CS 225

#11: Stacks and Queues

February 8, 2018 · *Fagen-Ulmschneider, Zilles*

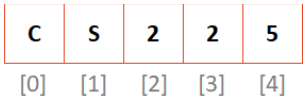
List Implementation #2: _____

List.h

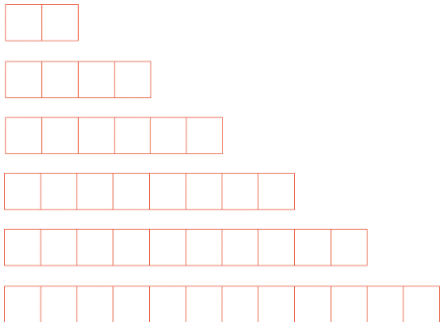
```
 1  #pragma once
 2
 3  template <typename T>
 4  class List {
 5    public:
 …       /* ... */
28    private:
29
30
31
32  };
```

Implementation Details and Analysis:

What is the running time of `insertFront()`?



➔ What is our resize strategy?

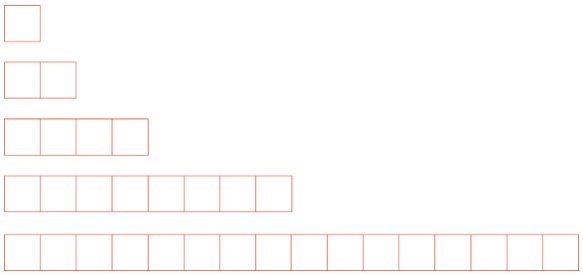


...total copies across all resizes: _____

...total number of insert operations: _____

...average (amortized) cost of copies per insert: _____

Array Resize Strategy #2:



...total copies across all resizes: _____

...total number of insert operations: _____

...average (amortized) cost of copies per insert: _____

Running Time:

	Singly Linked List	Array
Insert/Remove at **front**		
Insert after a **given** element		
Remove after a **given** element		
Insert at **arbitrary** location		
Remove at **arbitrary** location		

A List implementation in std

- `std::vector` implements a list with dynamic growth
- `#include <vector>` to use it!
- Documentation widely available, including on CBTF exams

Stack ADT

Function Name	Purpose

Queue ADT

Function Name	Purpose

Stack and Queue Implementations

Stack.h

```
 1  #pragma once
 2
 3  #include <vector>
 4
 5  template <typename T>
 6  class Stack {
 7    public:
 8      void push(T & t);
 9      T & pop();
10      bool isEmpty();
11
12    private:
13      std::vector<T> list_;
14  };
15
16  #include "Stack.hpp"
```

Stack.hpp

```
 3  template <typename T>
 4  void Stack<T>::push(const T & t) {
 5    list_.push_back(t);
 6  }
 7
 8  template <typename T>
 9  const T & Stack<T>::pop() {
10    const T & data = list_.back();
11    list_.pop_back();
12    return data;
13  }
```

Three designs for data storage in data structures:

1. T & data

2. T * data

3. T data

Implication of Design

	Storage by Reference	Storage by Pointer	Storage by Value
Lifecycle management of data?			
Possible to insert NULL?			
External data manipulation?			
Speed			

CS 225 – Things To Be Doing:

1. Programming Exam A starts Feb. 14 *(next Thursday)*
2. MP2 due Feb. 11 *(next Monday);* MP3 released Tuesday
3. lab_inheritance due Sunday
4. Daily POTDs