

Topic 3

Basic Data Structures continued

CSE1303 Part A

Data Structures and Algorithms

Basic Data Structures



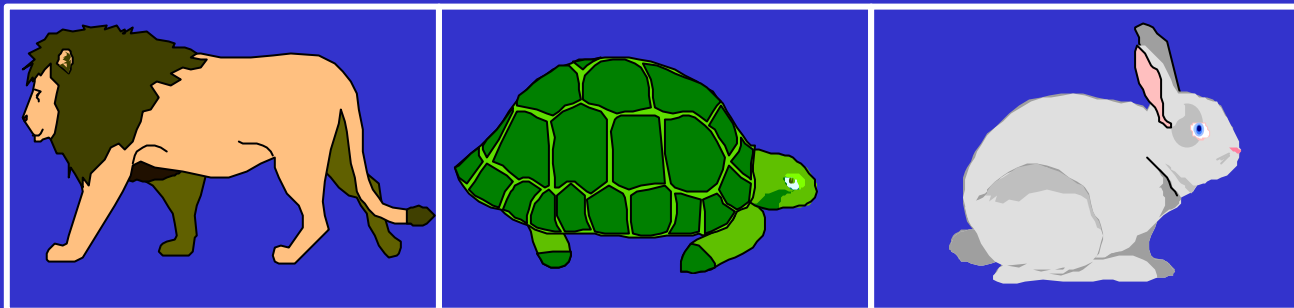
- Stacks
- Queues
- Lists

Overview

- What is a Queue?
- Queue Operations.
- Applications.
- Linear Implementation.
- Circular Implementation.

Before

Append

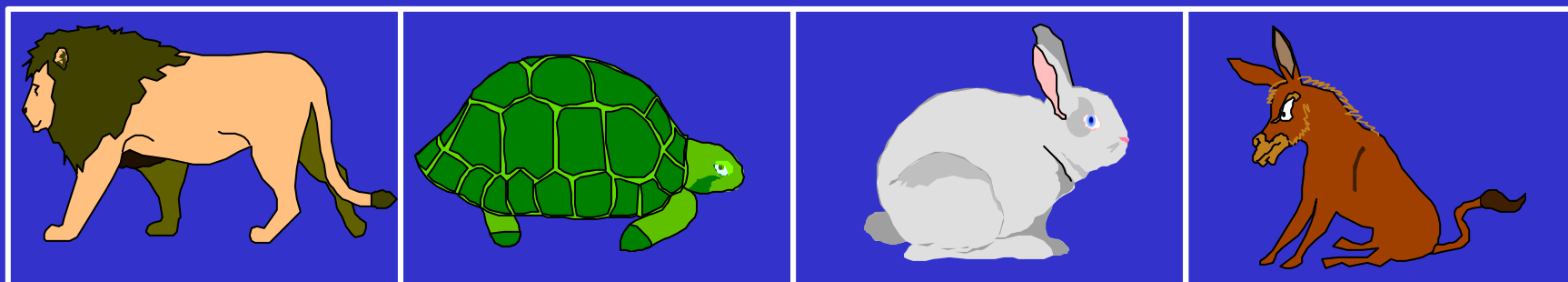


Front



Rear

After



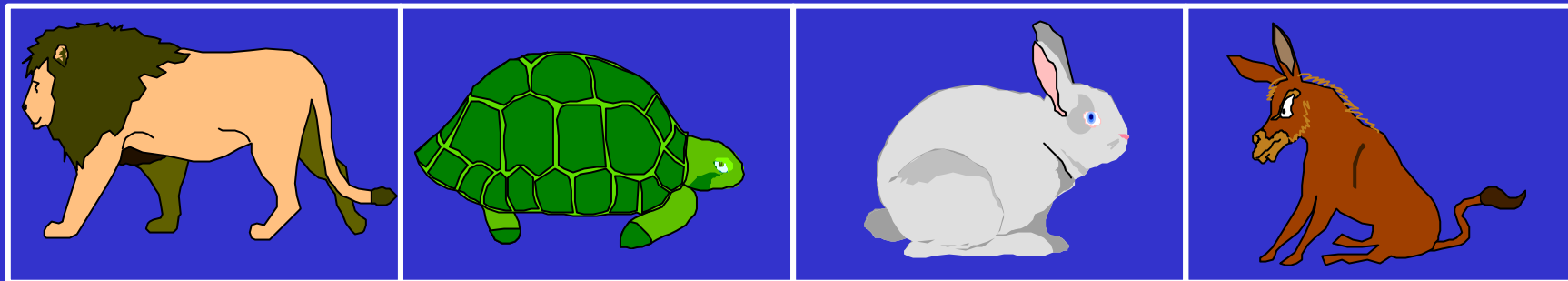
Front



Rear

Serve

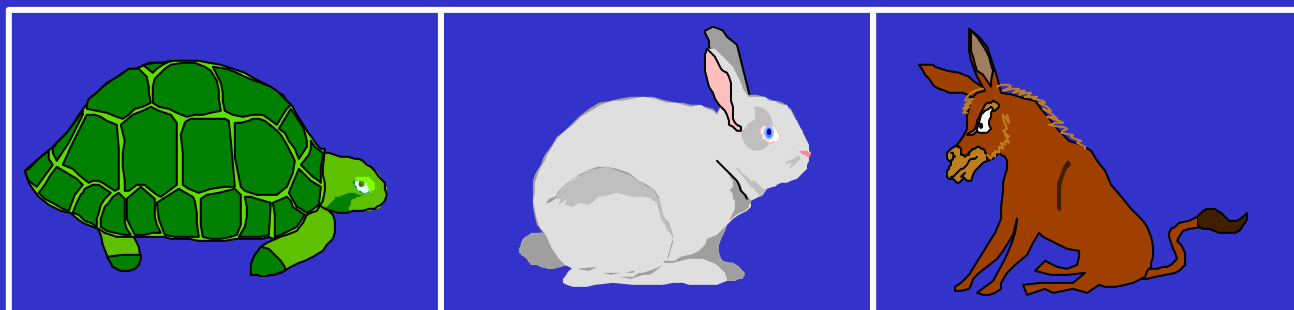
Before



↑
Front

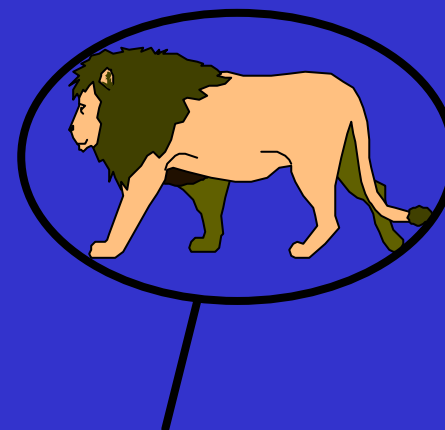
↑
Rear

After



↑
Front

↑
Rear



This comes off the queue

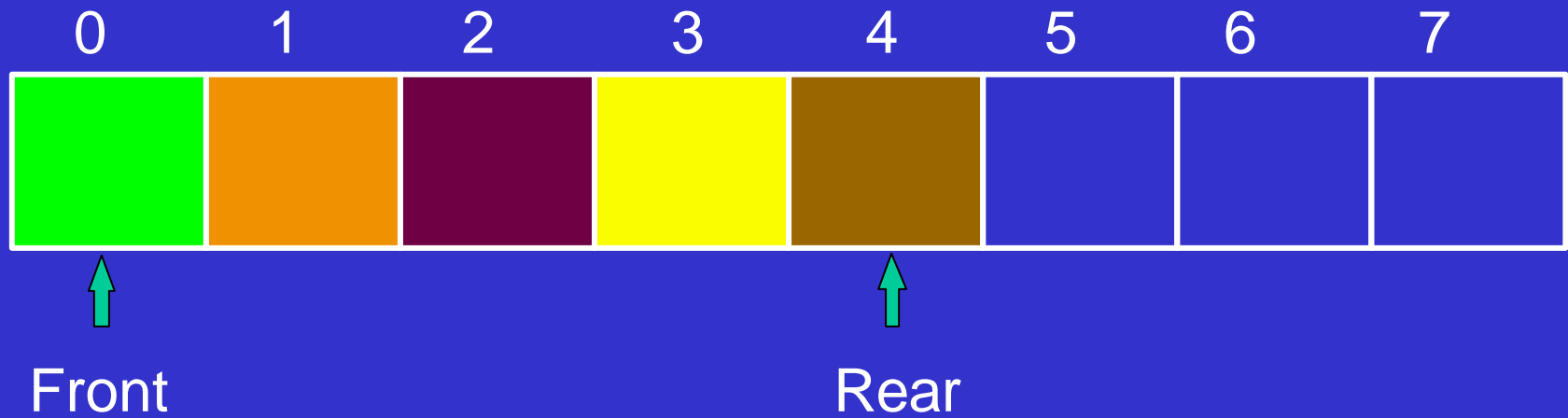
Operations

- Initialize the queue.
- Append an item to the rear of the queue.
- Serve an item from the front of the queue.
- Is the queue empty?
- Is the queue full?
- What size is the queue?

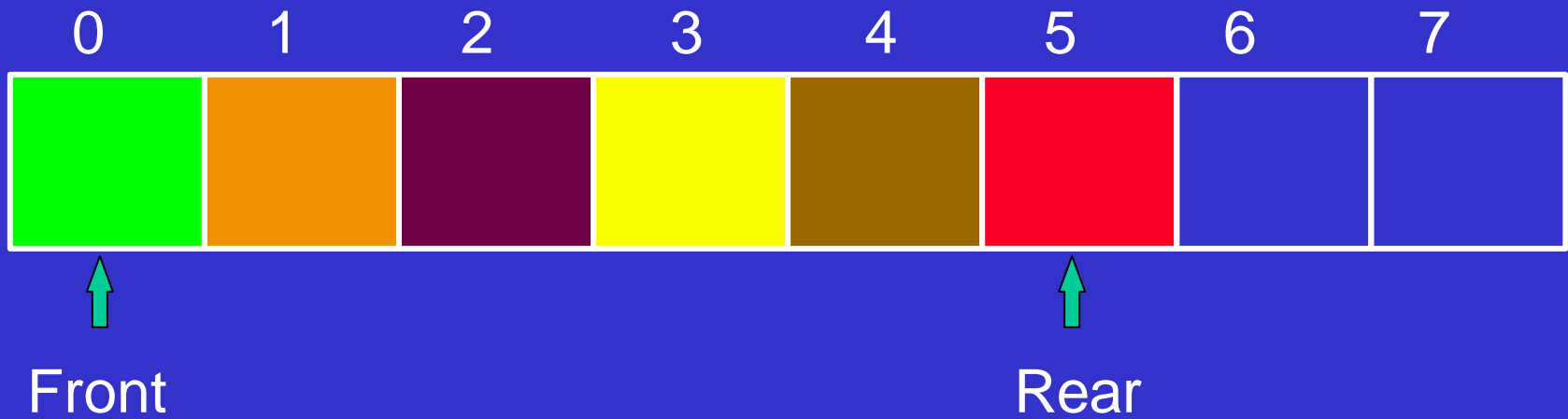
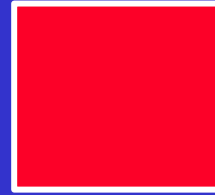
Applications

- In operating systems, e.g. printer queues, process queues, etc.
- Simulation programs.
- Algorithms.

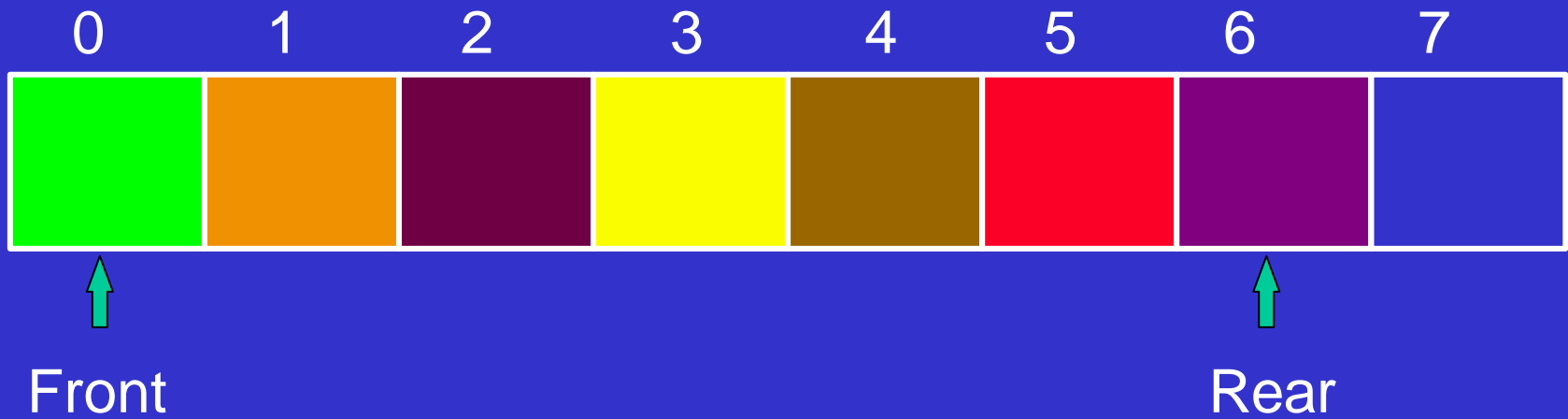
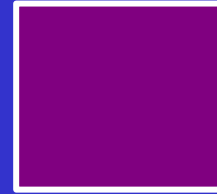
Linear Implementation



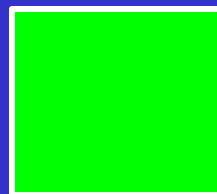
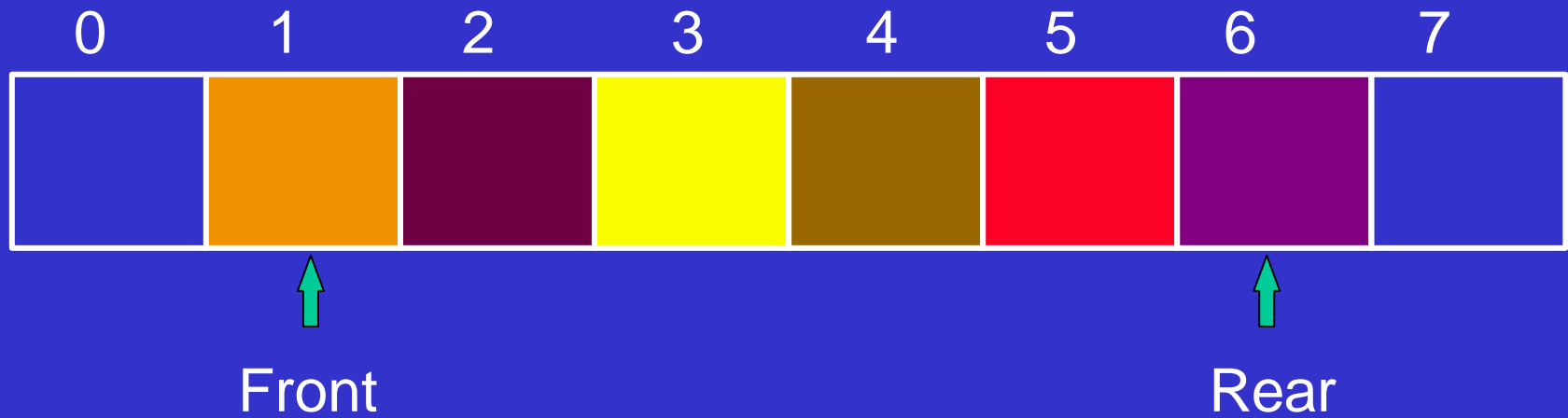
Append



Append

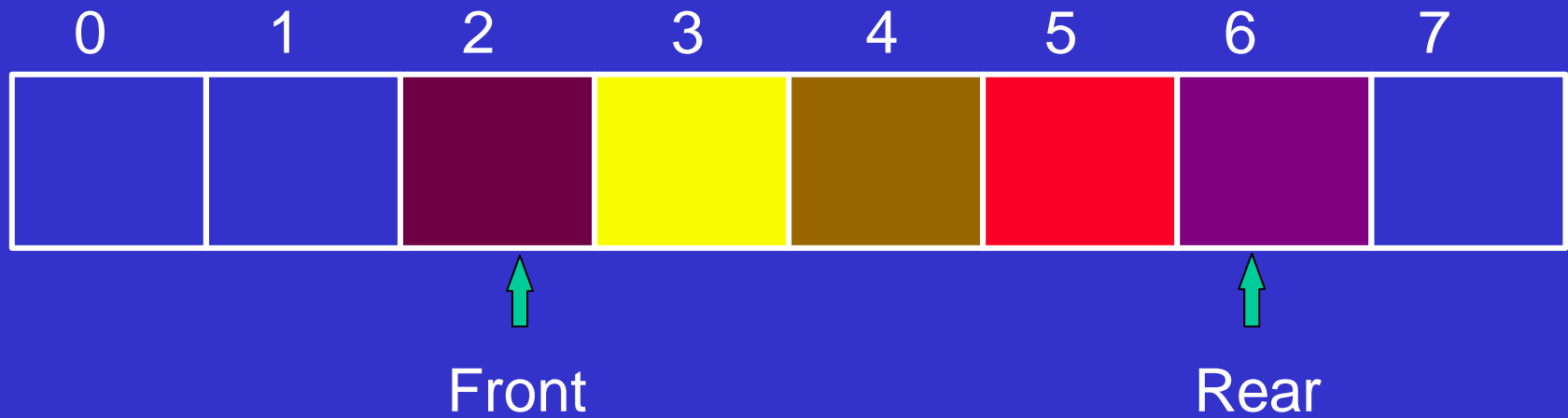


Serve



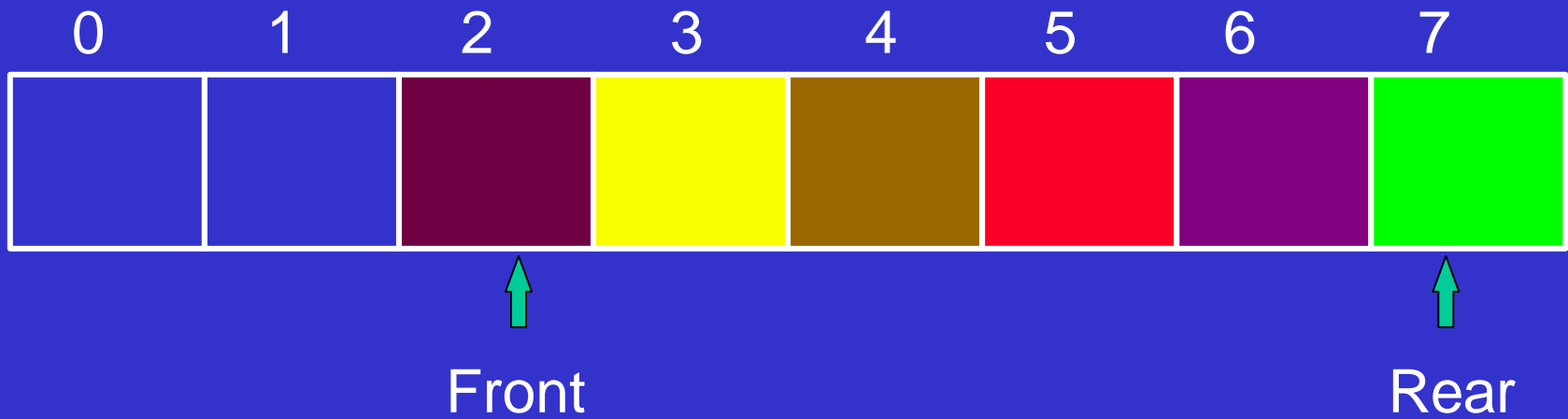
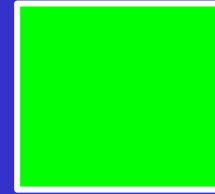
**This comes off the
queue**

Serve

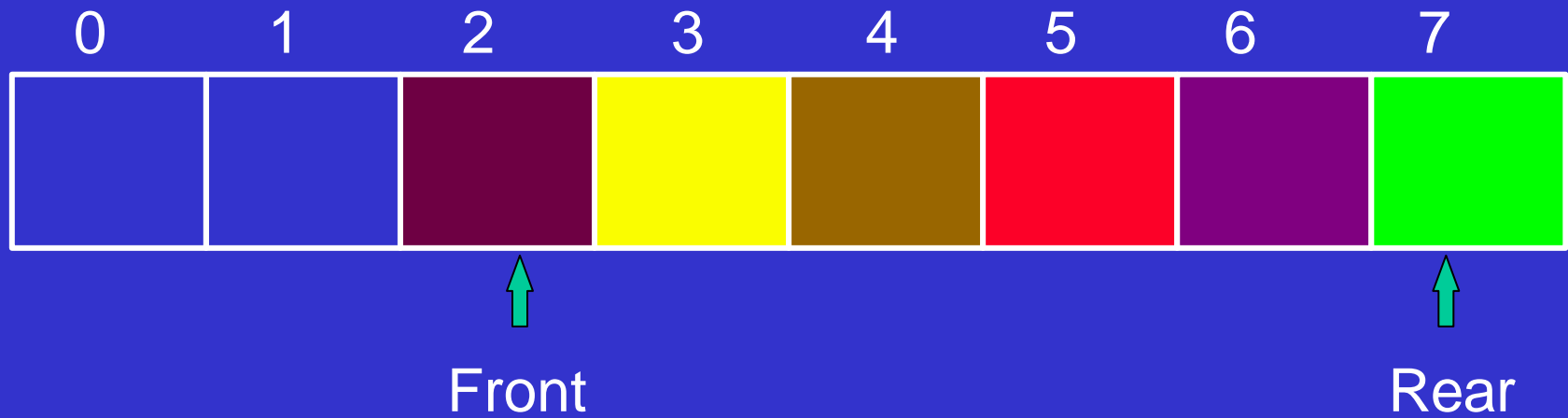
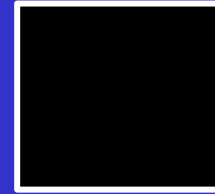


This comes off the queue

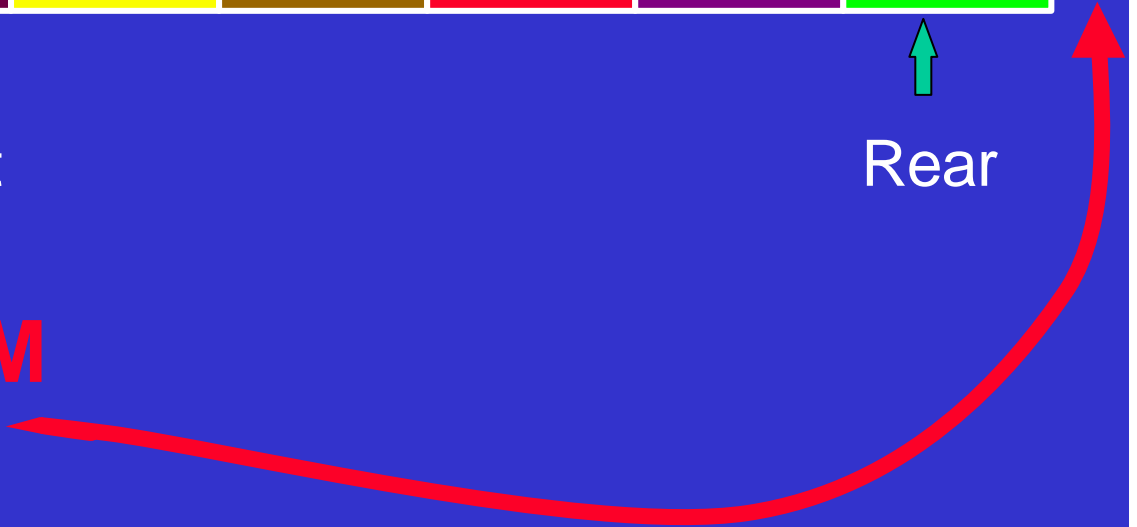
Append



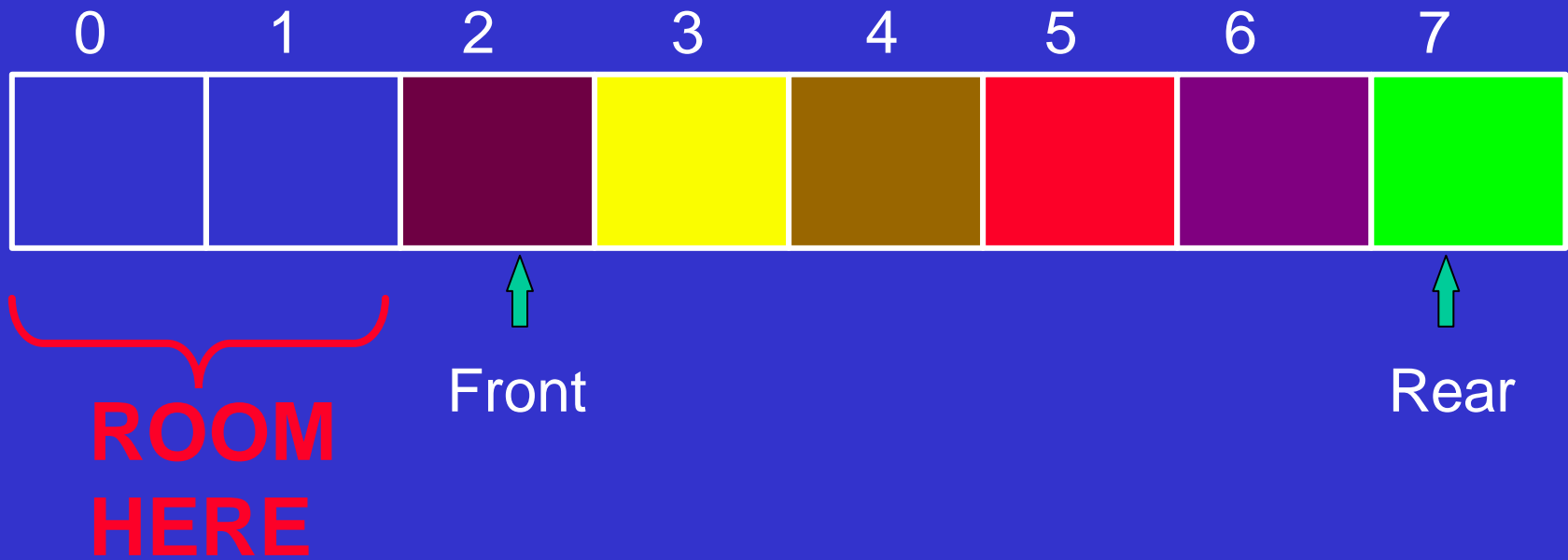
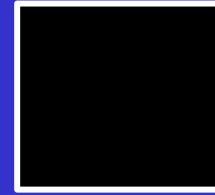
Append



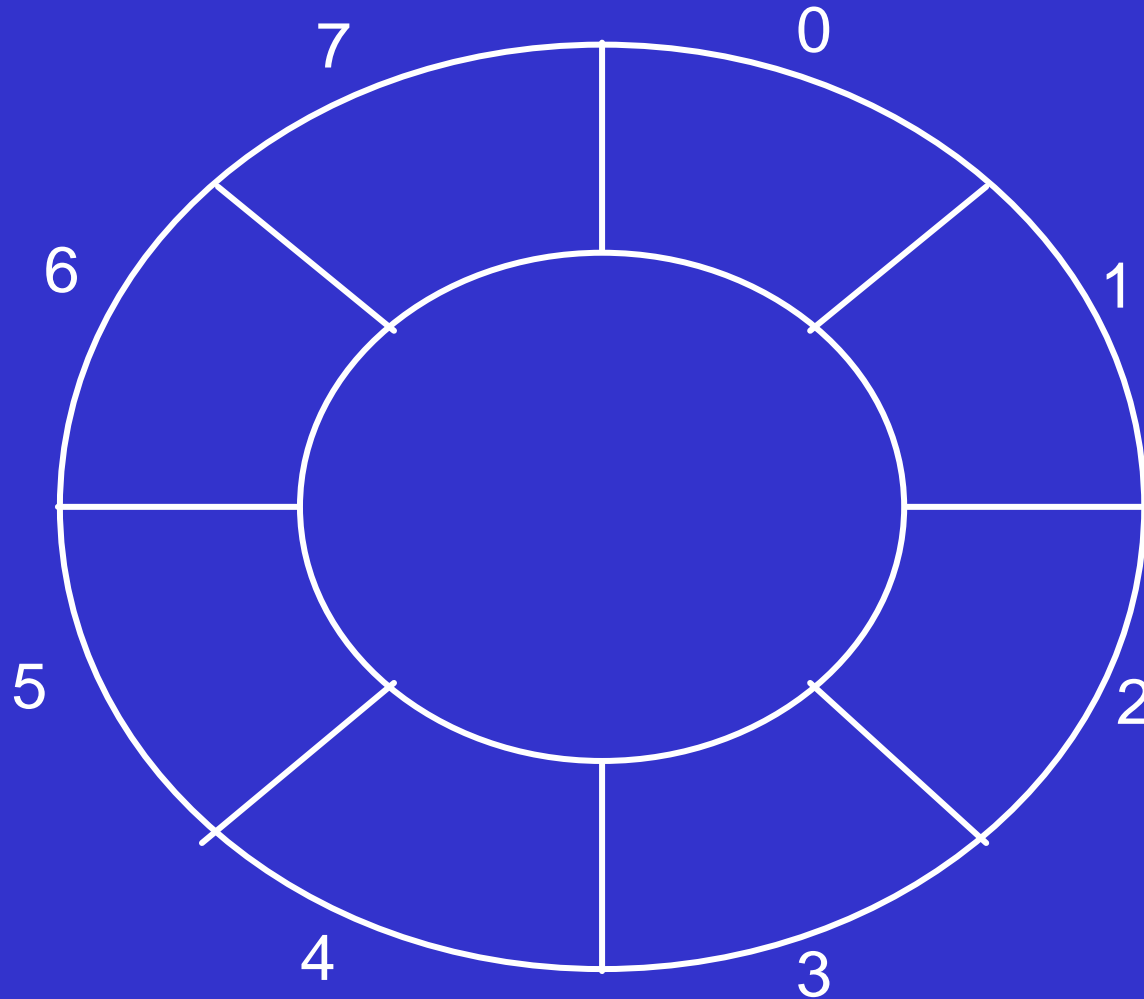
**NO ROOM
HERE**



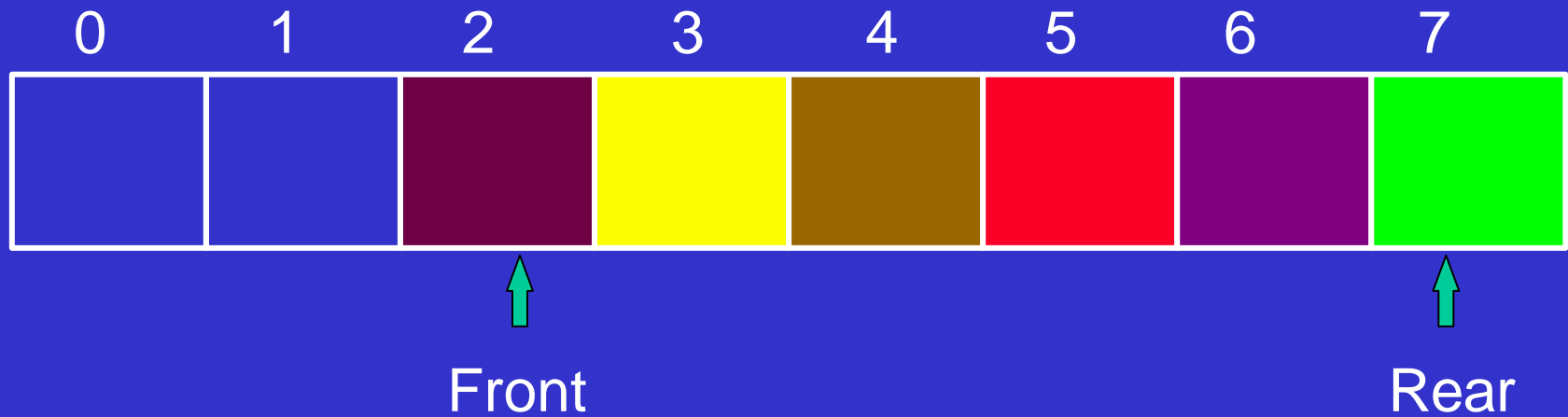
Append



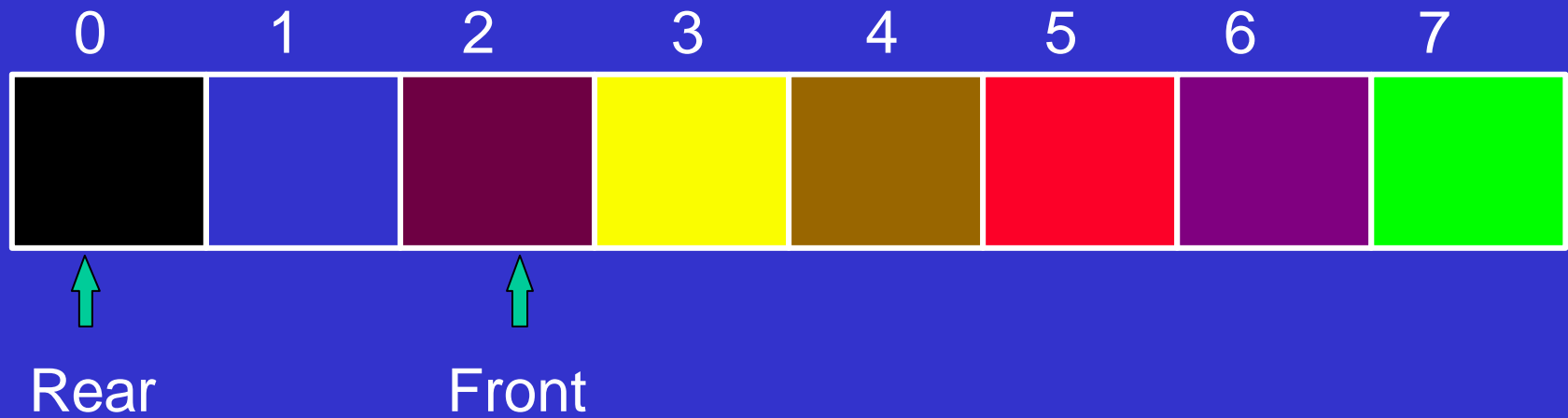
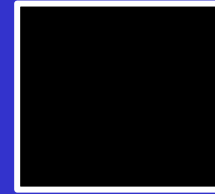
Circular Implementation



Circular Implementation



Append



```
#ifndef QUEUE.H
#define QUEUE.H
#include <stdbool.h>
#define MAXQUEUE 20

struct QueueRec
{
    int    count;
    int    front;
    int    rear;
    float  entry[MAXQUEUE];
};

typedef struct QueueRec Queue;

void initializeQueue(Queue* queuePtr);
bool queueEmpty(const Queue* queuePtr);
bool queueFull(const Queue* queuePtr);
void append(Queue* queuePtr, float item);
float serve(Queue* queuePtr);

#endif
```



```
#include <stdio.h>
#include <stdlib.h>
#include "queue.h"
```

```
void initializeQueue(Queue* queuePtr)
{
    queuePtr -> count = 0;
    queuePtr -> front = 0;
    queuePtr -> rear = MAXQUEUE-1;
}
```

queuePtr:

addr of Queue



Queue:

count: 0

front: 0

rear: 19

entry:

·
·
·

```
bool
queueEmpty(const Queue* queuePtr)
{
    if (queuePtr->count <= 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

```
bool
queueFull(Queue* queuePtr)
{
    if (queuePtr->count >= MAXQUEUE)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

```
void
append(Queue* queuePtr, float item)
{
    if (queueFull(queuePtr))
    {
        fprintf(stderr, "Queue is full\n");
        exit(1);
    }
    else
    {
        queuePtr->rear++;
        if (queuePtr->rear == MAXQUEUE)
        {
            queuePtr->rear = 0;
        }
        queuePtr->entry[queuePtr->rear] = item;
        queuePtr->count++;
    }
}
```

```
float serve(Queue* queuePtr)
{
    float item;

    if (queueEmpty(queuePtr))
    {
        fprintf(stderr, "Queue is empty\n");
        exit(1);
    }
    else {
        item = queuePtr->entry[queuePtr->front];
        queuePtr->front++;
        if (queuePtr->front == MAXQUEUE)
        {
            queuePtr->front = 0;
        }
        queuePtr->count--;
    }
    return item;
}
```

Revision

- Queue
- Main Operations
- Implementation.

Preparation

- Kruse 4.5, 4.6, 4.8
- Deitel & Deitel (2e) 12.1, 12.2, 12.4