# Day's Goals

Thursday, February 15, 2018 4:42 PM

- Destructor for SLL stack
- Recap: Stacks vs Queues
- Queue: the array approach
  linear vs circular

## Stacks vs. Queues

Both are complex data structures that allow for specific order in which operations on their data can be performed.

### Stack

- can only add to top "Push"
- can only remove from top "pop"

LIFO

### Queue

- can only add at the tail "enqueue"
- can only remove from head "dequeue"

FIFO

can implement

- array

using array or LL

- array: linear or circular

## I Que Linear (array)

- simplest approach
- "head" stays fixed
- tail tracks end of Q

dequeuing is inefficient

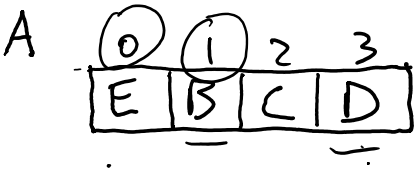
↳ worst case when Q is full

→ O(n)

## II Circular Array Q

Allow for both head and tail to shift when dequeuing and enqueuing

e.g. max Q size = 4




q.enque(A)

q.enque(B)

q.enque(C)

head = 1

tail = 1

q.enque(C)
q.deque() ←
q.enque(D)
→ q.enque(E)

If head and tail are equal, how do you tell if Q is full or empty?

keep count of Q size.

maxQsize == queSize

# Q ADP (generic)

private:
- head
- tail
- queSize // count
- data

public:
- init // constructor
- isEmpty()
- isFull()
- enque(newItem)

item deque
disp()

## Implementation: Circular Array

```
#define MAXSIZE 5

class QueArrCir
{
    private:
        int head, tail, queSize
        string a[MAXSIZE] // data

    public:
        QueArrCir // constructor
        {
            head=0;
            tail = 0;
            queSize=0;
        }
        bool isEmpty()
        {
            queSize ==0
            return true;
        }
```

```
bool isFull()
{
   queSize == MAXSIZE
   return true
}

void enque (string item)
{
   // when enquing we only deal
   //(w/ the tail

   if (!isFull())
   {
      a[tail] = item;
      if (tail == MAXSIZE-1)
         tail = 0
      else
         tail++
   }
   else
   {
      " Que is full"
   }
```

tail was 4