Data Structures
Array Lists
CS 225
September 1, 2023
Brad Solomon & G Carl Evans

No class on Monday Sept 4
Learning Objectives

Review fundamentals of array list

Introduce array list implementations

Consider extensions to lists
List Implementations

1. Linked List

- Singly linked list

2. Array List

- Continuous memory allocation
Array List

1) Pointer to array (location)
2) Size - current # of items (unsigned int)
3) Capacity - max # of items (unsigned int)

- Size = \( y \)
- Capacity = \( z \)
- T* size
- T* capacity
- 8 bytes
- Size++
- One address post the end of array
- C faster
- Doesn't hurt us
#pragma once

template <typename T>
class List {
public:
    /* --- */
private:
    T *data_; = 0
    T *size; = 5
    T *capacity; = 6
    /* --- */
};
Array List: [ ]

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

Every object: find(data)

\[
3 = i \text{ (index)}
\]

Find address of index @ data:

\[
\text{data + index - sizeof(T)}
\]

\[
0 + 3 \times 4 = 12
\]

\[
5 \times 0 = 0
\]

\[
C_p h: r_a + 12
\]

\[
\text{random access + index}
\]

\[
O(1)
\]
Array List: `insertAtFront(data)`

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

Time Complexity: $O(n)$
Array List: `insert(data, index)`

```
C  S  2  2  5
```

- To insert `B` at index 0, all items to the right of the insertion point need to be moved right by one position.
- `O(n)` time complexity for `n` items.
  - `n` items are being moved.
Array List: `remove(index)`

- `C S 2 2 5`
- Memory layout:
  - `C S 2 2 5` (original array)
  - After removal:
    - `C S 2 5` (reduced array)

- Comments:
  - "Tombstoning" - save work until later
  - Example: `Array: 1 2 3 4 5 6 7`
Array List: \texttt{pushback(data)}

| C | S | 2 | 2 | 5 | D |  |

- \texttt{insert at back}
- \texttt{push-back()}  
- \texttt{remove(size)}
- \texttt{--size,}

\[\text{Size} = \text{data} ;\]
\[\text{Size} \quad + + ;\]

\[O(1)\]
Array List: `insert(data, index)`

- Time complexity: \( O(n) \) to insert
Resize Strategy: +2 elements every time

1) How many copies per realloc?
   For iteration i, 2i reallocs

2) How many reallocs? (N objects total)
   \[ k = \# \text{reallocs} = \frac{N}{2} \]

Total # of copies
\[ \sum 2i = k(k+1) \]
   \[ i = 1 \]
   \[ k^2 + k \]

\[ \frac{k^2 + k}{2} = \frac{N^2 + 2N}{4} \]
   for \( N \) insertions
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 <strong>arbitrary</strong> location</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert after <strong>given</strong> element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove after <strong>given</strong> element</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Insert at <strong>arbitrary</strong> location</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove at <strong>arbitrary</strong> location</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search for an input <strong>value</strong></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:

![Diagram of a linked list with nodes containing the values C, S, 2, 7, 7, and None. The head of the list is marked with an arrow pointing to the first node containing C.]
Thinking critically about lists: tradeoffs
Thinking critically about lists: tradeoffs

```
2  7  5  9  7  14  1  0  8  3
```

```
0  1  2  3  5  7  7  8  9  14
```
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?