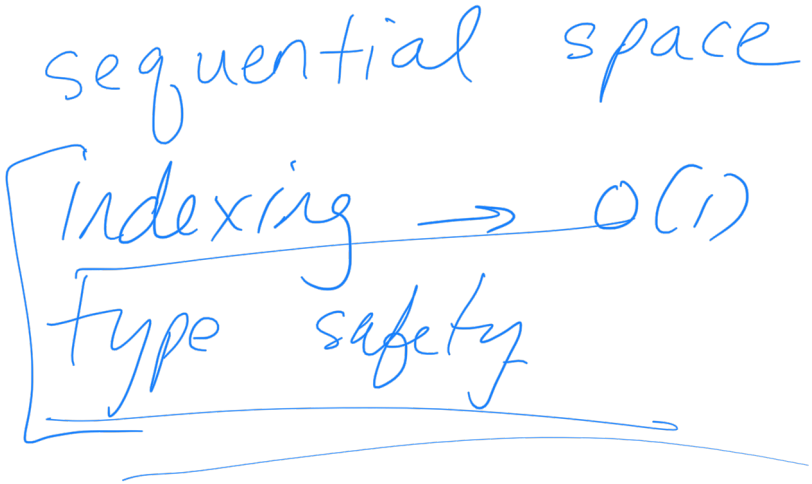
Arrays & Linked Lists

- Arrays
- Linked Lists

Arrays



Arrays: Properties and Operations

- fixed size
- elements accessed through indexing: an O(1) operation

Arrays: Properties and Operations

For the questions below, assume that you have a variable pointing to the next open spot. Also, let N = the number of elements in the array.

How much work is required to *insert* an element into an unsorted array?

O(1)

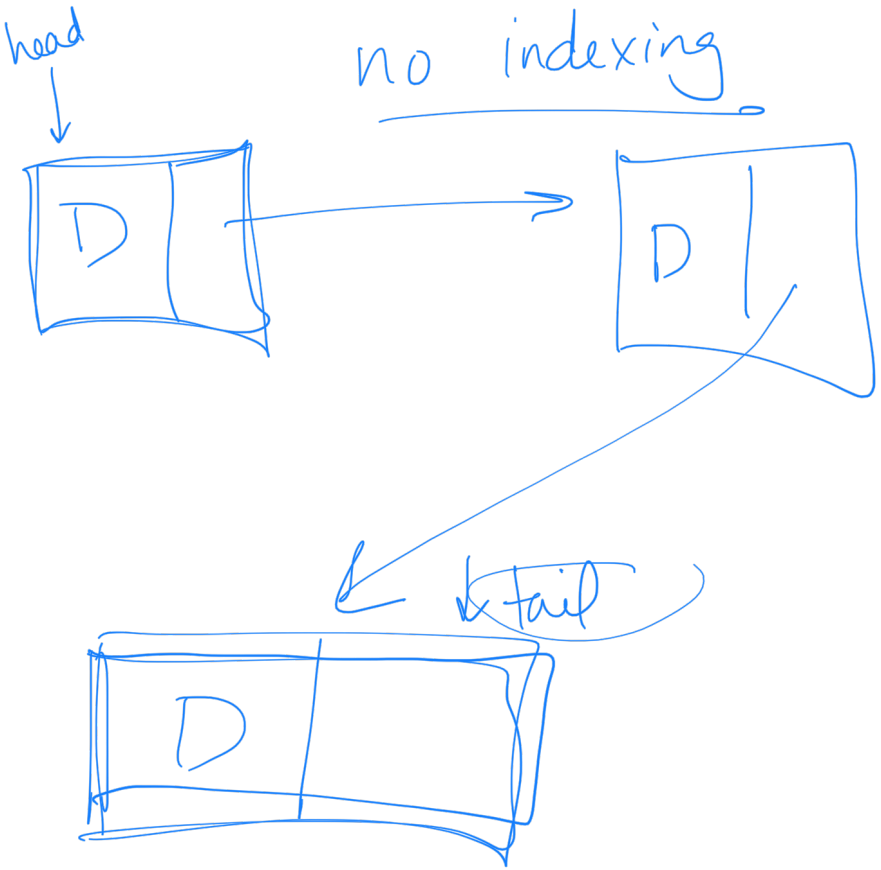
How much work is required to *insert* an element into a sorted array (so that the array is still sorted)?

O(N)

How much work is it to dynamically resize an array?

O(N)

Linked Lists



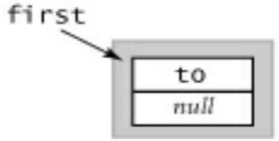
A linked list is...

null

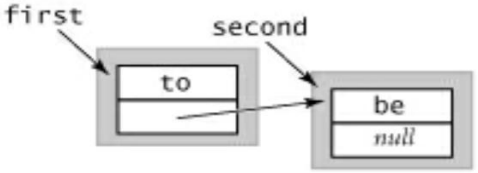
OR

a node pointing to a linked list

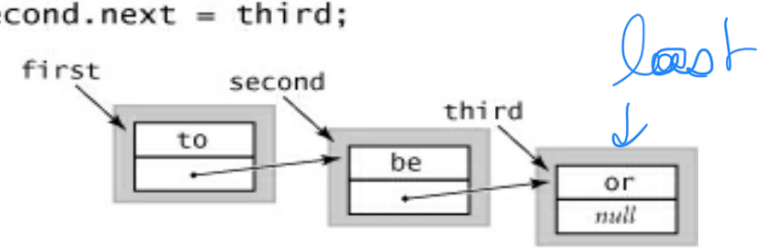
```
Node first  = new Node();
first.item  = "to";
```




```
Node second = new Node();
second.item = "be";
first.next  = second;
```




```
Node third  = new Node();
third.item  = "or";
second.next = third;
```



Linking together a list

Picture from [1]

Linked Lists: Properties and Operations

- no fixed size
- each element has a pointer to the next element (or null if it is the last element in the list)

Linked Lists: Properties and Operations

For these questions, let N be the number of elements in the list. Also, assume there are pointers to the head and to the tail.

How much work is required to *add* an element to the beginning of the list?

O(1)

How much work is required to *add* an element to the end of the list?

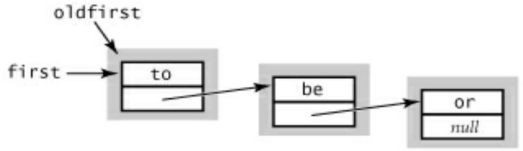
O(1)

How much work is required to *insert* an element into the middle of the list?

O(N)

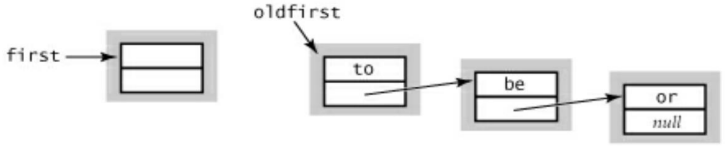
save a link to the list

```
Node oldfirst = first;
```



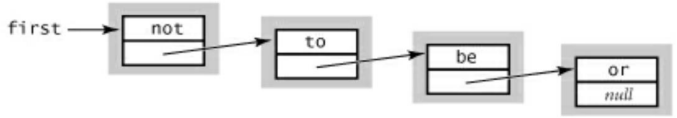
create a new node for the beginning

```
first = new Node();
```



set the instance variables in the new node

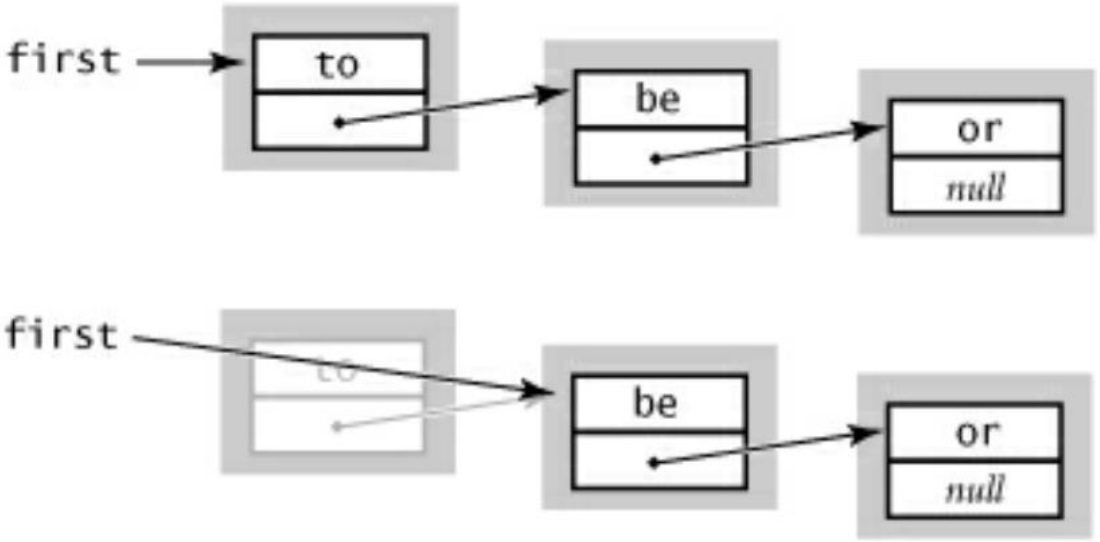
```
first.item = "not";
first.next = oldfirst;
```



Inserting a new node at the beginning of a linked list

Picture from [1]

```
first = first.next;
```

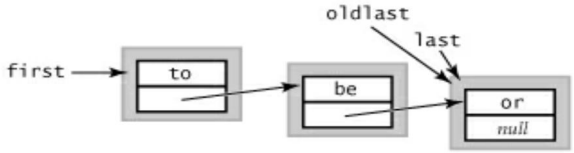


Removing the first node in a linked list

Picture from [1]

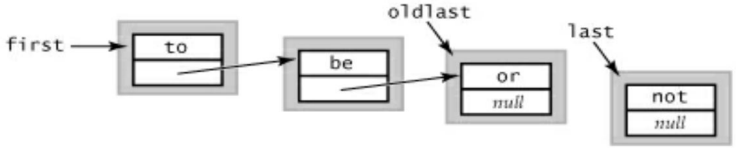
save a link to the last node

```
Node oldlast = last;
```



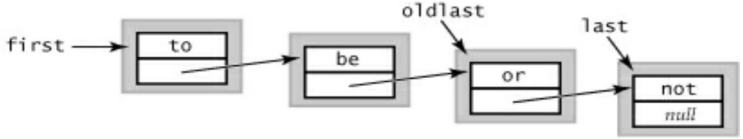
create a new node for the end

```
Node last = new Node();
last.item = "not";
```



link the new node to the end of the list

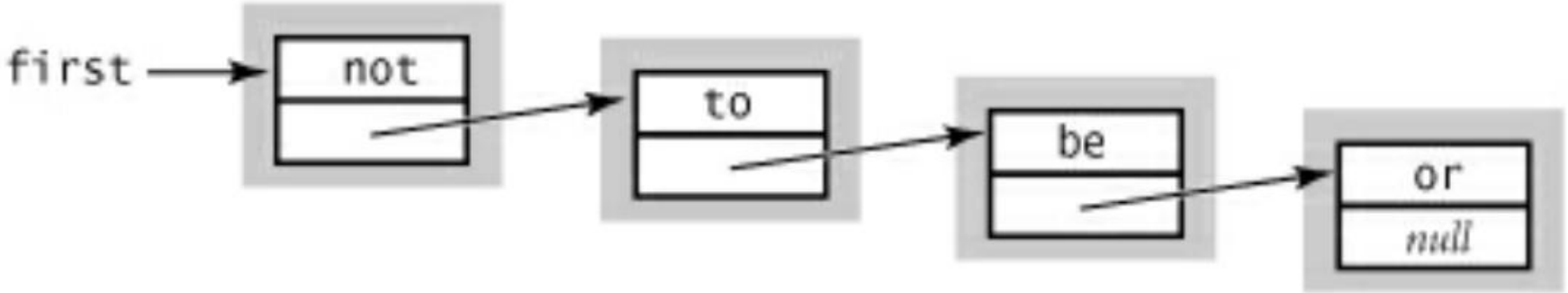
```
oldlast.next = last;
```



Inserting a new node at the end of a linked list

Picture from [1]

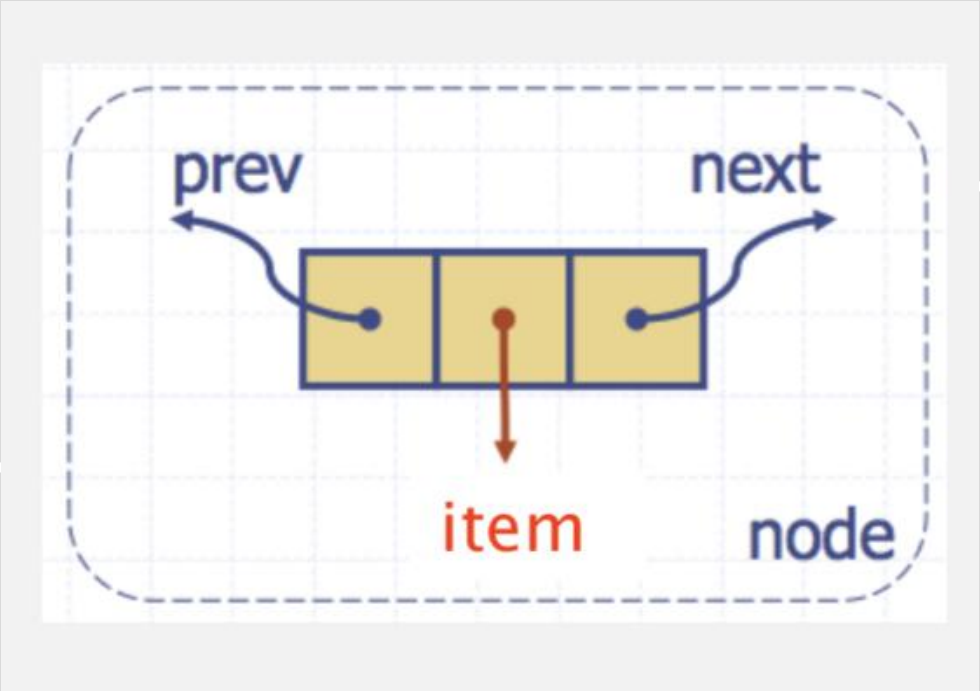
Singly linked list: each node has a **single** pointer pointing to the **next** node



Picture from [1]

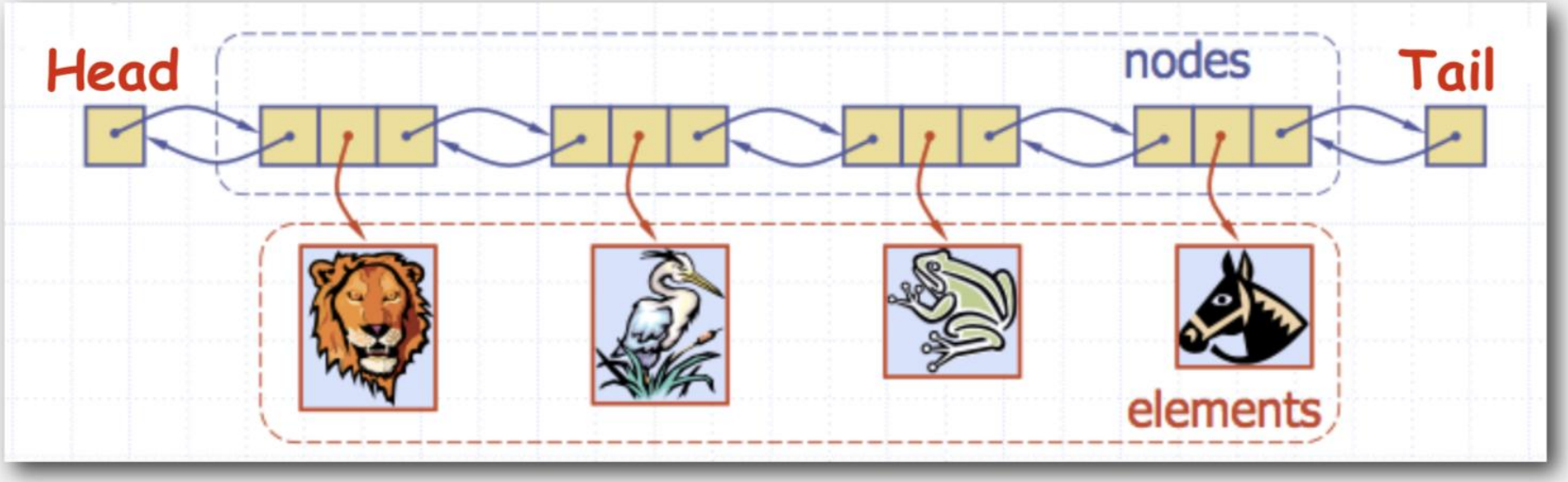
Doubly linked list...

Change the nodes to have two pointers



Picture from [2]

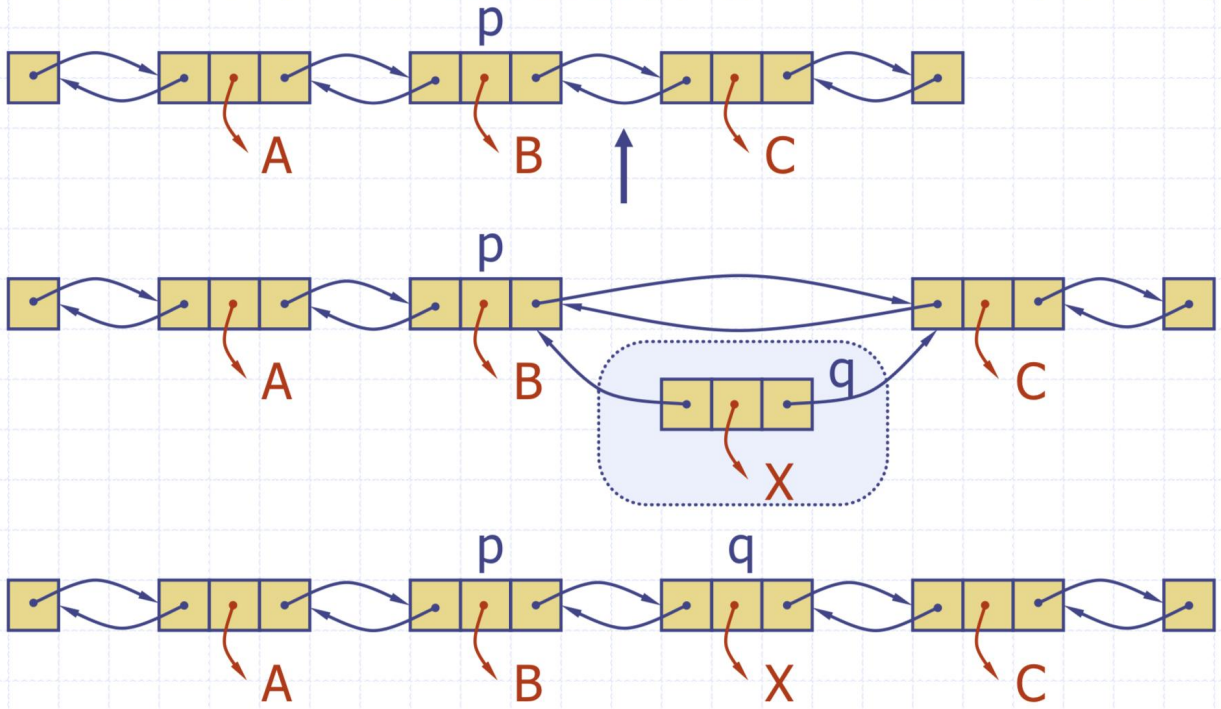
Doubly linked list: how do **head** and **tail** nodes differ from internal nodes?



Picture from [2]

Insertion

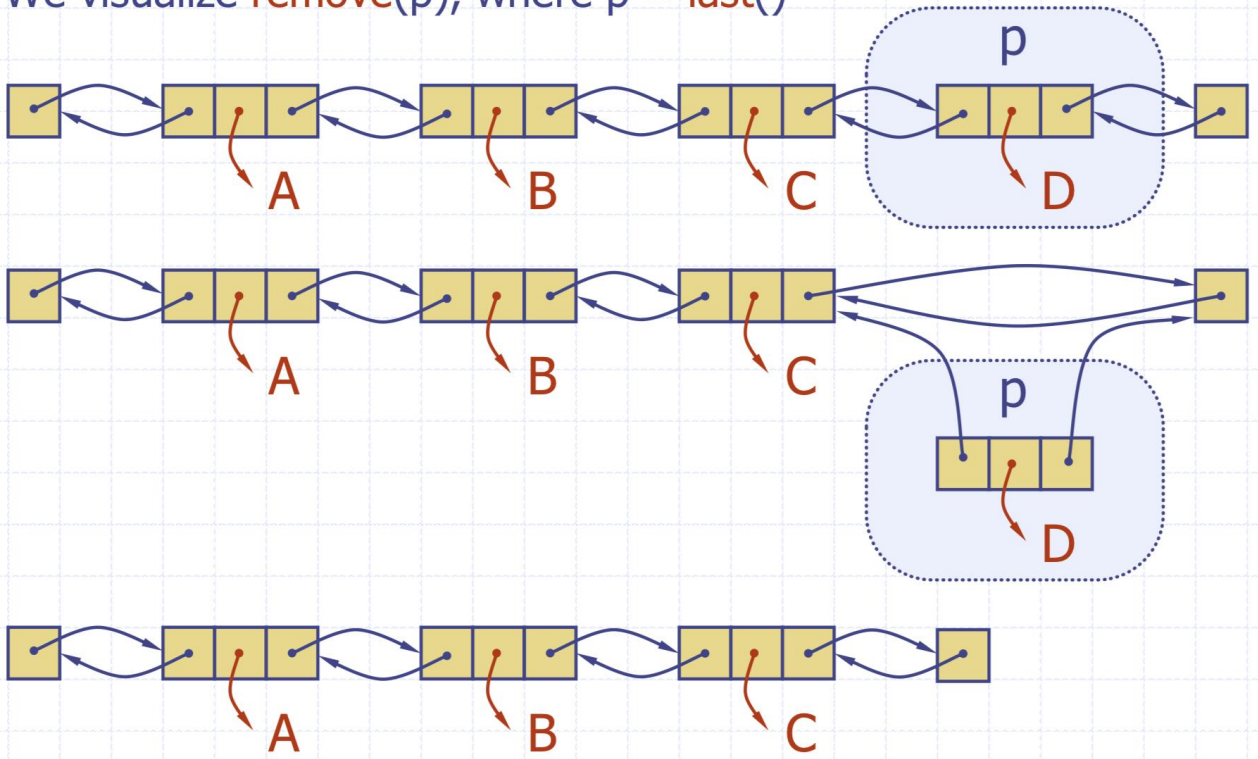
- We visualize operation insertAfter(p, X), which returns position q



Picture from [2]

Deletion

- We visualize remove(p), where p = last()



Picture from [2]

Arrays vs. Linked Lists

Arrays vs. Linked Lists

Arrays

Pros:

- easy access to any element
- doesn’t require space for pointers

Cons:

- fixed size

Linked Lists

Pros:

- no fixed size--easily growable or shrinkable
- adding to front and back of list is easy

Cons:

- space overhead for pointers
- no direct access to internal nodes

Warning:
An ArrayList is not a linked list! In this class, you may only use ArrayLists in your projects if you are given explicit permission to do so.

Practice. Given *N* elements, what would be the runtime of a *findMin* operation if...

- ...the elements are in a sorted array? O(1)
- ...the elements are in an unsorted array? O(N)
- ...the elements are in a sorted singly linked list? O(1)
- ...the elements are in an unsorted singly linked list? O(N)
- ...the elements are in a sorted doubly linked list? O(1)
- ...the elements are in an unsorted doubly linked list? O(N)

Practice. Given N elements, what would be the runtime of a *findMax* operation if...

- ...the elements are in a sorted array? $O(1)$
- ...the elements are in an unsorted array? $O(N)$
- ...the elements are in a sorted singly linked list? $O(1)$
- ...the elements are in an unsorted singly linked list? $O(N)$
- ...the elements are in a sorted doubly linked list? $O(1)$
- ...the elements are in an unsorted doubly linked list? $O(N)$

Practice. Given N elements, what would be the runtime of a *findMedian* operation if...

- ...the elements are in a sorted array? $O(1)$
- ...the elements are in an unsorted array? $O(N \log N)$
- ...the elements are in a sorted singly linked list? $O(N)$
- ...the elements are in an unsorted singly linked list? $O(N \log N)$
- ...the elements are in a sorted doubly linked list? $O(N)$
- ...the elements are in an unsorted doubly linked list?

$N \log N + N$

$O(N \log N)$

Practice. Given N elements, what would be the runtime of a *find(x)* operation if...

- ...the elements are in a sorted array? $O(\log N)$
- ...the elements are in an unsorted array? $O(N)$
- ...the elements are in a sorted singly linked list? $O(N)$
- ...the elements are in an unsorted singly linked list? $O(N)$
- ...the elements are in a sorted doubly linked list? $O(N)$
- ...the elements are in an unsorted doubly linked list? $O(N)$

References

[1] *Algorithms, Fourth Edition;* Robert Sedgewick and Kevin Wayne (and associated slides)
[2] *Data Structures & Algorithms in Java,* Goodrich & Tamassia (and associated slides)