**Linked List**

Now we look into Linked Lists that are sorted.

**?** When do we use Linked Lists, and when do we use arrays?

* Linked Lists are easier to insert/modify and more flexible because of pointers. If we want to insert something in the middle of the array, we have to move all members by 1.
* Also, if we don't know how many members needed, use Linked Lists.
* Arrays are easier to index specific values. If we are not moving the data, the pointers are taking additional spaces, and have more operations (dereference).

Text, letter

Description automatically generated

Text, letter

Description automatically generated

1. **Searching the list**

+) Linked list not sorted

Text

Description automatically generated with medium confidence

+) Linked list is sorted 🡪 Other conditions for stopping

Text

Description automatically generated with medium confidence

1. **Insert into List**

Special cases to consider:

* Inserting at head
* Inserting at end
* Inserting at middle

\*\*We need to keep track of also **the previous node** (unless we do node->getNext()->getData())

Text

Description automatically generated

1. **Delete a Node**

Also needs 2 pointers (to have access to the previous node)

Text

Description automatically generated

1. **Destructor**

Text

Description automatically generatedSince now that every node is dynamically allocated, we **need destructor** to avoid memory leak.

// Here we are changing head from beginning to end, and p is the previous head (that gets deleted)

1. **Copy Constructor**

Text

Description automatically generatedIt's obvious that we need to make a **deep copy** for the linked list, not shalow copy.

**NOTE that "head" in line 4 is the "head" of the new LinkedList (that we call copy on)**

**🡪 DO NOT create a new head !!!**

1. **Assignment Operator**

lhs = rhs

This is similar to copy constructor, but we must deal with **edge cases**

* lhs is not empty 🡪 Avoid memory leak
* rhs is the same as lhs 🡪 Don't need to copy at all.

// Note that we are not comparing "same contents", we are comparing "same objects"

Graphical user interface, text, application

Description automatically generated