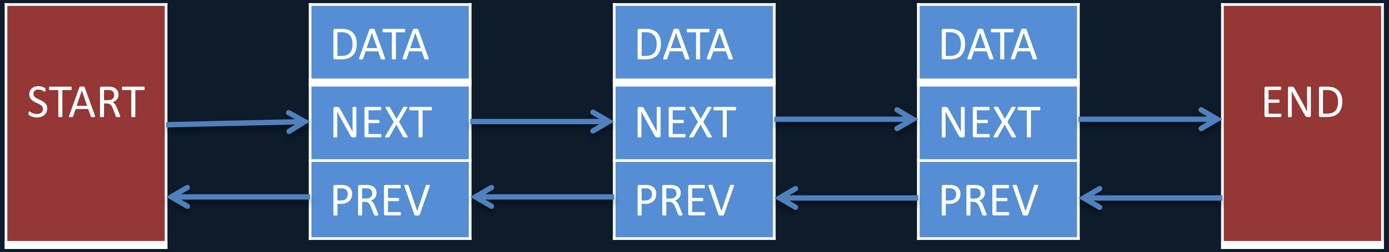
- A singly linked list only has the pointer to the following node.
- Singly linked lists take up less memory, as they don’t need to store the previous pointer
- They are also simpler to implement.
- Typically, if you only need to traverse the data in a single direction, a singly linked list is better.



Sentinel Values

- Linked lists often have special nodes called the sentinel nodes to represent the start and the end of the list.

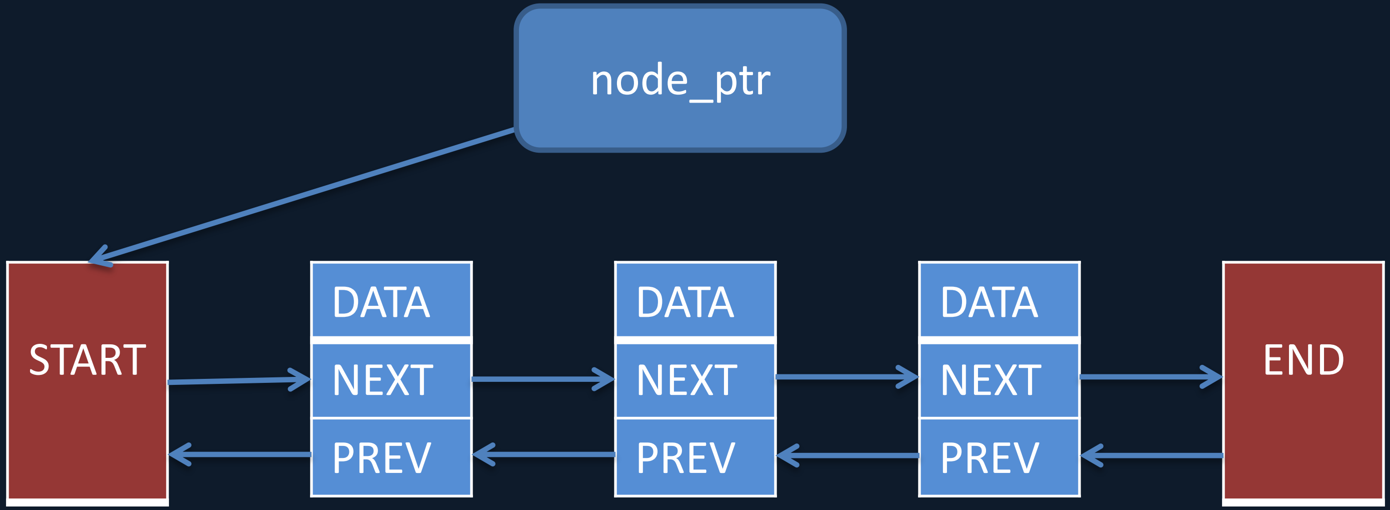
- This allows you to know when to finish iterating.



Iterating Through a Linked List

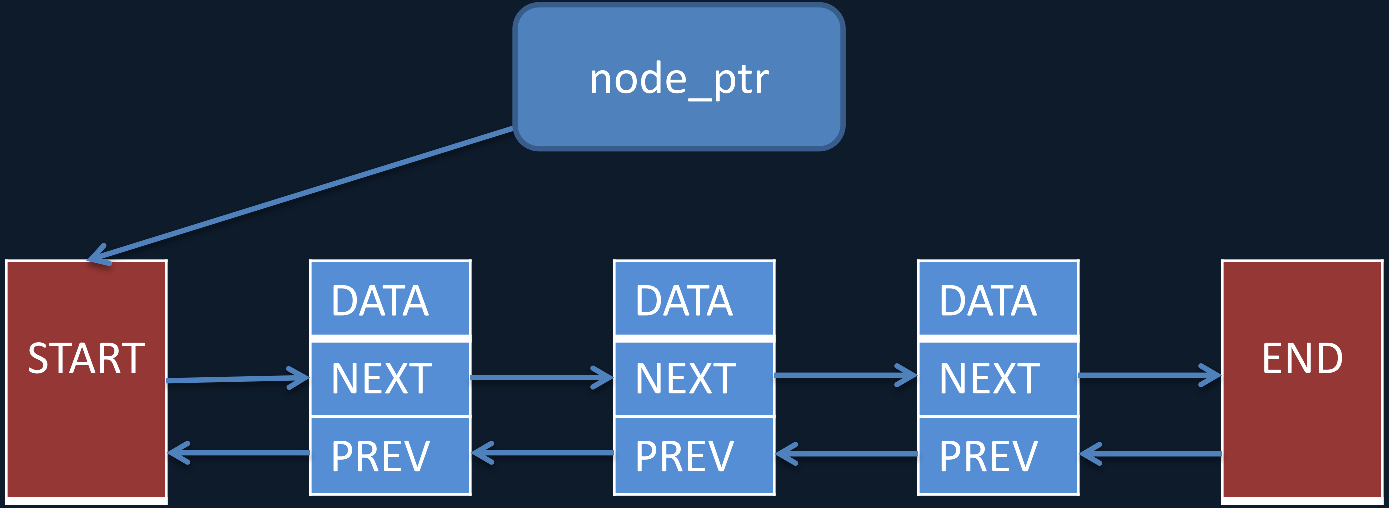
- To iterate through a linked list, we start with a pointer to the first node.
- We then set the pointer to be equal to its own next pointer.
- Repeat.

Iterating Through a Linked List



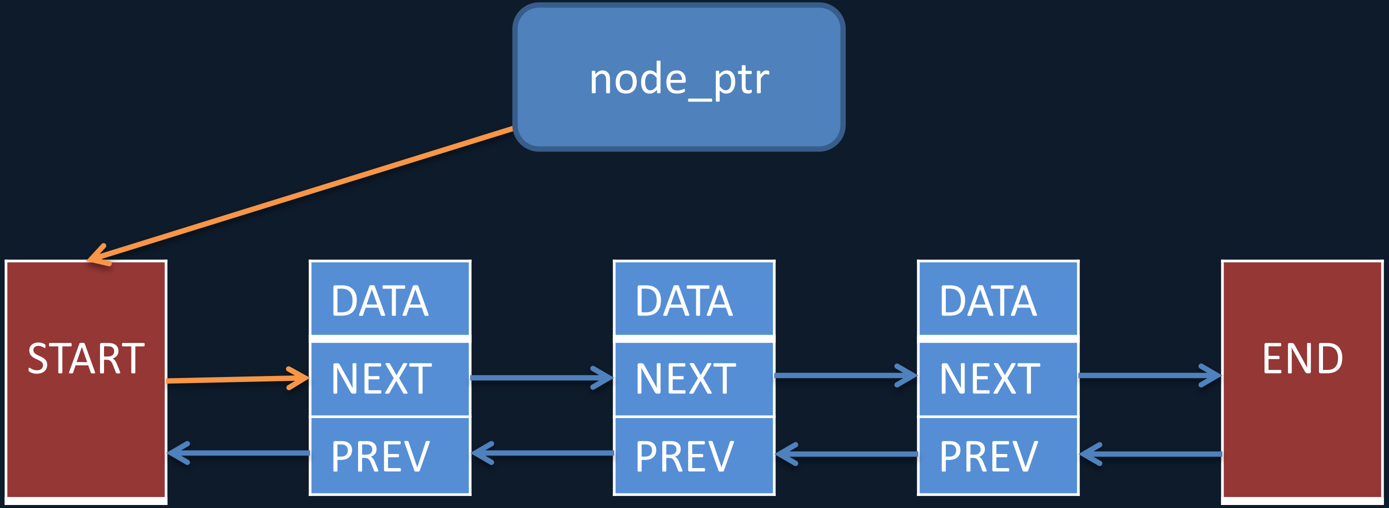
Iterating Through a Linked List

- node_ptr = node_ptr->next;



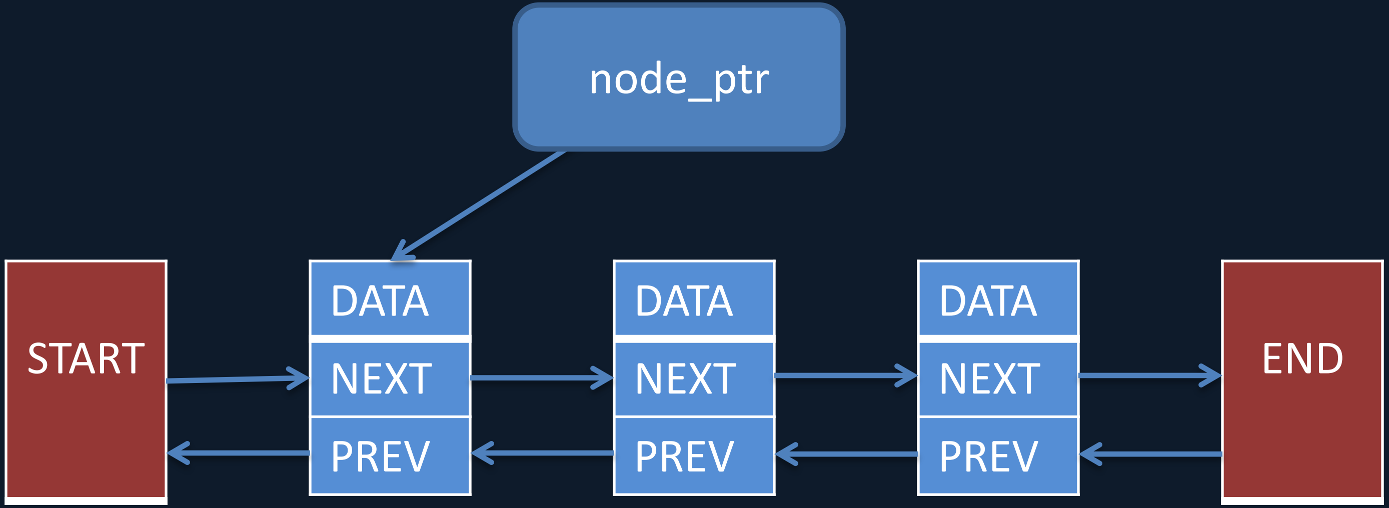
Iterating Through a Linked List

- node_ptr = node_ptr->next;



Iterating Through a Linked List

- node_ptr = node_ptr->next;



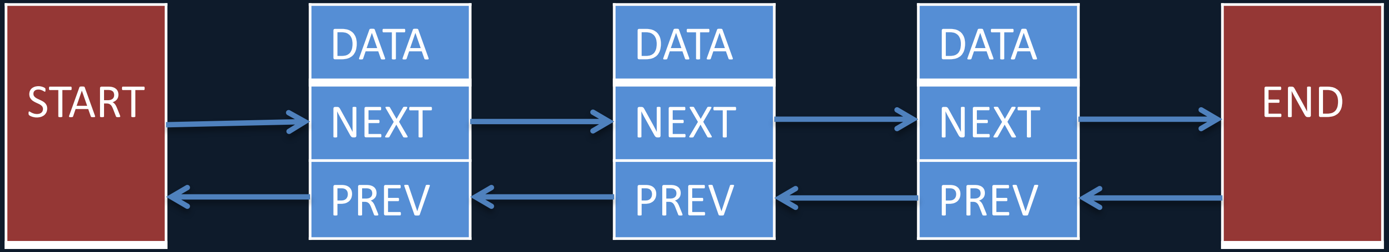
Iterating Through a Linked List

```
FUNCTION PrintLinkedListNodes(start_node, end_node)

    current_node = start_node;

    WHILE current_node != end_node
        print(current_node.data);
        current_node = current_node.next;
    END_WHILE

END FUNCTION
```

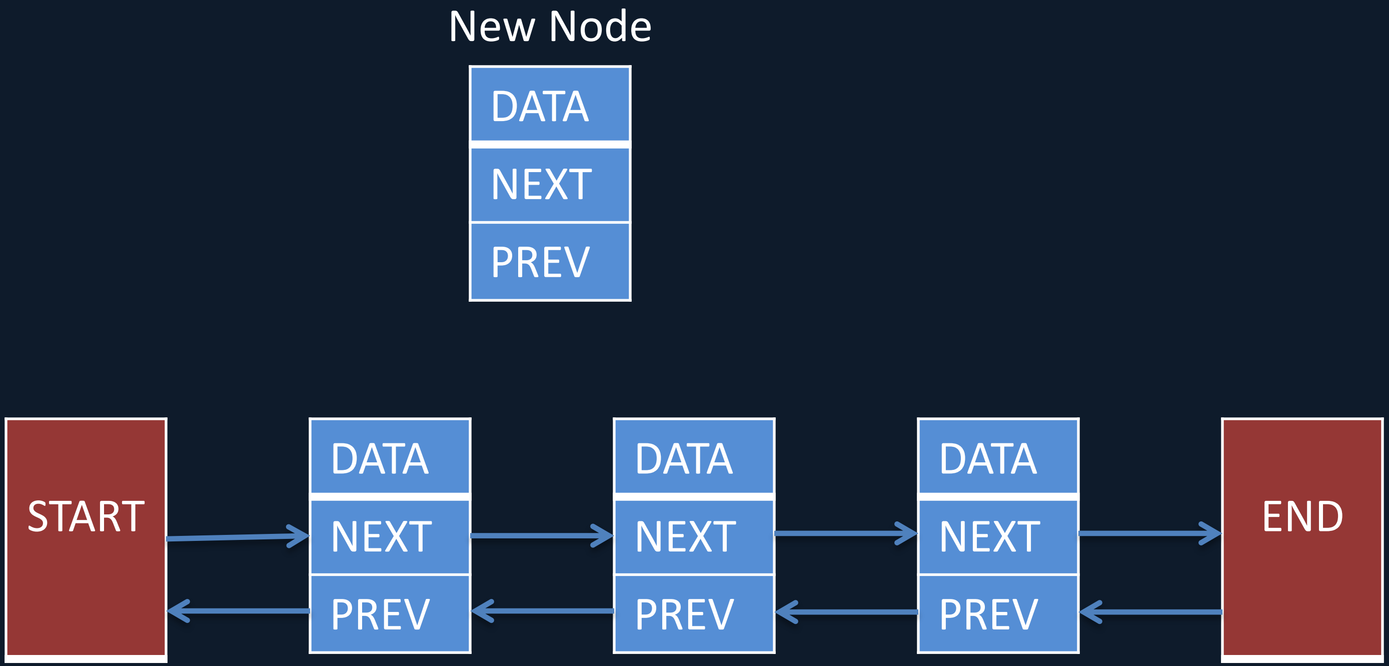
Adding to a Linked List

- Adding to a Linked List can be a bit fiddly.
- Unlike an array, adding a node is the same regardless of where you add it to in the list
- First you create the new node and set its data to be whatever data you want to store.
- Set its next and previous pointers to the element before and after where you want it to be in the list.
- Then point those node's pointers to point back to the new node.

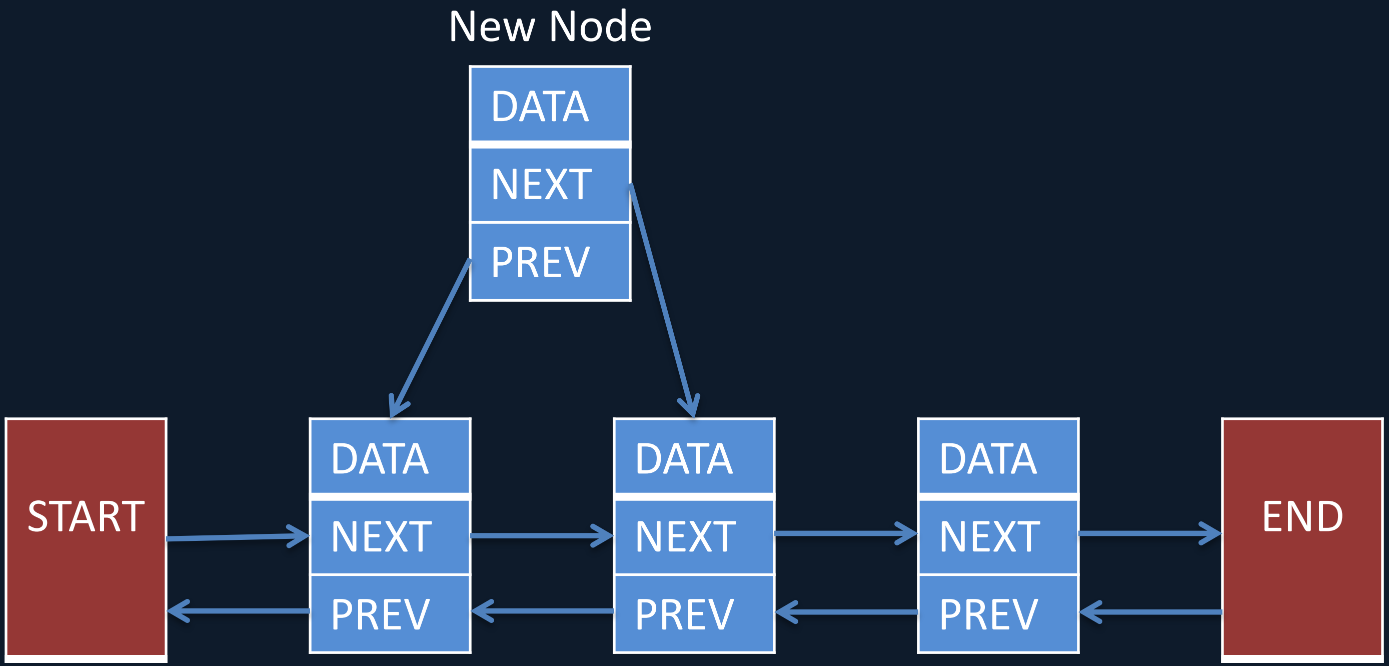
Adding to a Linked List



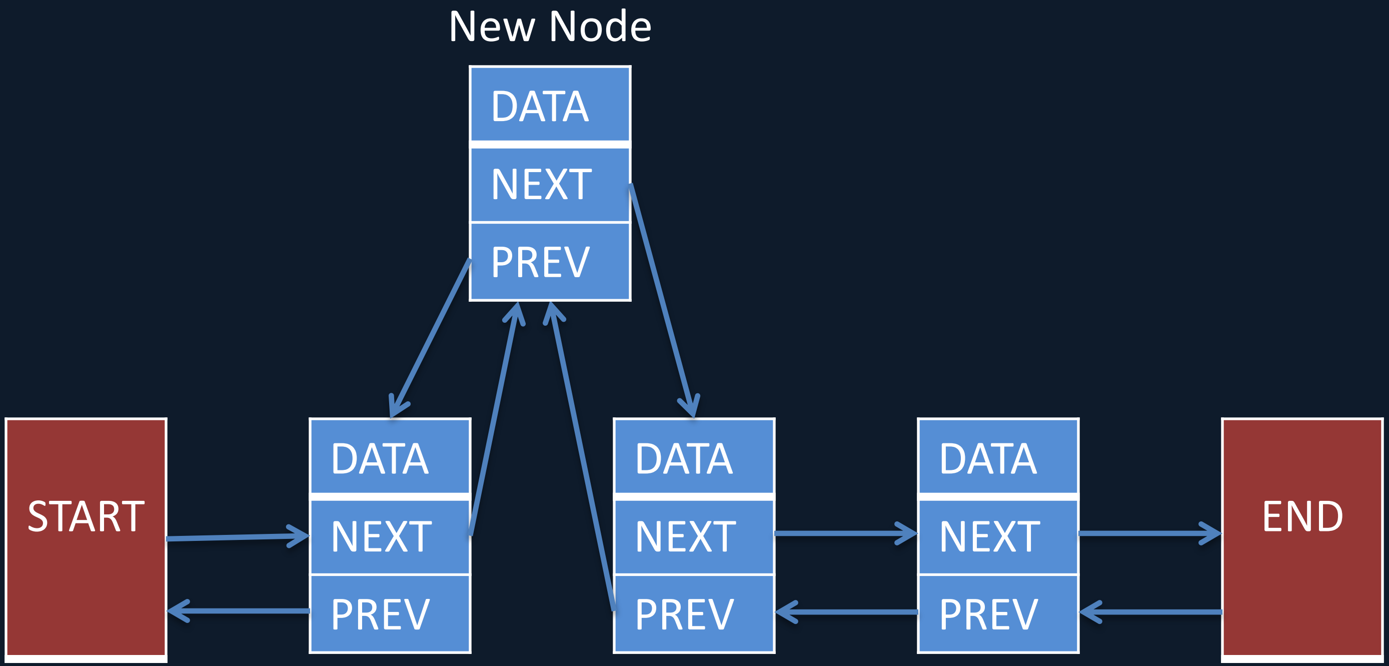
Adding to a Linked List



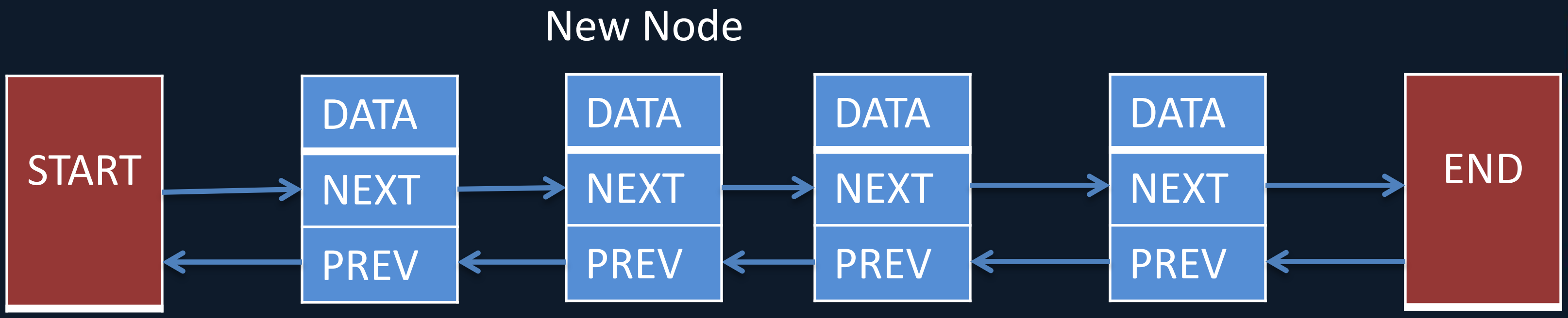
Adding to a Linked List



Adding to a Linked List



Adding to a Linked List



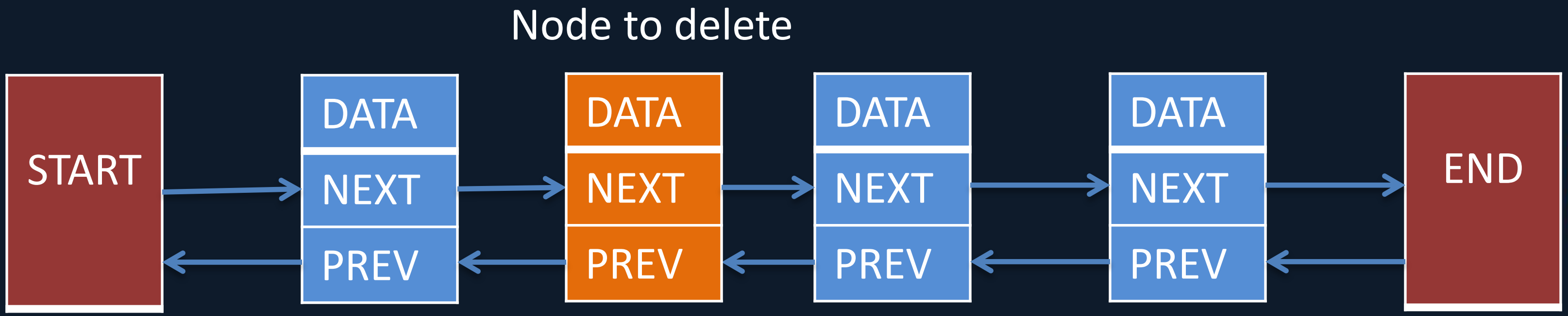
Adding to a Linked List

```
FUNCTION AddNewNode(new_data, node_before, node_after)
    new_node = new Node;

    new_node.data = new_data;

    new_node.next = node_after;
    new_node.prev = node_before;

    node_before.next = new_node;
    node_after.prev = new_node;
END FUNCTION
```

Removing From a Linked List

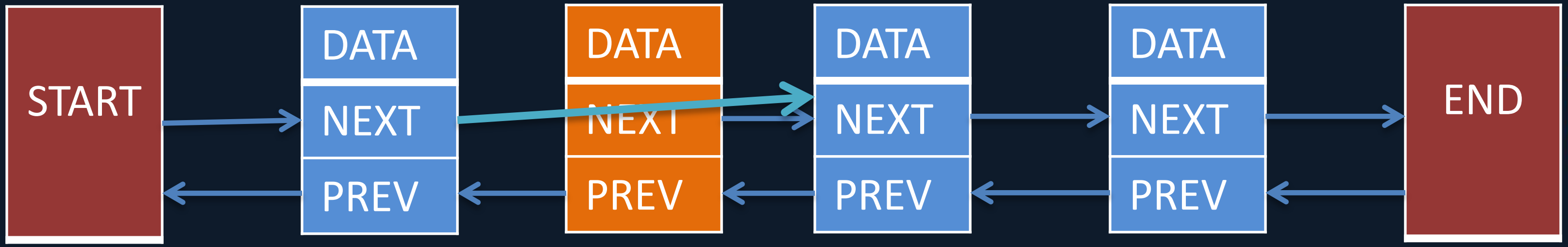
- To remove a node from a linked list we need to take the nodes before and after it and tie them to each other, so they skip over the node to remove
- Then we can just deallocate the node

Removing From a Linked List



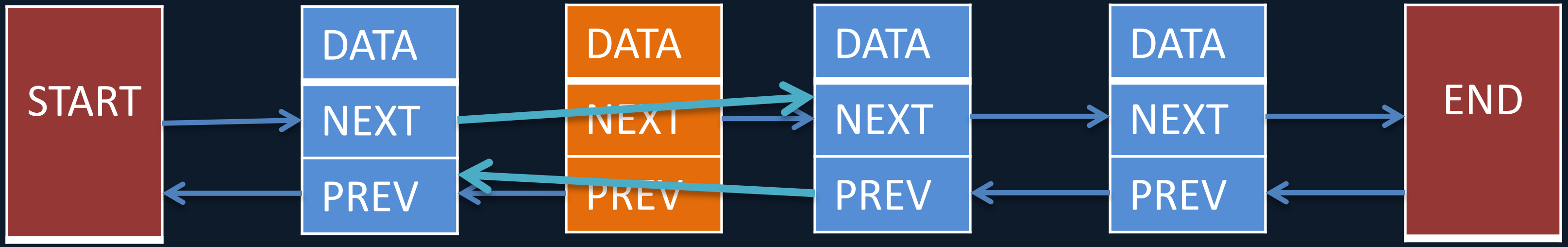
Removing From a Linked List

Node to delete

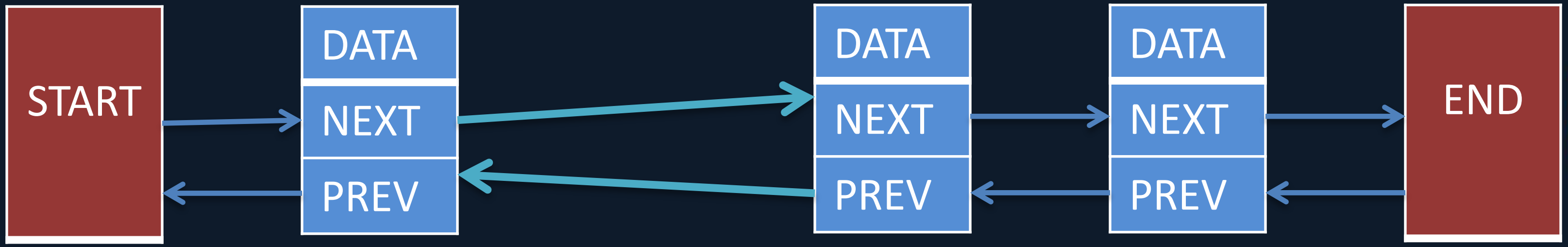


Removing From a Linked List

Node to delete



Removing From a Linked List



Removing From a Linked List

```
FUNCTION RemoveNode(node_to_remove)
    next_node = node_to_remove.next;
    prev_node = node_to_remove.prev;

    prev_node.next = next_node;
    next_node.prev = prev_node;

    delete node_to_remove;
END FUNCTION
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Summary

- Linked Lists are another way of storing data
- They are typically faster than arrays in situations where elements will be added, removed and shuffled regularly.

References

- Sedgewick, R, and Wayne, K “*Algorithms*”, 4th Ed, Chp 1, Addison-Wesley (2011)