CS 225

#12: Lists, Stacks, Queues, and Design

February 22, 2021 · *G Carl Evans*

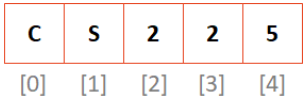
List Implementation #2: _____

Alternate List.h

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

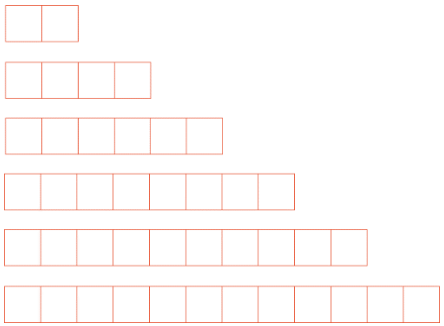
Implementation Details and Analysis:

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



➔ What is our resize strategy?

Array Resize Strategy #1:

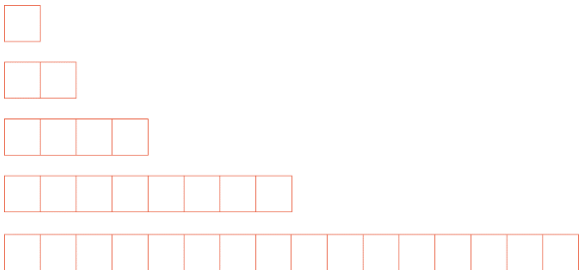


...total copies across all resizes: _____

...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		

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

Example 1

```
Queue<int> q;
q.enqueue(3);
q.enqueue(8);
q.enqueue(4);
q.dequeue();
q.enqueue(7);
q.dequeue();
q.dequeue();
q.enqueue(2);
q.enqueue(1);
q.enqueue(3);
q.enqueue(5);
q.dequeue();
q.enqueue(9);
```

Example 2

```
Queue<char> q;
q.enqueue('m');
q.enqueue('o');
q.enqueue('n');
…
q.enqueue('d');
q.enqueue('a');
q.enqueue('y');
q.enqueue('i');
q.enqueue('s');
q.dequeue();
q.enqueue('h');
q.enqueue('a');
```

Accessing Every Element in Our List / Queue / [Anything]
Suppose we want to look through every element in our data structure. What if we don't know what our data structure even looks like?

CS 225 – Things To Be Doing:

1. mp_stickers due today
2. Daily POTDs