

Array and List Operations

Prof. Wade Fagen-Ulmschneider

I ILLINOIS

Arrays and Lists are both ordered collections of data. There are several key operations common to both all ordered collections that are worth analyzing.

Array



List



Access a Given Index

Objective: Access a given index in the collection.

Array:	O(1) Direct access via offset formula
List:	O(n) Must traverse every element to reach the index

Insert at front

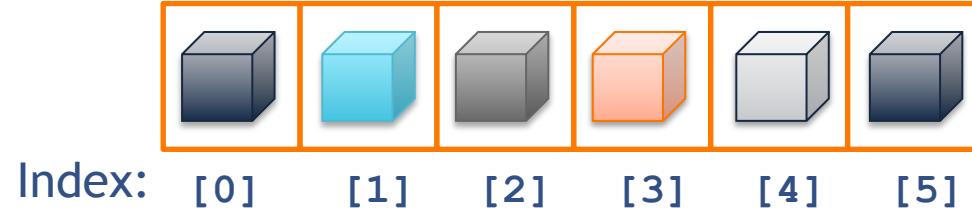
Objective: Insert an element at the front

Array:	O(1)* Amortized constant time when array size is doubled when copied
List:	O(1) Fixed constant time by adding the new data at the head of the list

Find Data

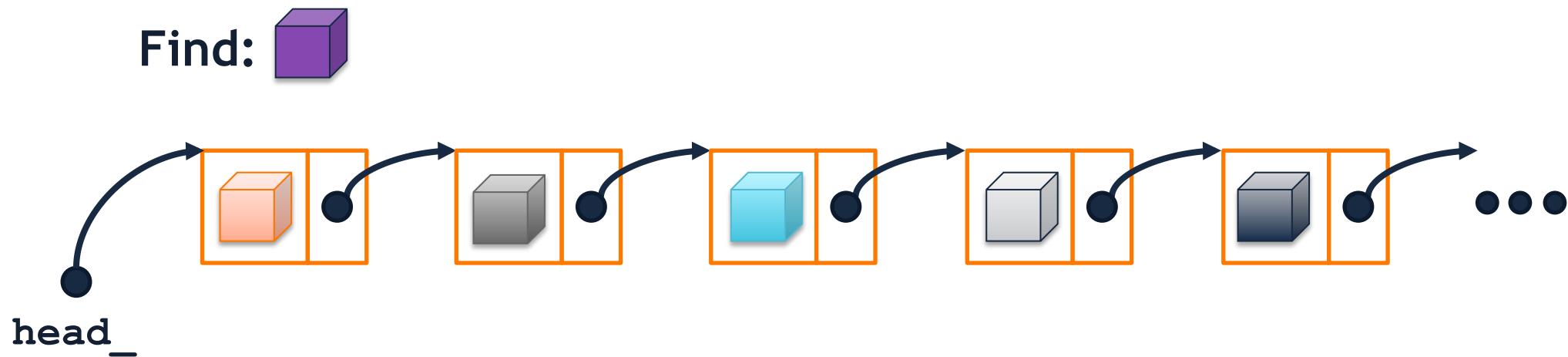
Objective: Given data, find the location of that data in the collection.

Find: 



Find Data

Objective: Given data, find the location of that data in the collection.



Find Data

Objective: Given data, find the location of that data in the collection.

Array:	O(n) Requires searching up to the full array, one-by-one, to find a match
List:	O(n) Requires searching up to the full list, one ListNode at a time, to find a match

array/ex4/main.cpp

```
28 // Find a specific `target` cube in the array:  
29 Cube target = Cube(400);  
30 for (unsigned i = 0; i < cubes.size(); i++) {  
31     if (target == cubes[i]) {  
32         std::cout << "Found target at [" << i << "]" << std::endl;  
33     }  
34 }
```

linked-memory/List.hpp

```
40 /**
41 * Finds and returns the ListNode containing `data` in the
42 * List or `nullptr` if the data is not found.
43 */
44 template <typename T>
45 typename List<T>::ListNode *List<T>::_find(const T & data) {
46     ListNode *thru = head_;
47     while (thru != nullptr) {
48         if (thru->data == data) { return thru; }
49     }
50
51     return nullptr;
52 }
```

Find Data in a Sorted Array

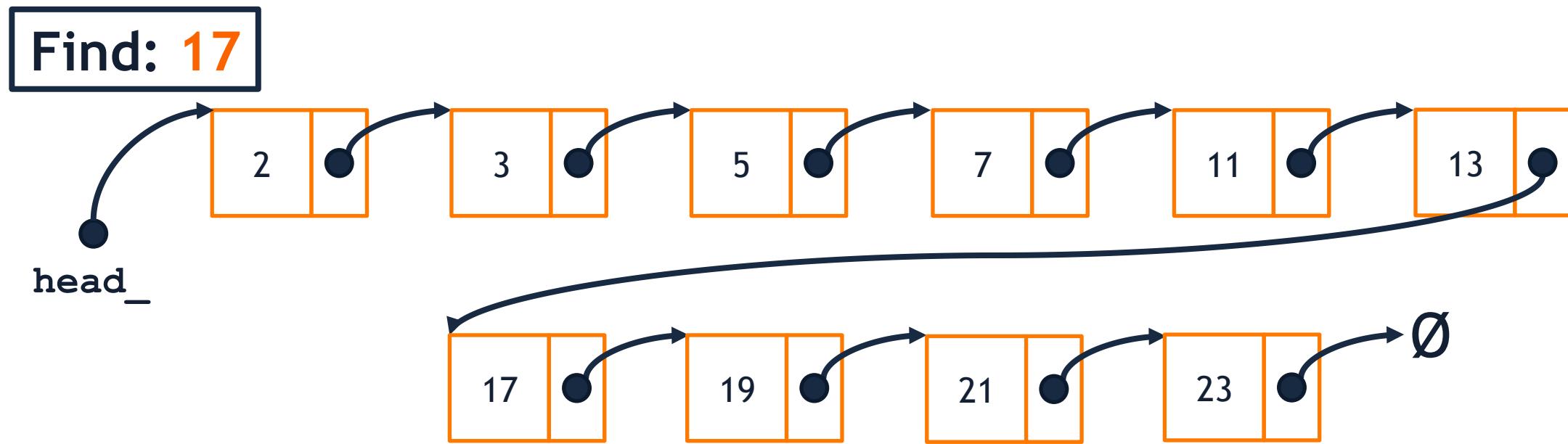
Objective: Given data, find the location of that data in the collection.

Find: 17

2	3	5	7	11	13	17	19	21	23
Index: [0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

Find Data in a Sorted Array

Objective: Given data, find the location of that data in the collection.



Find Data

Objective: Given data, find the location of that data in the collection.

Unsorted Array:	$O(n)$ Requires searching up to the full array, one-by-one, to find a match
Sorted Array:	$O(\lg(n))$ Using binary search, the run-time on an array becomes logarithmic
All Lists:	$O(n)$ Requires searching up to the full list, one ListNode at a time, to find a match - even if the list is sorted

Insert After

Objective: Given an element (ListNode or index), insert a new element immediately afterwards.

Insert After

Objective: Given an element (array index), insert a new element immediately afterwards.

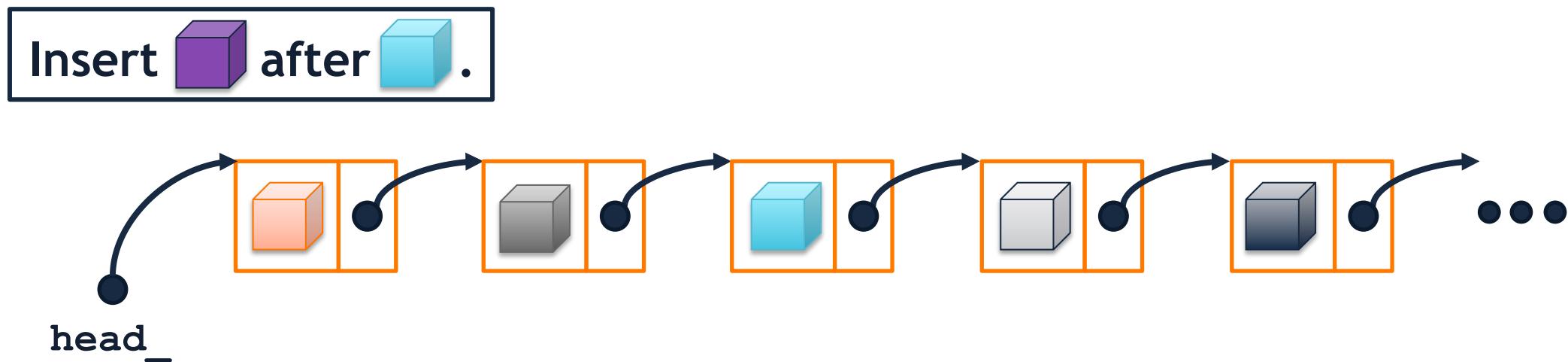
Insert  after .



Index: [0] [1] [2] [3] [4] [5]

Insert After

Objective: Given an element (ListNode), insert a new element immediately afterwards.



Insert After

Objective: Given an element (ListNode or index), insert a new element immediately afterwards.

Array:	O(n) Requires copying up to half of the array to make space for the new element
List:	O(1) Fixed constant time to add a new element after finding the previous element

Delete After

Objective: Given an element (ListNode or index), delete the element immediately afterwards.

Array:	O(n) Requires copying up to half of the array to remove the blank space
List:	O(1) Fixed constant time to remove a ListNode and repair the list

Array and List Operations

- Arrays and lists are both ordered collections that have complex tradeoffs between run-time and flexibility.
- We will build data structures using these primitive structures.

