Queues

Johan Montelius

KTH

HT22

a stack

- Operations: push() and pop(),
- First value pushed is the last item popped - *last in first out*

a stack

- Operations: push() and pop(),
- First value pushed is the last item popped - *last in first out*

Perfect when guiding an execution of a program.

when a stack fails

when a stack fails

when a stack fails

Please push yourself on the stack and it will soon be your turn.

the queue

the queue

First in first out.

the queue

First in first out.

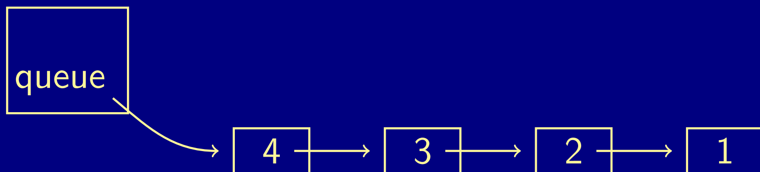
- enqueue(): Add item to the queue.

the queue

First in first out.

- enqueue(): Add item to the queue.
- dequeue() : Remove item that has been the longest in the queue.
- empty() : Is the queue empty?

a linked list implementation



a linked list implementation

```
public class Queue()<T> {

  Node queue;

  private class Node {
    T item;
    Node next;

    private Node(T itm, Node nxt) {
      this.item = itm;
      this.next = nxt;
    }
  }

  :
```

a linked list implementation

```
  :
public enqueue(T itm) {
    this.queue = Node(itm, queue);
}
```

a linked list implementation

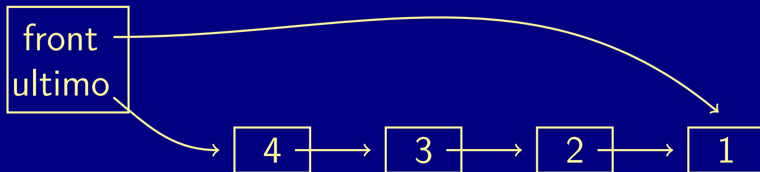
```
public dequeue() {
  if (this.queue == null)
     return null;
  Node prv = null;
  Node nxt = this.queue;
  while (nxt.next != null) {
    prv = nxt;
    nxt = nxt.next;
  }
  if (prv == null)
    this.queue = null;
  else
    prv.next = null;
  return nxt.item;
```

a linked list implementation

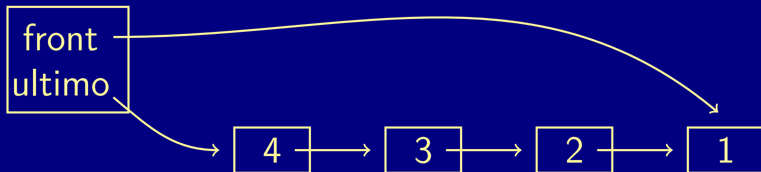
a linked list implementation

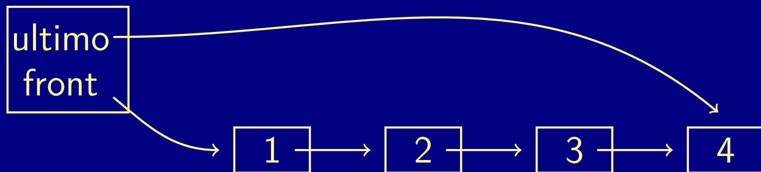
There must be a simpler way.

a linked list implementation



a linked list implementation





a linked list implementation

```
public class Queue()<T> {

  Node front;
  Node ultimo;
  :
```

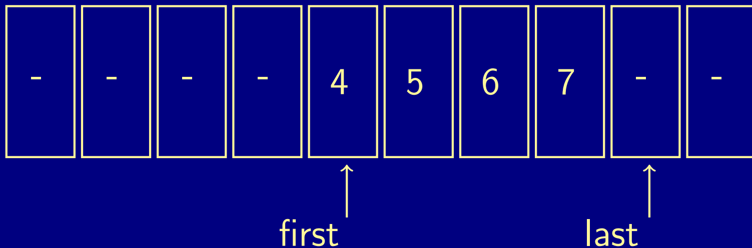
let's try an array

```
public class QueueArray<T> {

    T[] queue;

    int first = 0;
    int last = 0;

    int size =  10;

    public QueueArray() {
        this.queue = (T[]) new Object[this.size];
    }
    :
```

the empty array

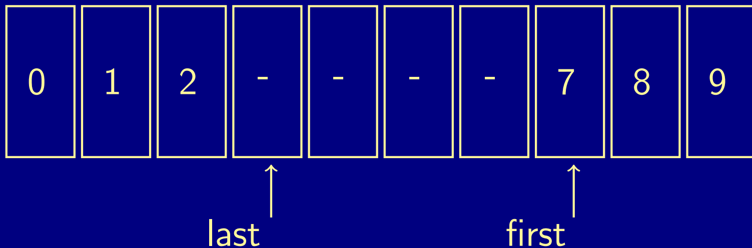
```
public boolean empty() {
    return (first == last);
}
```

when first <= last



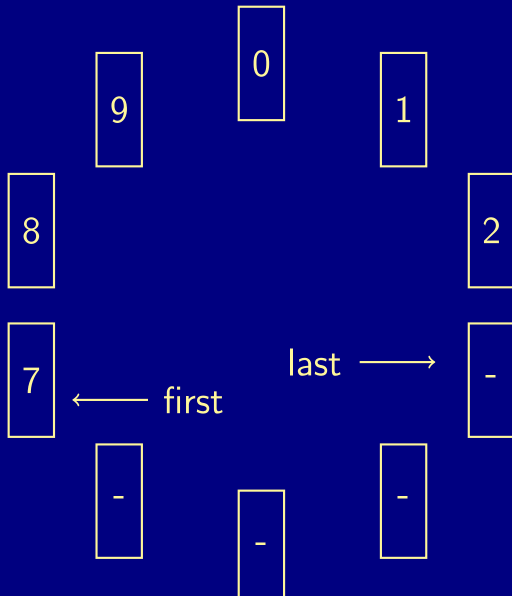
size = 10

when last < first



size = 10

modulo the size



dynamic size

- What should we do if the queue is full?

dynamic size

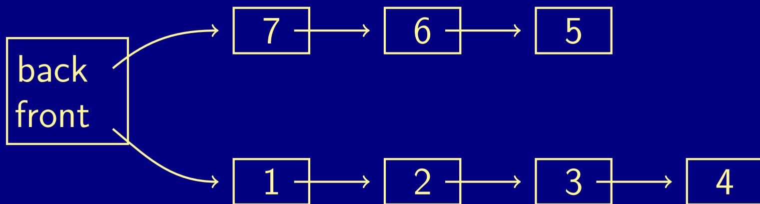
- What should we do if the queue is full?
- Should we shrink the size of the array?

an alternative implementation

an alternative implementation

An implementation often used in functional programming languages.

an alternative implementation



a linked list, almost as before

```
public class Queue()<T> {

  Node back;
  Node front;

  private class Node {
    T item;
    Node next;

    private node(T itm, Node nxt) {
      this.item = itm;
      this.next = nxt;
    }
  }
```

a linked list, almost as before

```
public void enqueue(T itm) {
  back = Node(itm, back);
}

public T dequeue() {
  T itm = front.item;
  front = front.next;
  return itm;
}
:
```

a linked list, almost as before

```
public void enqueue(T itm) {
  back = Node(itm, back);
}

public T dequeue() {
  T itm = front.item;
  front = front.next;
  return itm;
}
:
```

Perfect, ... just one small detail.

when to use queues

- Concurrent programs: handle requests in the order that they arrive.

when to use queues

- Concurrent programs: handle requests in the order that they arrive.
- Breadth-first traversal of a tree, task are generated but need not be solved right away.