Singly Linked List - Insertion

In this lesson, we'll learn how to insert an element in a singly linked list.

We'll cover the following

Structure

Each node contains a value and a pointer to the next node.

```
struct Node {
    int val;
    Node* next;

    Node (int val) {
        this->val = val;
        this->next = NULL;
    }
}
```

Insertion

There are 3 cases:

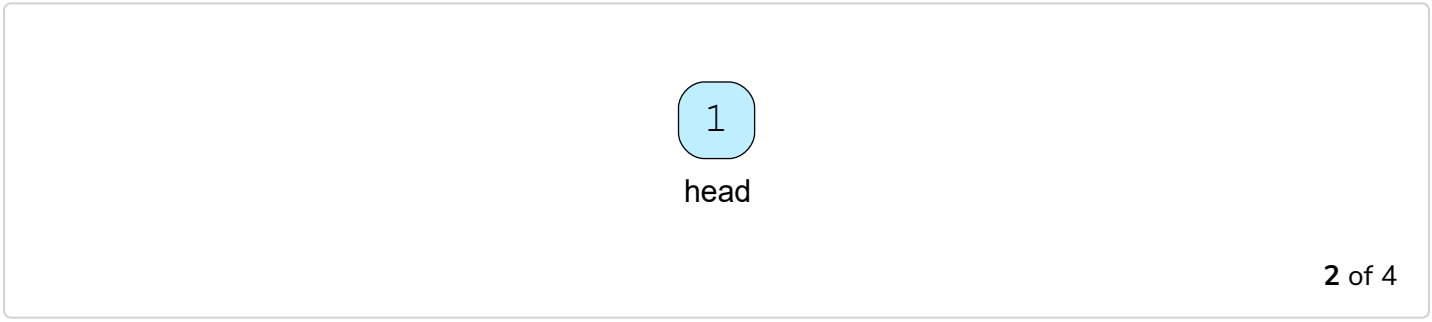
- Insert at beginning
- Insert at some given position
- Insert at end

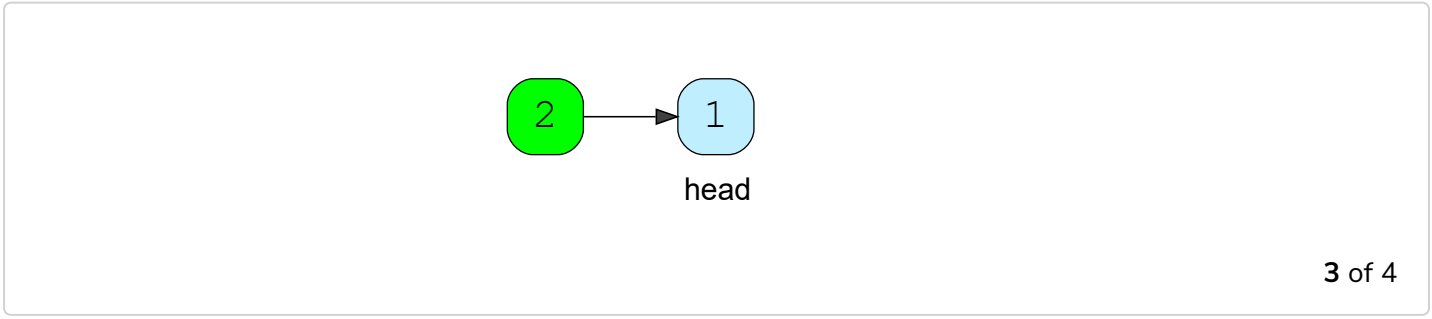
The worst-case complexity of insertion in a linked list is $O(N)$.

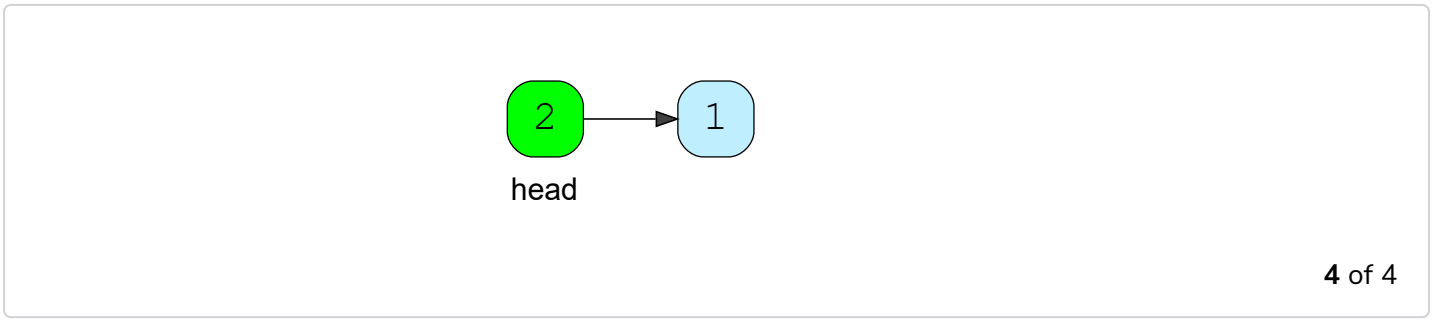
Insert at head

- Empty List: New node becomes the head

- Otherwise, a new node points to the existing head. The new node is now the head.







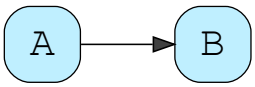

```
#include<iostream>

using namespace std;

struct Node {
  int val;
  Node* next;

  Node(int val) {
    this -> val = val;
  }
};

void print_list(Node* head) {
  struct Node* pCrawl = head;
  cout << " -> ";
  while (pCrawl != NULL) {
```

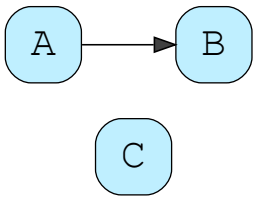
```
    cout << (pCrawl -> val) << " -> ";
    pCrawl = pCrawl -> next;
  }

  cout << "\n";
}

void insert_at_head(Node* &head, int val) {
  if (head == NULL) {     // Empty List
    head = new Node(val);
    return;
  }
  Node* newNode = new Node(val);
  newNode -> next = head;
  head = newNode;
}

int main() {
  Node* head = NULL;
  print_list(head);
  insert_at_head(head, 3); print_list(head);
  insert_at_head(head, 2); print_list(head);
  insert_at_head(head, 1); print_list(head);
}
```

Insert at given position

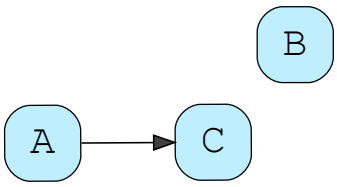
Let's say you want to insert at position 3. Iterate through 3 nodes. Let this 3^{rd} node be A and the next node be B. To insert C between A and B:

- Create new node C
- A->next points to C
- C->next points to B

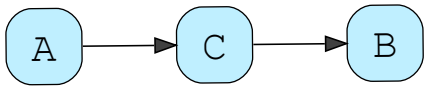




2 of 4



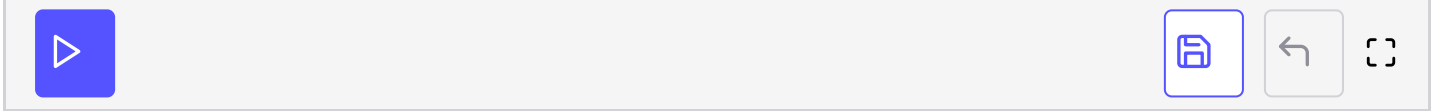
3 of 4



4 of 4

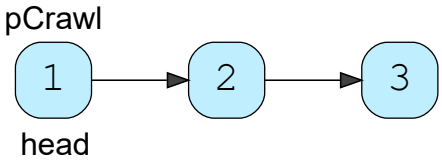
```
#include<iostream>

using namespace std;

struct Node {
  int val;
  Node* next;

  Node(int val) {
    this -> val = val;
```

```
  }
};

void print_list(Node* head) {
  struct Node* pCrawl = head;
  cout << " -> ";
  while (pCrawl != NULL) {
    cout << (pCrawl -> val) << " -> ";
    pCrawl = pCrawl -> next;
  }
  cout << "\n";
}

void insert_at_head(Node* &head, int val) {
  if (head == NULL) {     // Empty List
    head = new Node(val);
    return;
  }
  Node* newNode = new Node(val);
  newNode -> next = head;
  head = newNode;
}


void insert_at_position(Node* &head, int val, int pos) {
  struct Node* pCrawl = head;
  for (int i = 0; i < pos - 1; i++)
    pCrawl =  pCrawl -> next;

  Node *A = pCrawl;
  Node *B = pCrawl->next;
  Node* C = new Node(val);
  A -> next = C;
  C -> next = B;
}


int main() {
  Node* head = NULL;
  print_list(head);
  insert_at_head(head, 4); print_list(head);
  insert_at_head(head, 1); print_list(head);
  insert_at_position(head, 2, 1); print_list(head);  // 0-based position
  insert_at_position(head, 3, 2); print_list(head);
}
```

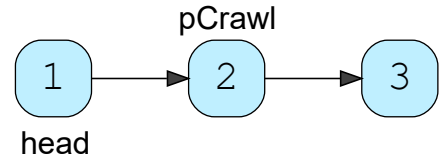


Insert at end

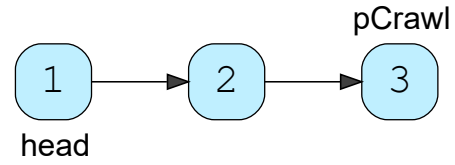
1. Iterate to last node (A)
2. Create a new node
3. A->next points to the new node



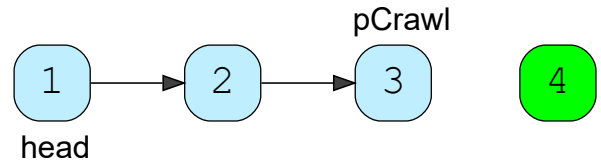
1 of 5



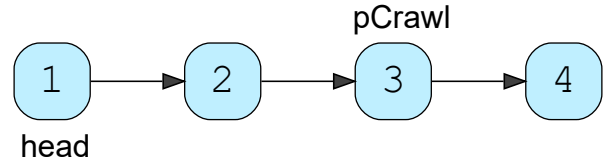
2 of 5



3 of 5



4 of 5



5 of 5

```
#include<iostream>

using namespace std;

struct Node {
  int val;
  Node* next;
```

```
  Node(int val) {
    this -> val = val;
  }
};

void print_list(Node* head) {
  struct Node* pCrawl = head;
  cout << " -> ";
  while (pCrawl != NULL) {
    cout << (pCrawl -> val) << " -> ";
    pCrawl = pCrawl -> next;
  }
  cout << "\n";
}

void insert_at_end(Node* &head, int val) {
  // List is empty
  if (head == NULL) {
    head = new Node(val);
    return ;
  }
  struct Node* pCrawl = head;
  while(pCrawl->next != NULL) {        // iterate to last node
    pCrawl =  pCrawl -> next;
  }
  pCrawl -> next = new Node(val);
}


int main() {
  Node* head = NULL;
  print_list(head);
  insert_at_end(head, 1); print_list(head);
  insert_at_end(head, 2); print_list(head);
  insert_at_end(head, 3); print_list(head);
}
```

Finally, let's see the deletion operation on a singly linked list in the next lesson.