CS 225

Data Structures

January 29 – Amortized Analysis

G Carl Evans

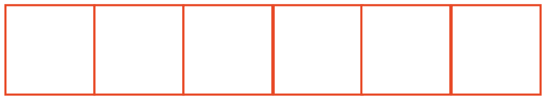
List.h

```
#pragma once

template <typename T>
class List {
public:
        void push_back(const T &data);
private:
        T *storage_;
        T *insert_;
        T *cap_;

        void resize_();

};
```



Array Implementation

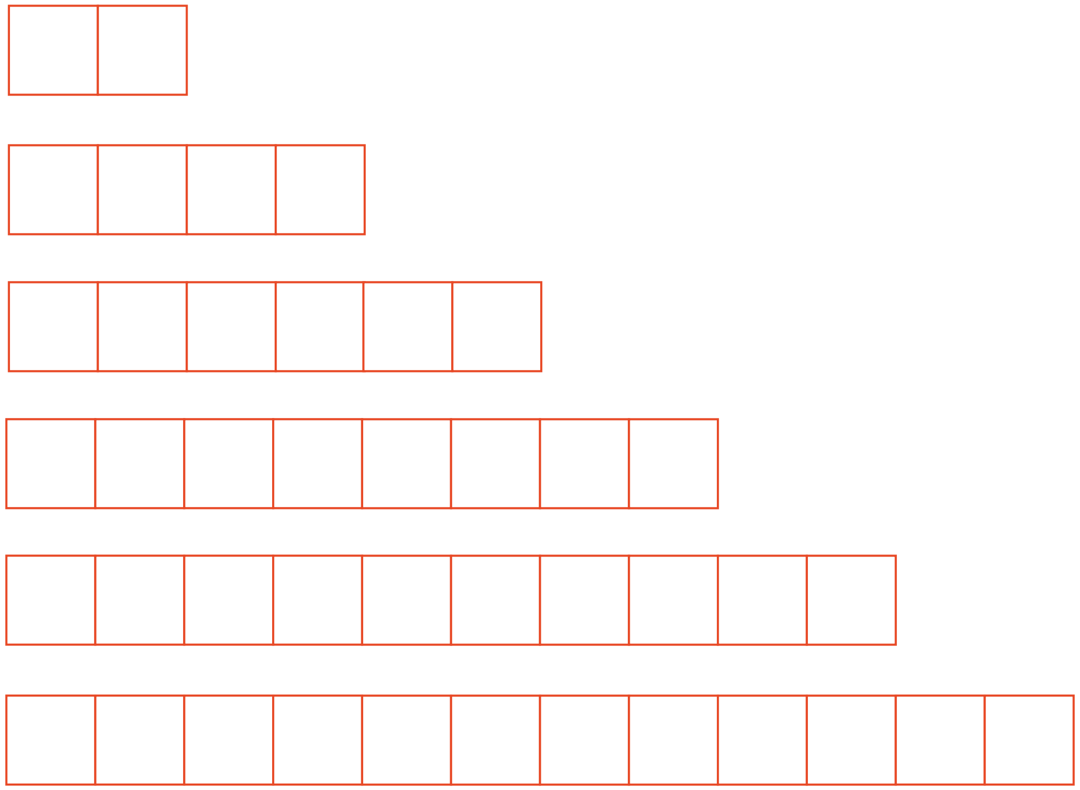
```
template <typename T>
Void List<T>::push_back(const T & data){
  if(insert_ == cap_){
    resize();
  }

  *insert_ = data;
  ++insert_;
};

```

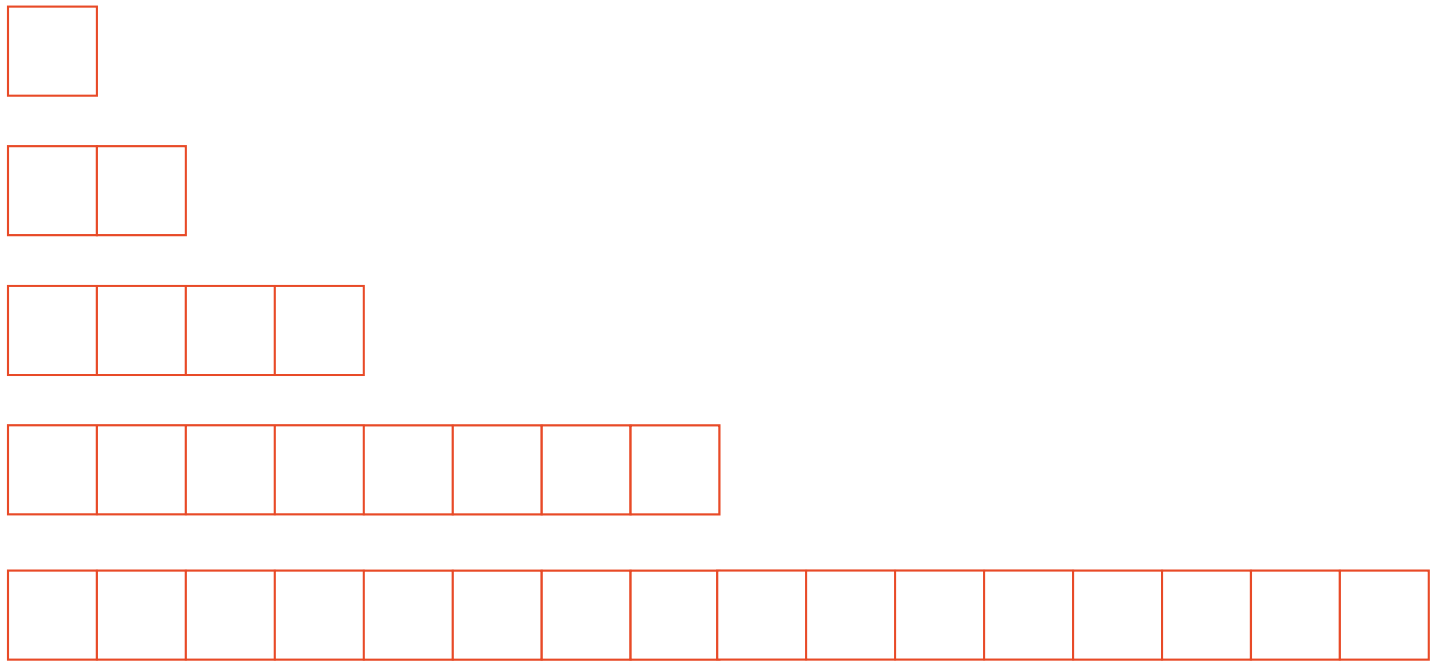
Amortized Analysis

Resize Strategy: +2 elements every time



Resize Strategy: +2 elements every time

Resize Strategy: x2 elements every time



Resize Strategy: x2 elements every time

List Implementations

	Singly Linked List	Array
Insert/Remove at front/back		
Insert/Remove given a pointer		
Insert/Remove at arbitrary index		
Find a value in a list		
Fina a value in a sorted list		

Queue ADT

- [Order]:

- [Implementation]:

- [Runtime]:

Stack ADT

- [Order]:
- [Implementation]:
- [Runtime]:

Queue.h

```
#pragma once

template <typename T>
class Queue {
  public:
    void enqueue(T e);
    T dequeue();
    bool isEmpty();

  private:
    T *items_;
    unsigned capacity_;
    unsigned size_;
};
```

What type of implementation is this Queue?

How is the data stored on this Queue?

Queue.h

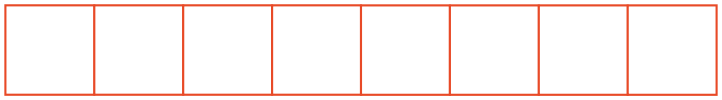
```
#pragma once

template <typename T>
class Queue {
  public:
    void enqueue(T e);
    T dequeue();
    bool isEmpty();

  private:
    T *items_;
    unsigned capacity_;
    unsigned size_;
};
```

What type of implementation is this Queue?

How is the data stored on this Queue?



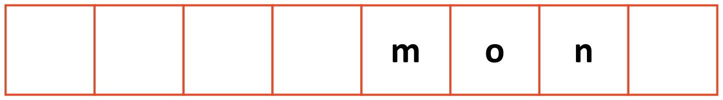
```
Queue<int> q;
q.enqueue(3);
q.enqueue(8);
q.enqueue(4);
q.dequeue();
q.enqueue(7);
q.dequeue();
q.dequeue();
q.enqueue(2);
q.enqueue(1);
q.enqueue(3);
q.enqueue(5);
q.dequeue();
q.enqueue(9);
```

Queue.h

```cpp
#pragma once

template <typename T>
class Queue {
  public:
    void enqueue(T e);
    T dequeue();
    bool isEmpty();

  private:
    T *items_;
    unsigned capacity_;
    unsigned size_;
};
```



```cpp
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);
```