

STACKS | QUEUES

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Agenda

1 Stacks

2 Queues

3 Bibliography



**Centro de
Informática**
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Stack ADT

Policy: **LIFO** = last-in, first-out

Operations:

- `void clear(Stack s);`
- `void push(Stack s, E it);`
- `E pop(Stack s);`
- `E topValue(Stack s);`
- `int length(Stack s);`

Implementations based on: arrays and **linked lists**



**Centro de
Informática**
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Stack ADT implemented as a linked list

Composite type (Stack):

```

1  Link top;           // reference to the first element
2  int size;           // number of elements

```

Differently from the List implementation as a linked list: **no header node**

Algorithm: Stack `create_stack()`

```

1  s.top  $\leftarrow$  NULL;
2  s.size  $\leftarrow$  0;
3  return s;

```



**Centro de
Informática**
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Stack ADT implemented as a linked list

Algorithm: void push(Stack s, E it)

```
1 s.top ← create_link(it, s.top);
2 s.size++;
```

`push(s, 7)`, just after `create_stack()`

top → null



**Centro de
Informática**
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Stack ADT implemented as a linked list

Algorithm: void push(Stack s, E it)

```
1 s.top ← create_link(it, s.top);
2 s.size++;
```

push(s, 7), just after create_stack()



**Centro de
Informática**
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Stack ADT implemented as a linked list

Algorithm: void push(Stack s, E it)

```
1 s.top  $\leftarrow$  create_link(it, s.top);  
2 s.size++;
```

push(*s*, 7), just after create_stack()



Centro de
Informática
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Stack ADT implemented as a linked list

Algorithm: void push(Stack s, E it)

```
1  $s.top \leftarrow create\_link(it, s.top);$   
2  $s.size++;$ 
```

Better presented as follows



Centro de
Informática
UFPE



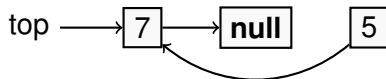
UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Stack ADT implemented as a linked list

Algorithm: void push(Stack s, E it)

- 1 $s.top \leftarrow \text{create_link}(it, s.top);$
 - 2 $s.size++;$
-

push(s, 5)



Centro de
Informática
UFPE



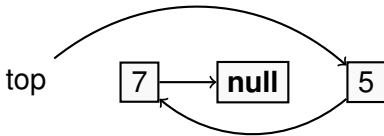
UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Stack ADT implemented as a linked list

Algorithm: void push(Stack s, E it)

```
1 s.top  $\leftarrow$  create_link(it, s.top);  
2 s.size++;
```

push(*s*, 5)

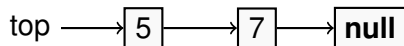


Stack ADT implemented as a linked list

Algorithm: void push(Stack s, E it)

```
1  $s.top \leftarrow create\_link(it, s.top);$   
2  $s.size++;$ 
```

Better presented as follows



Centro de
Informática
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Stack ADT implemented as a linked list

Algorithm: E pop(Stack s)

```

1 if s.top = NULL then error;
2 it ← s.top.element;
3 s.top ← s.top.next;
4 s.size--;
5 return it;

```

pop(s), just after push(s, 2)



**Centro de
Informática**
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Stack ADT implemented as a linked list

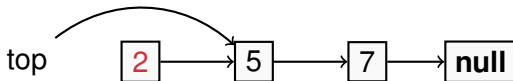
Algorithm: E pop(Stack s)

```

1 if  $s.top = NULL$  then error;
2  $it \leftarrow s.top.element$ ;
3  $s.top \leftarrow s.top.next$ ;
4  $s.size--$ ;
5 return  $it$ ;

```

$pop(s)$, just after $push(s, 2)$



**Centro de
Informática**
UFPE



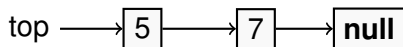
UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Stack ADT implemented as a linked list

Algorithm: E pop(Stack s)

```
1 if  $s.top = NULL$  then error;  
2  $it \leftarrow s.top.element$ ;  
3  $s.top \leftarrow s.top.next$ ;  
4  $s.size--$ ;  
5 return  $it$ ;
```

After memory deallocation (by the user or by the GC)



**Centro de
Informática**
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Agenda

1 Stacks

2 Queues

3 Bibliography



**Centro de
Informática**
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Queue ADT

Policy: **FIFO** = first-in, first-out

Operations:

- void clear(Queue q);
- void enqueue(Queue q, E it);
- E dequeue(Queue q);
- E frontValue(Queue q);
- int length(Queue q);

Implementations based on: arrays and **linked lists**



**Centro de
Informática**
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Queue ADT implemented as a linked list

Composite Type (Queue):

```
1  Link front; Link rear; int size ;                                // queue size
```

This implementation does consider a **header node**

Algorithm: Queue `create_queue()`

```
1  q.front  $\leftarrow$  q.rear  $\leftarrow$  create_link(NULL);                // header node
2  q.size  $\leftarrow$  0;
3  return q;
```



**Centro de
Informática**
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Queue ADT implemented as a linked list

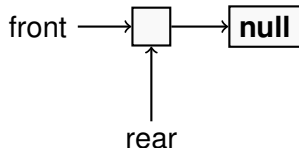
Algorithm: void enqueue(Queue q, E it)

```

1  q.rear.next  $\leftarrow$  create_link(it, NULL);
2  q.rear  $\leftarrow$  q.rear.next;
3  q.size++;

```

enqueue(*q*, 7), just after create_queue() (empty node: header)



Centro de
Informática
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Queue ADT implemented as a linked list

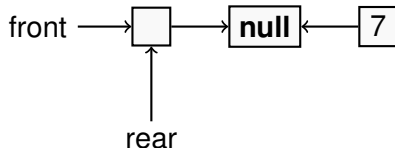
Algorithm: void enqueue(Queue q, E it)

```

1  q.rear.next ← create_link(it, NULL);
2  q.rear ← q.rear.next;
3  q.size++;

```

enqueue(*q*, 7), just after create_queue() (empty node: header)



Queue ADT implemented as a linked list

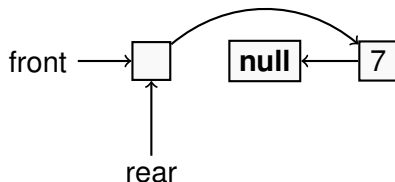
Algorithm: void enqueue(Queue q, E it)

```

1 q.rear.next  $\leftarrow$  create_link(it, NULL);
2 q.rear  $\leftarrow$  q.rear.next;
3 q.size++;

```

enqueue(*q*, 7), just after create_queue() (empty node: header)



**Centro de
Informática**
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Queue ADT implemented as a linked list

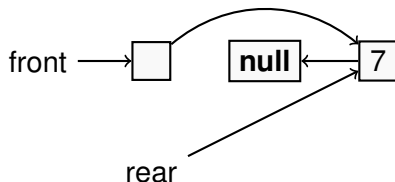
Algorithm: void enqueue(Queue q, E it)

```

1 q.rear.next ← create_link(it, NULL);
2 q.rear ← q.rear.next;
3 q.size++;

```

enqueue(*q*, 7), just after create_queue() (empty node: header)

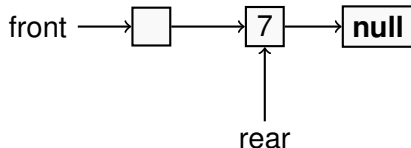


Queue ADT implemented as a linked list

Algorithm: void enqueue(Queue q, E it)

```
1 q.rear.next  $\leftarrow$  create_link(it, NULL);  
2 q.rear  $\leftarrow$  q.rear.next;  
3 q.size++;
```

Better presented as follows:



Centro de
Informática
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Queue ADT implemented as a linked list

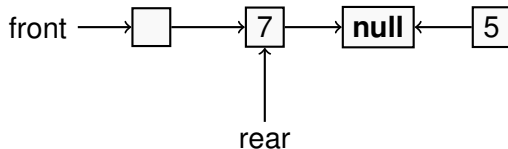
Algorithm: void enqueue(Queue q, E it)

```

1  q.rear.next  $\leftarrow$  create_link(it, NULL);
2  q.rear  $\leftarrow$  q.rear.next;
3  q.size++;

```

enqueue (q, 5)



Centro de
Informática
UFPE



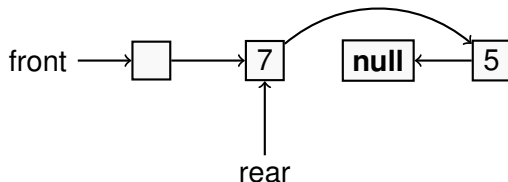
UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Queue ADT implemented as a linked list

Algorithm: void enqueue(Queue q, E it)

```
1 q.rear.next ← create_link(it, NULL);  
2 q.rear ← q.rear.next;  
3 q.size++;
```

enqueue (q, 5)

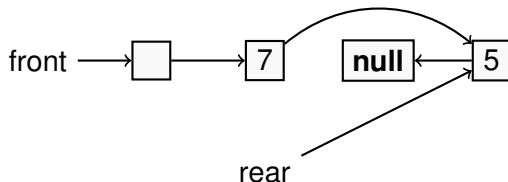


Queue ADT implemented as a linked list

Algorithm: void enqueue(Queue q, E it)

```
1 q.rear.next ← create_link(it, NULL);  
2 q.rear ← q.rear.next;  
3 q.size++;
```

enqueue(*q*, 5)



Centro de
Informática
UFPE



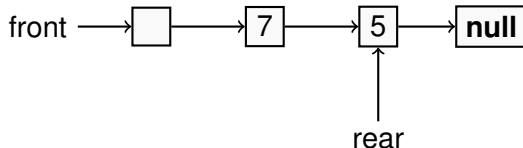
UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Queue ADT implemented as a linked list

Algorithm: void enqueue(Queue q, E it)

```
1 q.rear.next  $\leftarrow$  create_link(it, NULL);  
2 q.rear  $\leftarrow$  q.rear.next;  
3 q.size++;
```

Better presented as follows



Centro de
Informática
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Queue ADT implemented as a linked list

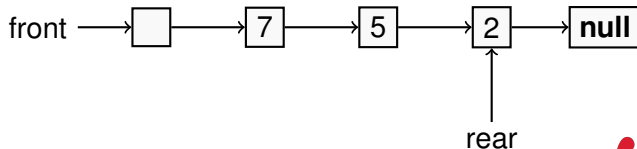
Algorithm: E dequeue(Queue q)

```

1 if q.size = 0 then error;
2 it ← q.front.next.element;
3 q.front.next ← q.front.next.next;
4 if q.front.next = NULL then q.rear ← q.front;
5 q.size--;
6 return it;

```

dequeue (q) , just after enqueue (q, 2)



Centro de
Informática
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Queue ADT implemented as a linked list

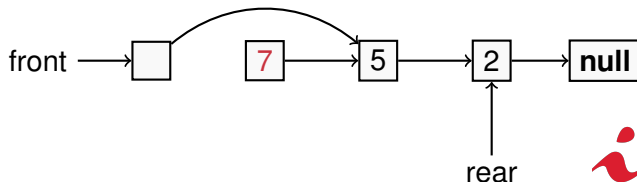
Algorithm: E dequeue(Queue q)

```

1 if  $q.size = 0$  then error;
2  $it \leftarrow q.front.next.element$ ;
3  $q.front.next \leftarrow q.front.next.next$ ;
4 if  $q.front.next = NULL$  then  $q.rear \leftarrow q.front$ ;
5  $q.size--$ ;
6 return  $it$ ;

```

dequeue (q) , just after enqueue (q, 2)



Queue ADT implemented as a linked list

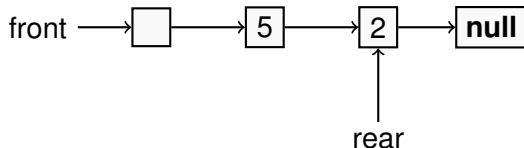
Algorithm: E dequeue(Queue q)

```

1 if  $q.size = 0$  then error;
2  $it \leftarrow q.front.next.element$ ;
3  $q.front.next \leftarrow q.front.next.next$ ;
4 if  $q.front.next = NULL$  then  $q.rear \leftarrow q.front$ ;
5  $q.size--$ ;
6 return  $it$ ;

```

After memory deallocation (by the user or by the GC)



**Centro de
Informática**
UFPE



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Agenda

1 Stacks

2 Queues

3 Bibliography



**Centro de
Informática**
UFPE

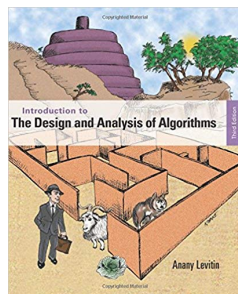


UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Bibliography



Chapter 4 (pp. 117–131)
Clifford Shaffer.
*Data Structures and
 Algorithm Analysis.*
 Dover, 2013.



Chapter 1 (pp. 25–28).
Anany Levitin.
*Introduction to the Design and
 Analysis of Algorithms.*
 3rd edition. Pearson. 2011.



**Centro de
 Informática**
 UFPE



UNIVERSIDADE
 FEDERAL
 DE PERNAMBUCO

STACKS | QUEUES

Gustavo Carvalho
(ghpc@cin.ufpe.br)

Universidade Federal de Pernambuco
Centro de Informática, 50740-560, Brazil

