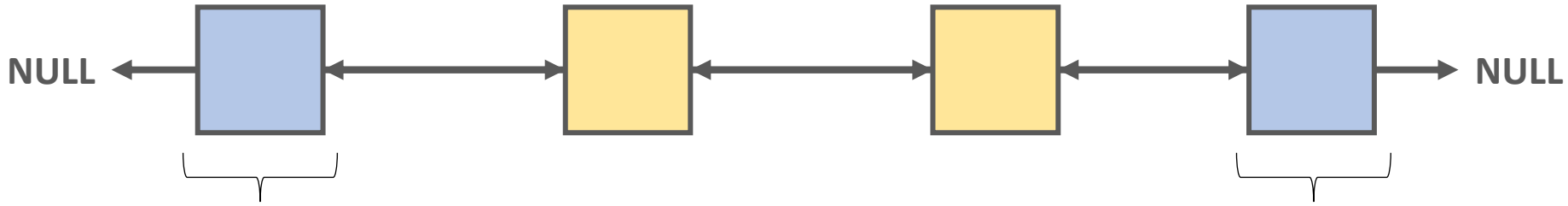
Doubly Linked Lists
(Algorithms and Data Structures)

Doubly Linked Lists

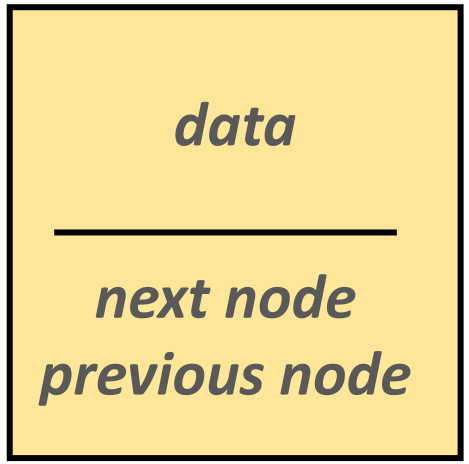
- it is another **data structure** – so the aim is to be able to store items efficiently (*insertion* and *removal* operations)
- arrays have a huge disadvantage: there may be „holes” in the data structure and we have to shift a lot of items
- this problem can be eliminated by **doubly linked lists**



*we have access to the first node of the linked list (**head node**)*

*we have access to the last node of the linked list (**tail node**)*

Linked Lists



- every node stores the **data** itself and **references** to the next node and to the previous node in the linked list
- this is why **doubly linked lists** need more memory than linked lists
- it has an advantage – there can not be „holes” in the data structure so **there is no need for shifting items**

Doubly Linked Lists

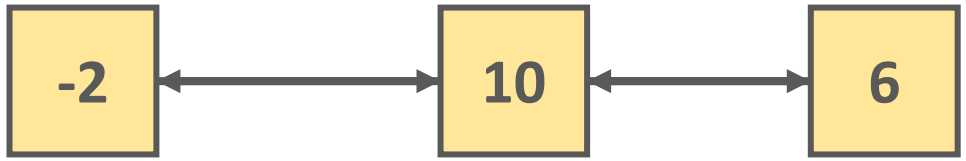
Doubly Linked Lists

Doubly Linked Lists

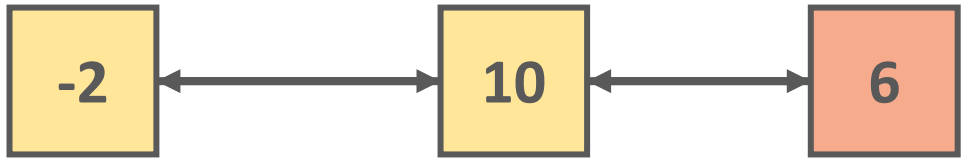
Doubly Linked Lists

Doubly Linked Lists

Doubly Linked Lists



Doubly Linked Lists



Doubly Linked Lists

Doubly Linked Lists Advantages

- we store references to the **head node** and the **tail node** as well so these nodes can be accessed in **O(1)** running time
- it can be traversed in both directions (huge advantage)
- removing a given node is easier because there is a pointer to the **previous node** as well

Doubly Linked Lists Disadvantages

- need **more memory** because of the references (**2** pointers)
- a bit more **complicated to implement** because we have to handle both of the pointers
- still have not solved the main issue – how to search for arbitrary items faster than **O(N)** linear running time?