



THE UNIVERSITY
of ADELAIDE



CRICOS PROVIDER 00123M

School of Computer Science

COMP SCI 1103/2103 Algorithm Design & Data Structure

Complexity – Linked Lists

adelaide.edu.au

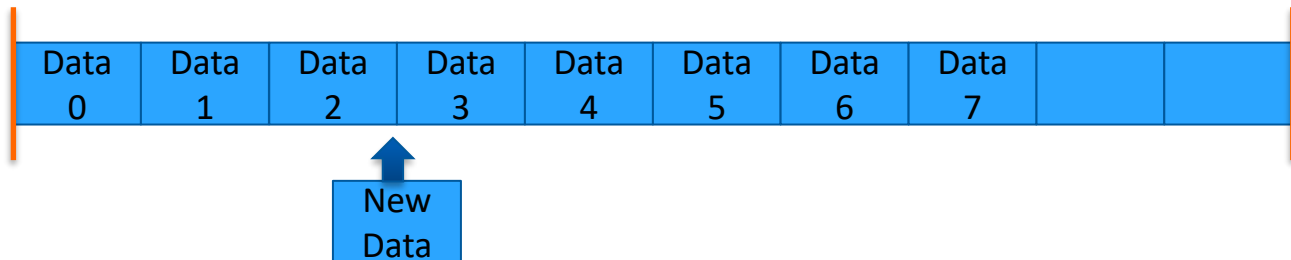
seek LIGHT

Overview

- Case Study - Linked list
 - Impact of data structure choices on run time

Impact on Complexity of Data Structures

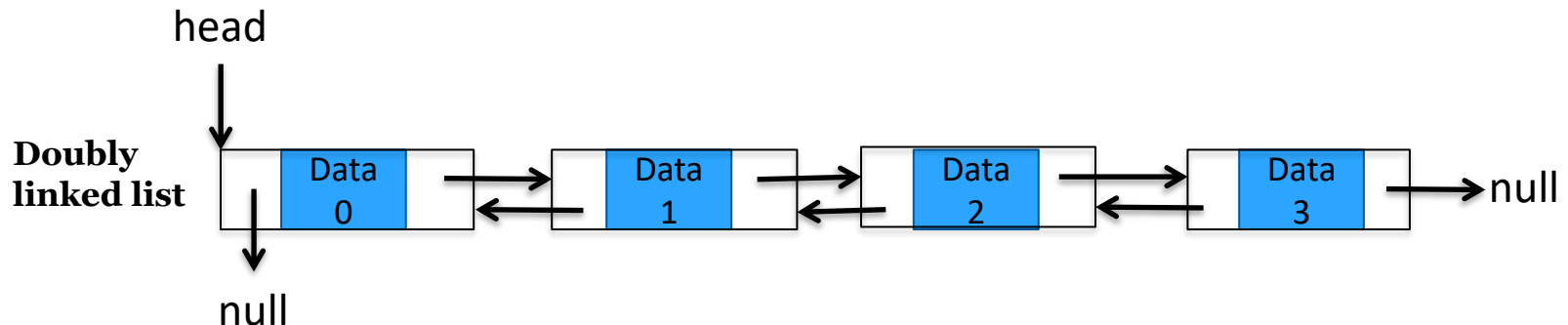
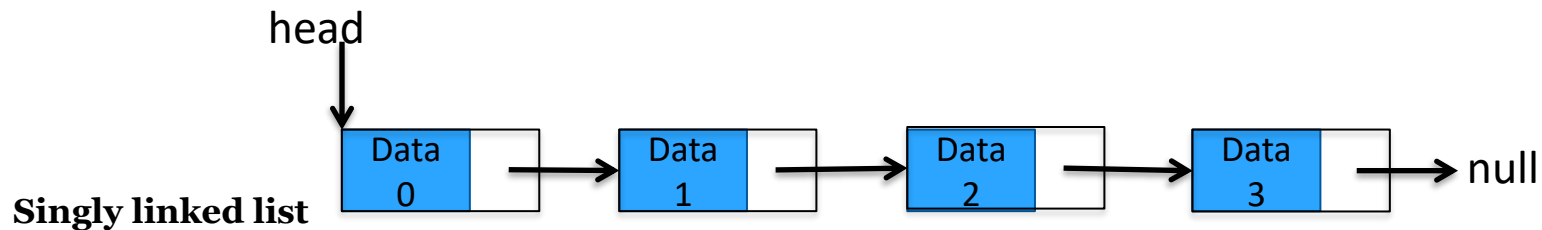
- In an array
 - the items are placed in **consecutive places** in the memory
 - Some operations in the middle of the list are costly
 - Insert
 - Delete



- Static structure with a fixed total size
- How much does it take to add an item at the end?
- Insert at the end takes $O(1)$
- But when there is not enough room, it takes $O(n)$
because the whole array needs to be moved to a larger space

Linked Lists

- A dynamic data structure for representing a list, in which each item of the list is stored in a separate object called **Node**
- Linked Lists have faster **insert/delete**, slower **access** compared to arrays.
- Nodes consist of an **item** and a **reference to the next Node**.
- We need a reference to the first node called **head**



Node Struct and Methods

We need a structure for nodes and a number of methods

```
Struct Node{
```

```
    Type data;
```

```
    Pointer nextNode;
```

```
}
```

```
Class LinkedList{
```

```
    Pointer head;
```

```
// methods including insert, delete, search and traverse
```

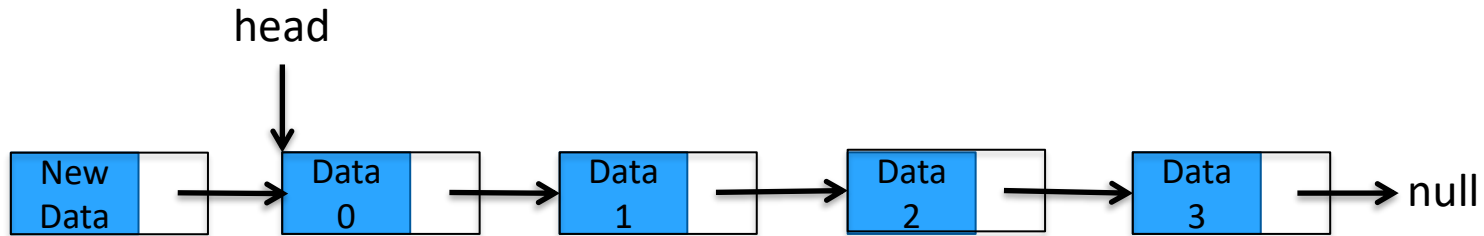
```
    InsertAtFront(newData);
```

```
    Pointer Search(item);
```

```
    InsertAfter(newData, itemBefore);
```

```
}
```

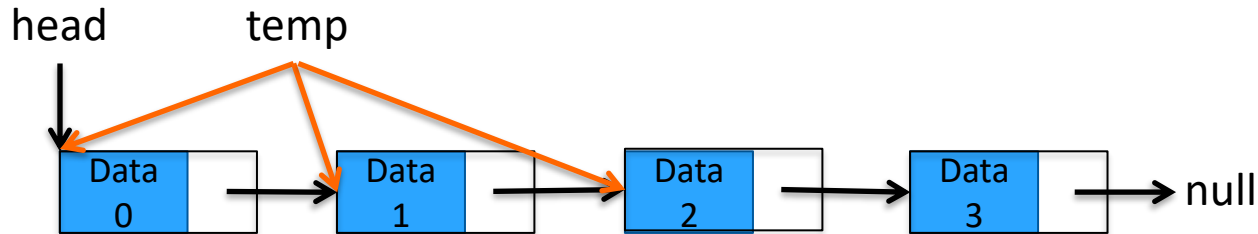

InsertAtFront method



```
InsertAtFront(newData){  
    newNode= new Node()  
    newNode.data= newData  
    newNode.nextNode= head  
    head = addressOf(newNode)  
}
```

Complexity?

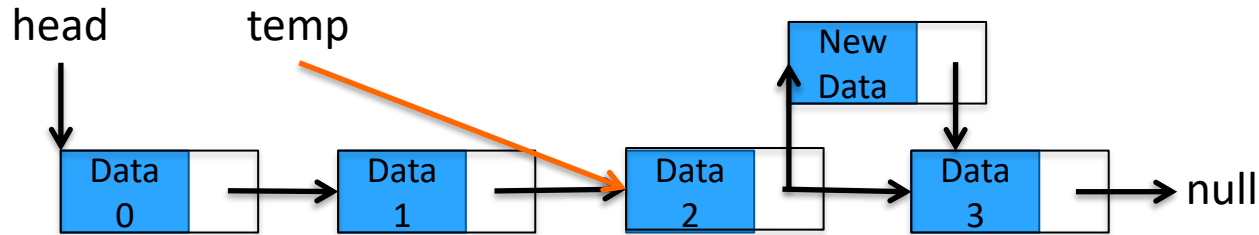
Search method



```
pointer Search(item){  
    temp=head  
    while(temp!=null and temp.data!=item)  
        temp= temp.nextNode  
    return temp  
}
```

Complexity?

InsertAfter method



```
InsertAfter(newData, itemBefore){  
    temp=Search(itemBefore)  
    if (temp=null)  
        error: item not found  
    newNode= new Node()  
    newNode.data= newData  
    newNode.nextNode= temp.nextNode  
    temp.nextNode= addressOf(newNode)  
}
```


Question for Discussion

- What affect does having a doubly linked list have on insert, search, delete...?
- When would it be preferable to use:
 - An array
 - A singly linked list
 - A doubly linked list

Summary on Linked lists

- We learned situations where array is not a good choice for representing lists
- We defined linked lists and learned a few methods for doing operations on linked lists
- Arrays:
 - Add at the end if enough space: $O(1)$
 - Due to fixed size, adding a new item at the end may take $O(n)$
 - Direct access to items by index number : $O(1)$
 - Shifts data when an item is added in the middle of the list or deleted from it: $O(n)$
- Linked Lists:
 - Dynamically grows or shrinks: add and remove take $O(1)$
 - No direct access by index number; Links should be followed: $O(n)$
 - Adding and removing items from the middle of the list include search: $O(n)$, but not as costly as shifting the data



THE UNIVERSITY
of ADELAIDE

