Arrays vs. Linked Lists

(Algorithms and Data Structures)

Arrays and Linked Lists

1.) DYNAMIC AND STATIC DATA STRUCTURES

- **arrays are static** data structures – we have to know the size of the data structures in advance (or we have to resize it)
- linked lists are **dynamic data structures** – they can grow organically based on the references (no resize operation needed)

Arrays and Linked Lists

2.) RANDOM ACCESS (RANDOM INDEXING)

- items in an array are located right next to each other in the main memory (RAM) this is why we can use **indexes**
- there is no random access in a **linked list** data structure

Arrays and Linked Lists

3.) MANIPULATING THE FIRST ITEMS

- we have to shift several items (all the items in worst-case) when manipulating the first items in **arrays**
- **linked lists** are dynamic data structures – we just have to update the references aound the *head node*

Arrays and Linked Lists

4.) MANIPULATING THE LAST ITEMS

- there can not be holes in the data structure when manipulating the last items in **arrays**
- **linked lists** have access to the first node (*head node*) exclusively so in this case we have to traverse the whole list in **O(N)** running time

Arrays and Linked Lists

5.) MEMORY MANAGEMENT

- **arrays** do not need any extra memory
- **linked lists** on the other hand do need extra memory because of the references (pointers)

Arrays and Linked Lists

Searching for an arbitrary item (or removing an arbitrary item) takes O(N) linear running time for both data structures

Arrays and Linked Lists

	Linked Lists	Arrays
search	O(N)	O(1)
insert at the start	O(1)	O(N)
insert at the end	O(N)	O(1)
waste space	O(N)	0