Queues, Deques and Dictionaries

Lecture 9

1107186 – Estruturas de Dados

Prof. Christian Azambuja Pagot
CI / UFPB

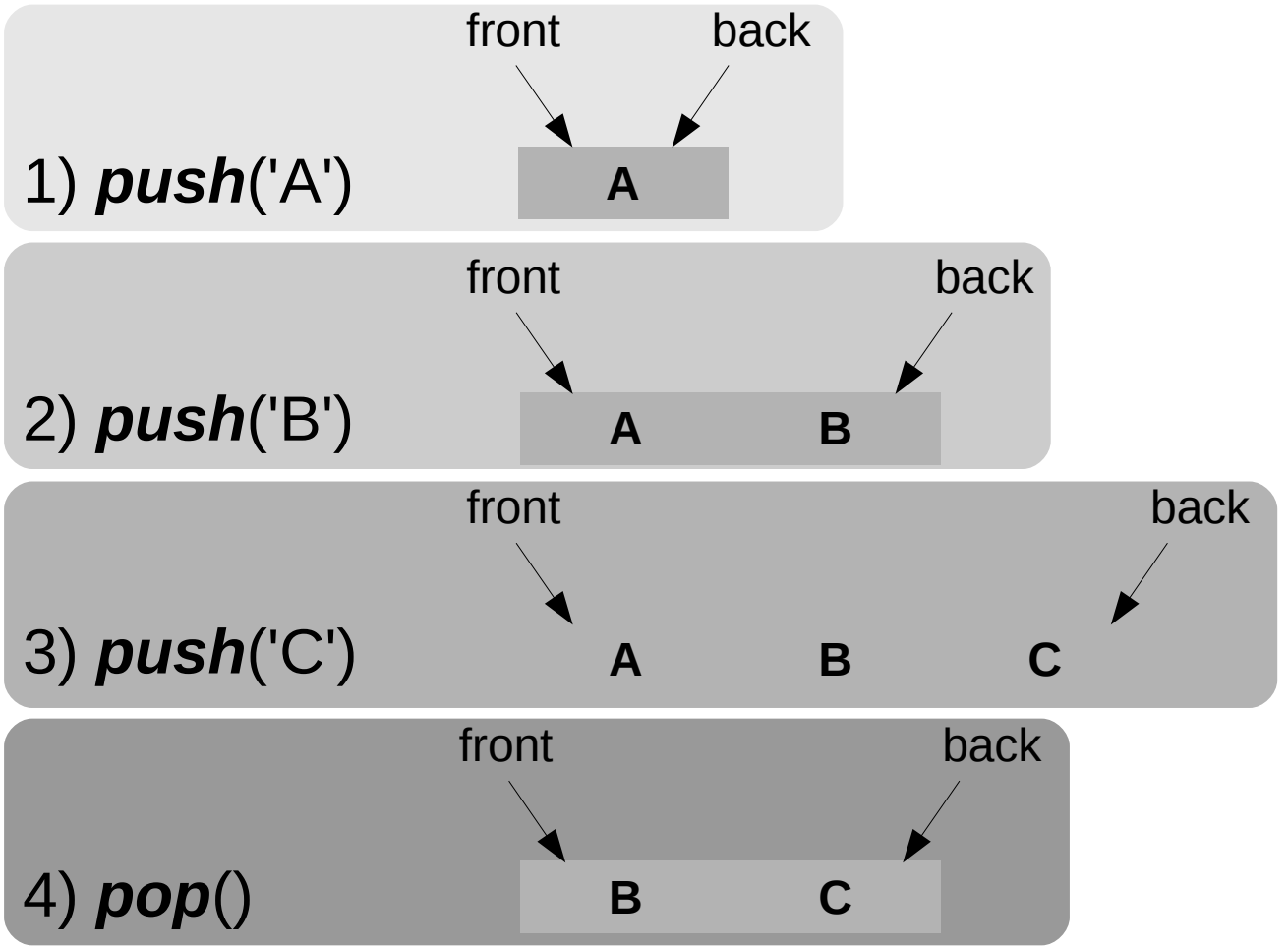
Universidade Federal da Paraíba
Centro de Informática

What is a Queue?

- A **queue** is an **abstract data type** where elements can be **inserted** (*pushed*) and **removed** (*popped*) according to the following policies:
  - **Push** inserts a new item in the **back** of the queue.
  - **Pop** removes the item at the **front** of the queue.
  - The above insert/remove policy is also called **FIFO** (**First In-First Out**).

What is a Queue?

- **Example**:

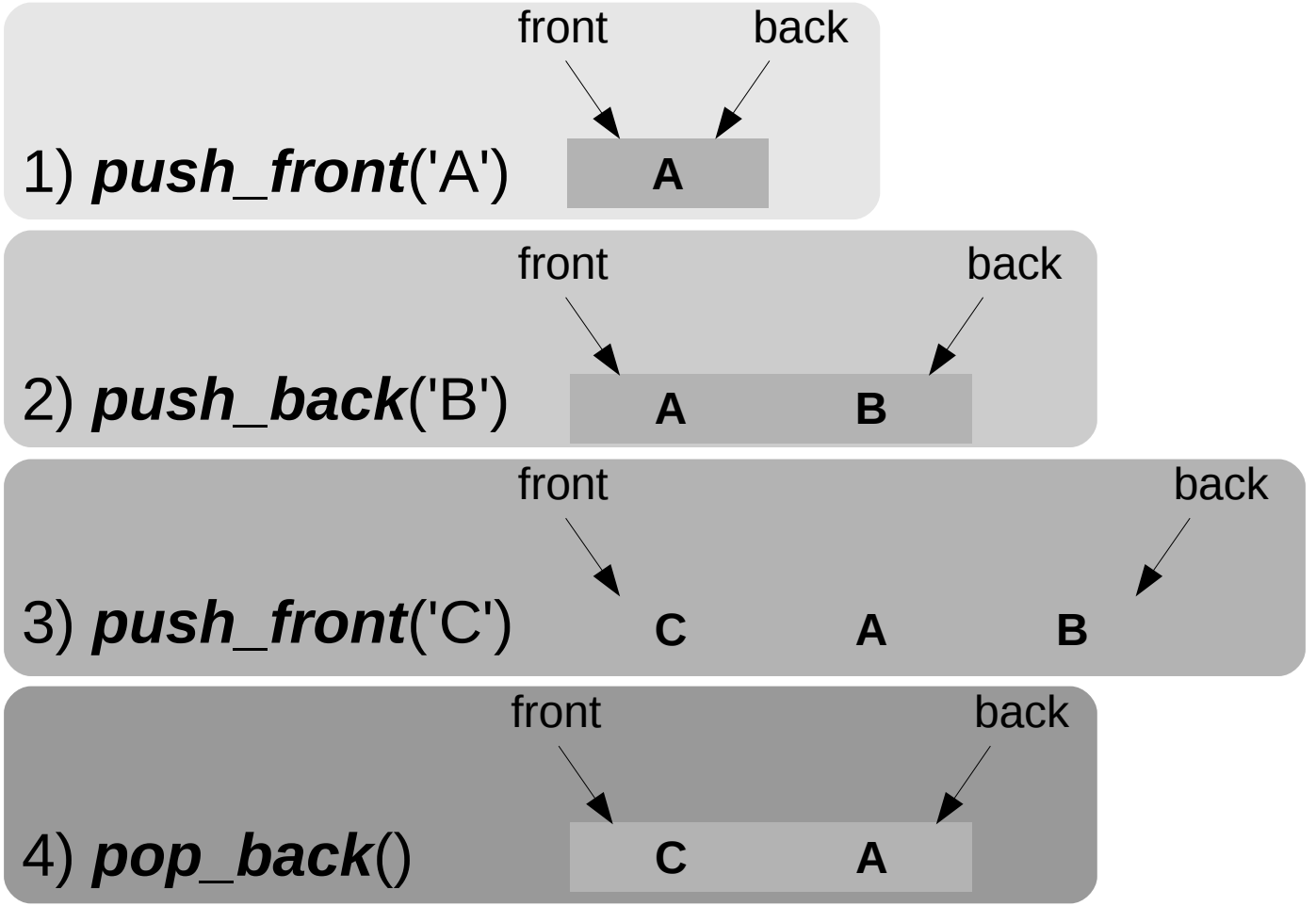


What is a Deque?

- Deque: **D**ouble **E**nded **Que**ue.
- A **deque** is an **abstract data type** where elements can be **inserted** (*pushed*) and **removed** (*popped*) according to the following policies:
  - **Push** inserts a new item in the **back** or in the **front** of the deque.
  - **Pop** removes the item at the **front** or at the **back** of the deque.

What is a Deque?

- **Example**:



What is a Dictionary?

- **Associative array**, **map**, **symbol table** or **dictionary**.
- A **dictionary** is an **abstract data type** composed by a collection of (key, value) pairs, where key is unique.

What is a Dictionary?

- **Example**:
  - Clients of a bank:

	Key (CPF)	
Data item	177829829-73	Christian
	888226782-75	Paulo
	891034820-28	Christian

What is a Dictionary?

- **Common operations**:
  - *Search*(D, k).
  - *Insert*(D, x).
  - *Delete*(D, x).
  - *Max*(D), *Min*(D).
  - *Predecessor*(D, k), *Successor*(D, k).

Searches for key *k* in D, and returns a pointer to the corresponding x item.

Add the *x* (data and key) item to D.

Given a pointer to the *x* item in D, removes it from D.

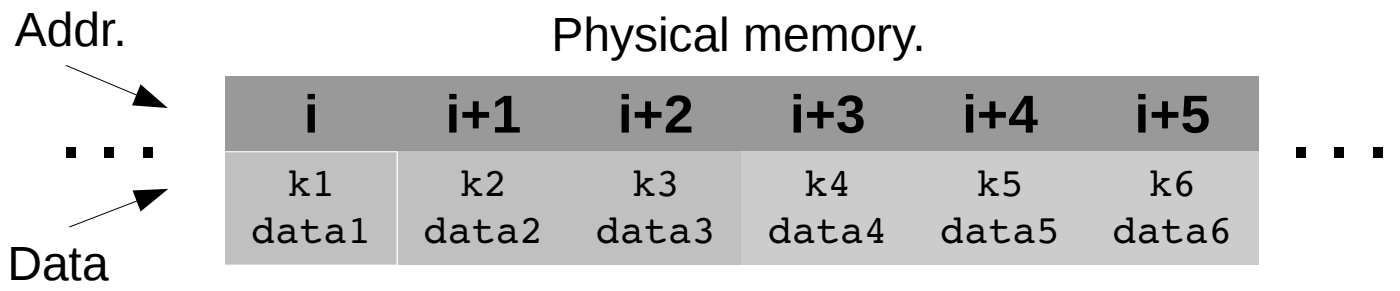
Returns a pointer to the data item with largest (or smallest) key.

Returns a pointer to the data item whose key is immediately before (after) *k* in sorted order.

D: dictionary.
k: key.
x: data item.

Analysis of Some Possible Dictionary Implementations (by Steven S. Skiena)

- **Array**:



Search item with key k.

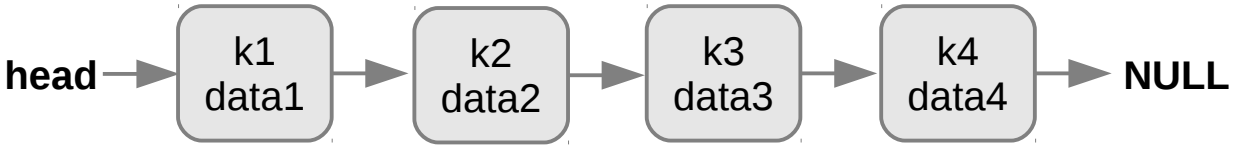
Insert x (data + key)

Receives the index to x

Operation	Unsorted Array	Sorted Array
Search(D, k)	$O(n)$	$O(\log n)$
Insert(D, x)	$O(1)$	$O(n)$
Delete(D, x)	$O(1)^*$	$O(n)$
Successor(D, k)	$O(n)$	$O(1)$
Predecessor(D, k)	$O(n)$	$O(1)$
Minimum(D)	$O(n)$	$O(1)$
Maximum(D)	$O(n)$	$O(1)$

Analysis of Some Possible Dictionary Implementations (by Steven S. Skiena)

- **Single Linked List**:



Search item with key k.

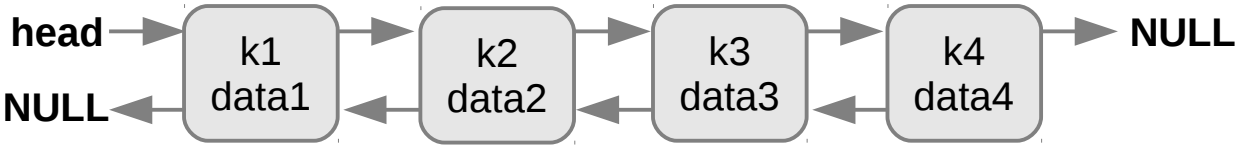
Insert x (data + key)

Receives a pointer to x

Operation	Unsorted List	Sorted List
Search(D, k)	$O(n)$	$O(n)$
Insert(D, x)	$O(1)$	$O(n)$
Delete(D, x)	$O(n)$	$O(n)$
Successor(D, k)	$O(n)$	$O(1)$
Predecessor(D, k)	$O(n)$	$O(n)$
Minimum(D)	$O(n)$	$O(1)$
Maximum(D)	$O(n)$	$O(1)$

Analysis of Some Possible Dictionary Implementations (by Steven S. Skiena)

- **Doubly Linked List**:



Search item with key k.

Insert x (data + key)

Receives a pointer to x

Operation	Unsorted List	Sorted List
Search(D, k)	$O(n)$	$O(n)$
Insert(D, x)	$O(1)$	$O(n)$
Delete(D, x)	$O(1)$	$O(1)$
Successor(D, k)	$O(n)$	$O(1)$
Predecessor(D, k)	$O(n)$	$O(1)$
Minimum(D)	$O(n)$	$O(1)$
Maximum(D)	$O(n)$	$O(1)$

Analysis of Some Possible Dictionary Implementations (by Steven S. Skiena)

- **Comparing all possibilities:**

Operation	UA	SA	USLL	SSLL	UDLL	SDLL
Search(D, k)	$O(n)$	$O(\log n)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Insert(D, x)	$O(1)$	$O(n)$	$O(1)$	$O(n)$	$O(1)$	$O(n)$
Delete(D, x)	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$
Successor(D, k)	$O(n)$	$O(1)$	$O(n)$	$O(1)$	$O(n)$	$O(1)$
Predecessor(D, k)	$O(n)$	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$
Minimum(D)	$O(n)$	$O(1)$	$O(n)$	$O(1)$	$O(n)$	$O(1)$
Maximum(D)	$O(n)$	$O(1)$	$O(n)$	$O(1)$	$O(n)$	$O(1)$

UA: Unsorted array **SA**: sorted array.
USLL: Unsorted Single Linked List **SSLL**: Sorted Single Linked List
UDLL: Unsorted Doubly Linked List **SDLL**: Sorted Doubly Linked List