Learning Objectives

Review fundamentals of array list

Introduce array list implementations

Consider extensions to lists
List Implementations

1. Linked List

head

```
C -> S -> 2 -> 2 -> 5 -> None
```

2. Array List

```
C | S | 2 | 2 | 5 |
```
Array List
#pragma once

template <typename T>
class List {
public:
    /* --- */
...  
    /* --- */
private:
    T *data_; 
    T *size; 
    T *capacity; 
    /* --- */
};

/* --- */
Array List: [ ]

C S 2 2 5

[ ]
Array List: `insertAtFront(data)`

```
C S 2 2 5
```
Array List: `insert(data, index)`

```
C  S  2  2  5
```
Array List: `remove(index)`

```
| C | S | 2 | 2 | 5 |
```

Array List: `insert(data, index)`
Resize Strategy: +2 elements every time
Resize Strategy: +2 elements every time
Resize Strategy: x2 elements every time
Resize Strategy: x2 elements every time
<table>
<thead>
<tr>
<th></th>
<th>Singly Linked List</th>
<th>Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look up arbitrary location</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert after given element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove after given element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert at arbitrary location</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove at arbitrary location</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search for an input value</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Thinking critically about lists: tradeoffs

The implementations shown are foundational.

Can we make our lists better at some things? What is the cost?
Thinking critically about lists: tradeoffs

Getting the size of a linked list has a Big O of:
Thinking critically about lists: tradeoffs

- Diagram of linked list with head pointers and node values.
- Two separate diagrams with arrows indicating the direction of the list.

1. Head of list 1: [4, 2, 5, 3, 1]
2. Head of list 2: [4, 2, 5, 3, 1]
Thinking critically about lists: tradeoffs

<table>
<thead>
<tr>
<th>2</th>
<th>7</th>
<th>5</th>
<th>9</th>
<th>7</th>
<th>14</th>
<th>1</th>
<th>0</th>
<th>8</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>14</td>
</tr>
</tbody>
</table>
Thinking critically about lists: tradeoffs
Thinking critically about lists: tradeoffs

When we discuss data structures, consider how they can be modified or improved!

**Next time:** Can we make a 'list' that is $O(1)$ to insert and remove? What is our tradeoff in doing so?