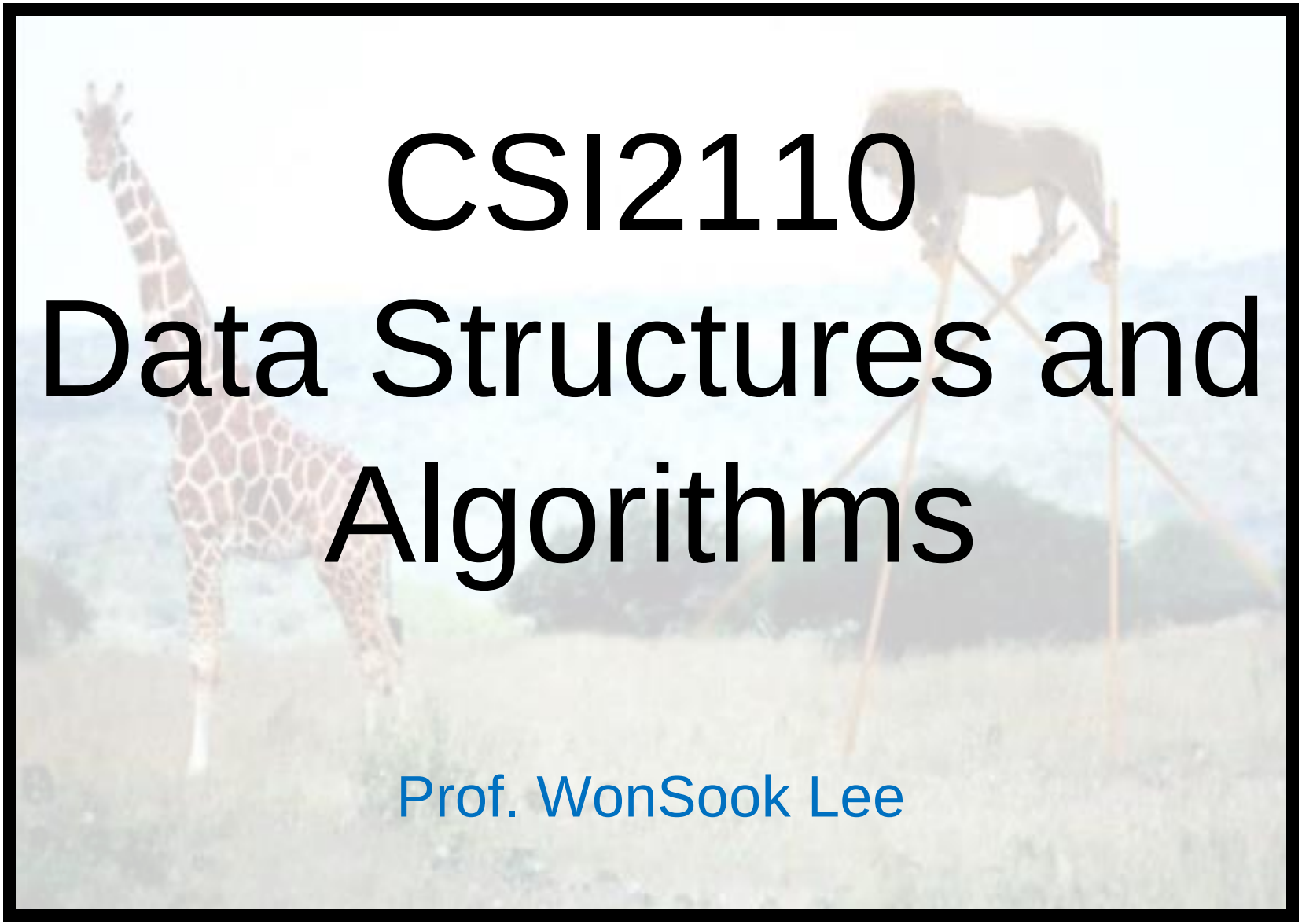
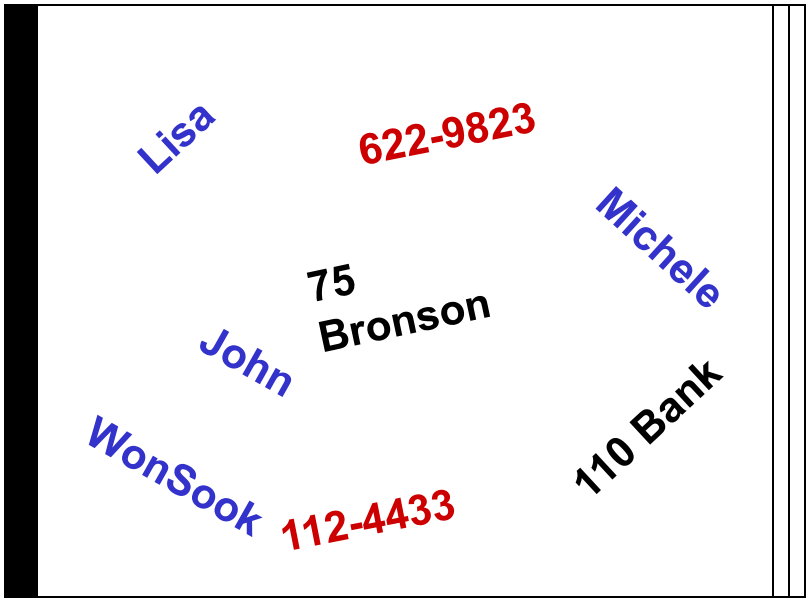


CSI2110
Data Structures and Algorithms

Prof. WonSook Lee

Data Structures ?

Example:
Electronic Phone Book



Contains different **DATA**:

- names
- phone number
- addresses

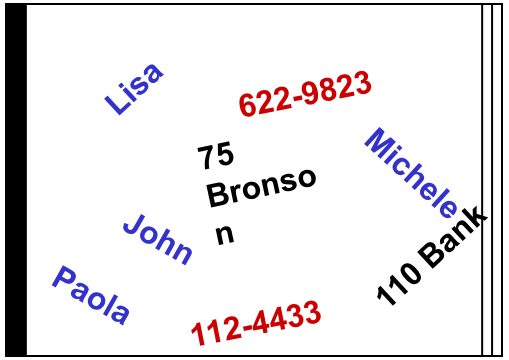
Need to perform certain **OPERATIONS**:

- add
- delete
- look for a phone number
- look for an address

How to organize the data so to optimize the efficiency of the operations

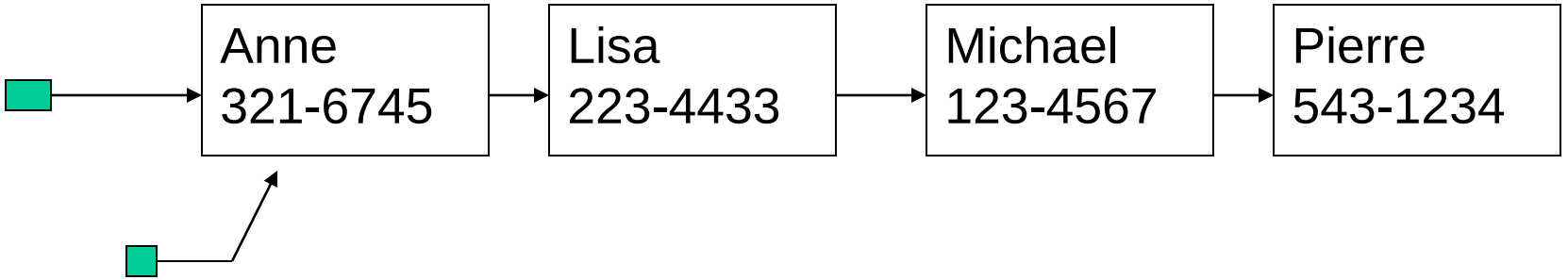
Data Structures ?

Example:



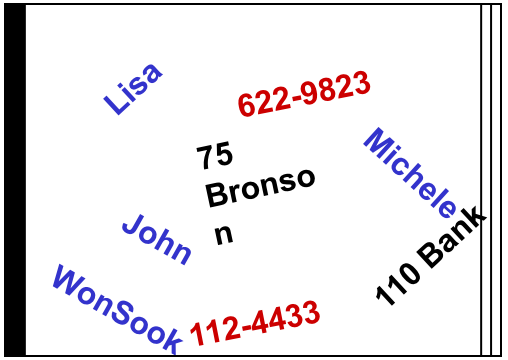
Lisa 223-4433	Pierre 543-1234	Michael 123-4567	Anne 321-6745

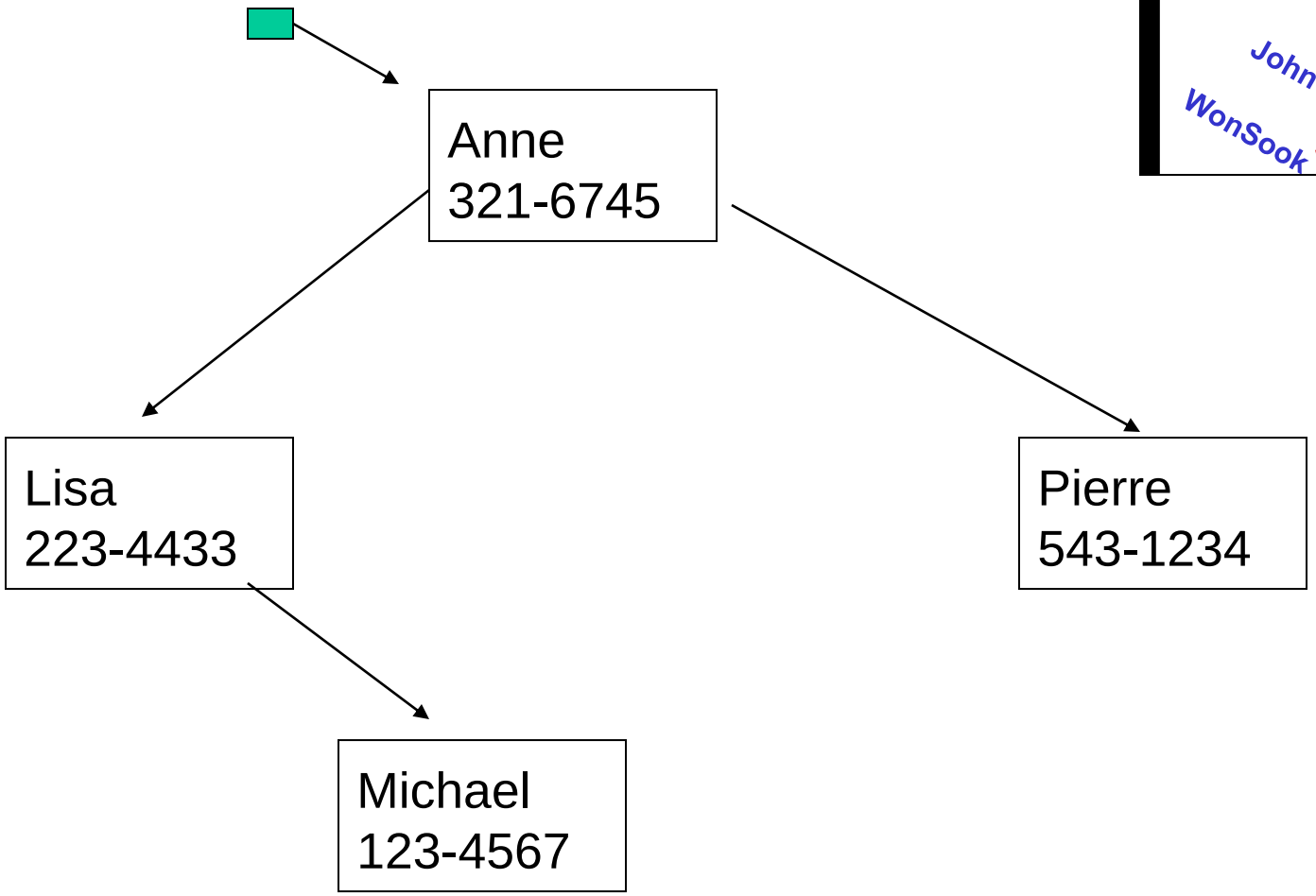
Anne 321-6745	Lisa 223-4433	Michael 123-4567	Pierre 543-1234



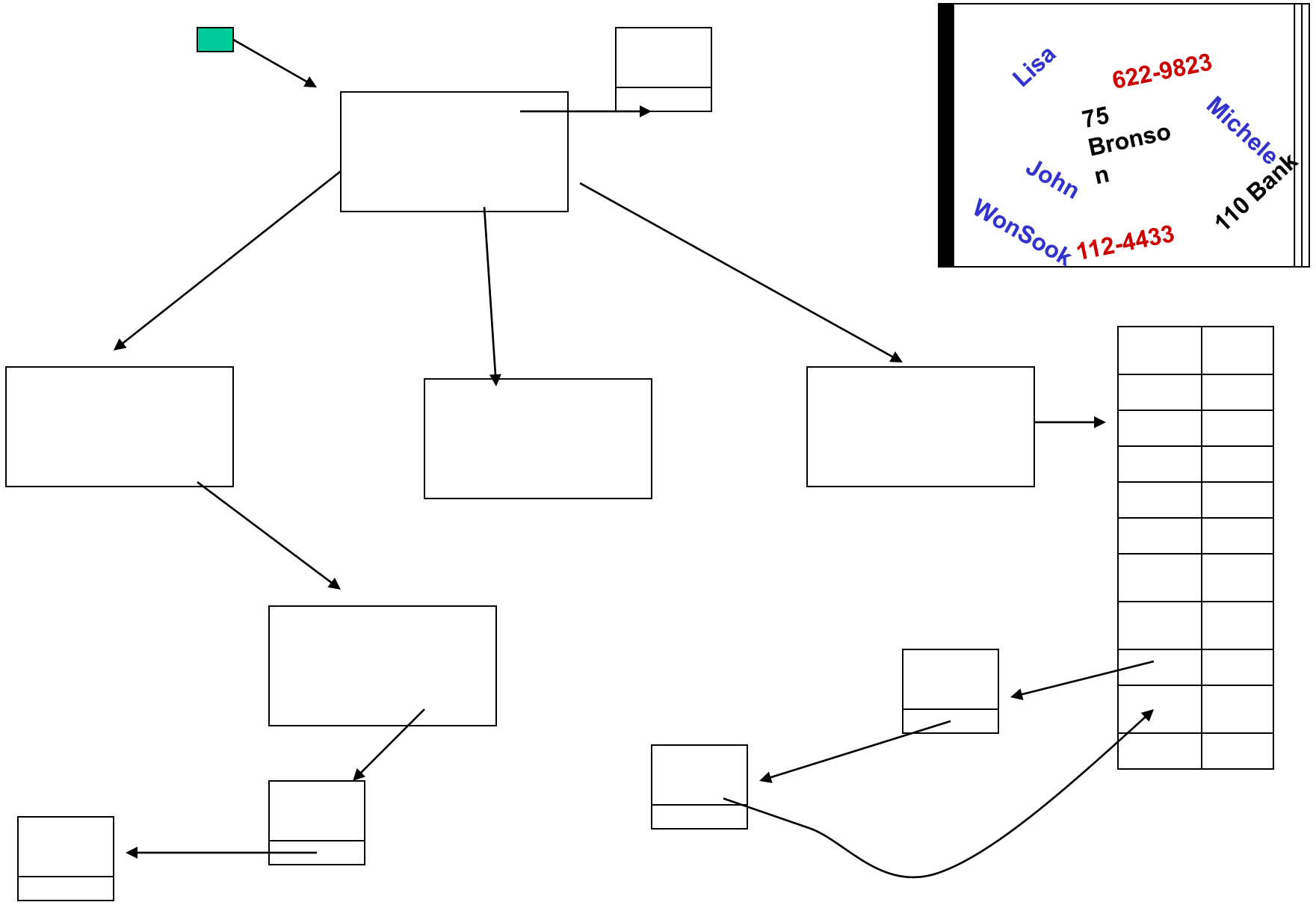
Data Structures ?

Example:



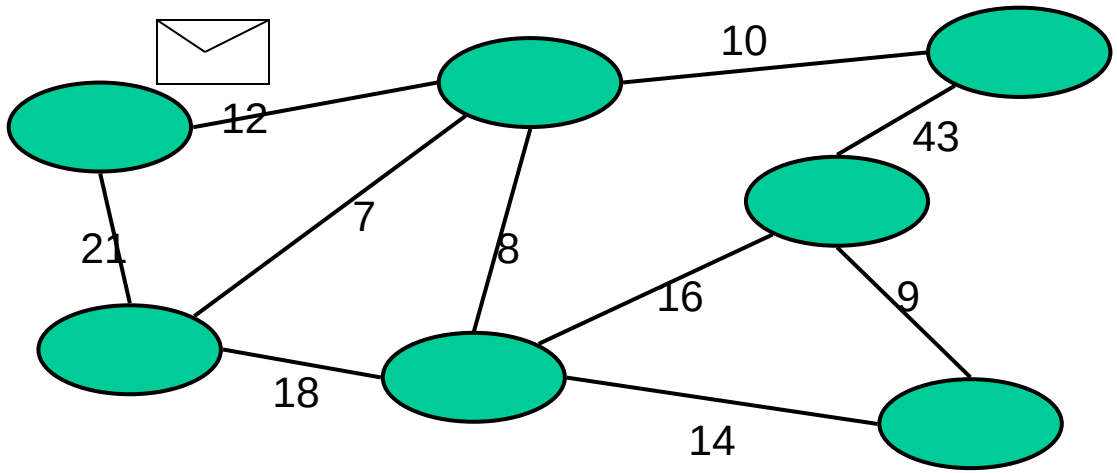


Data Structures ?



Data Structures ?

Example:
Finding the best route for an email message in a network



Contains **DATA**:

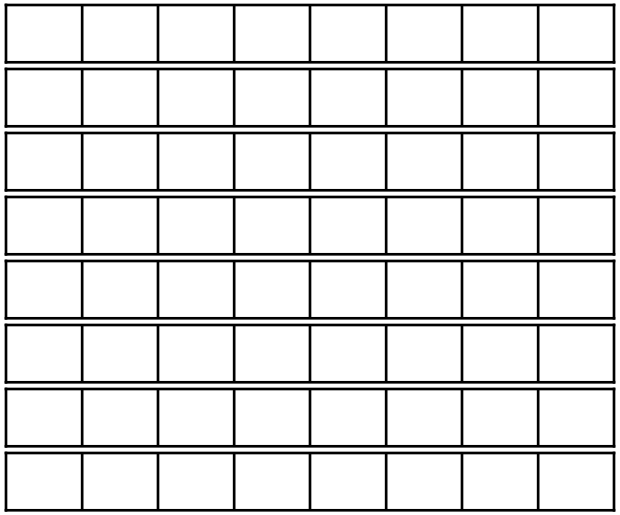
- network + traffic

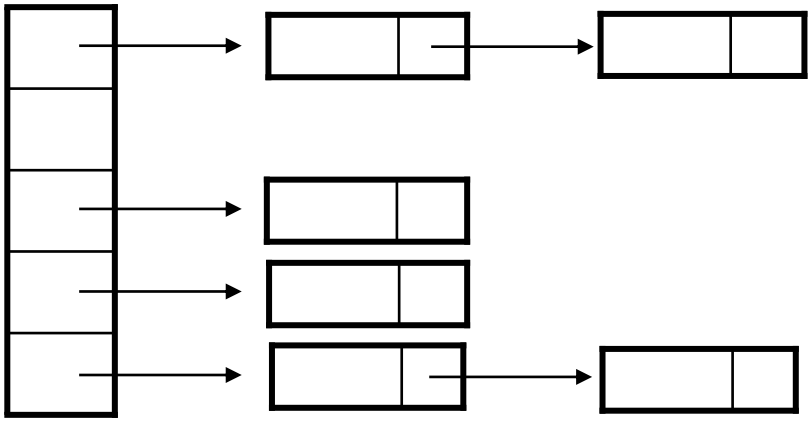
Need to perform certain **OPERATIONS**:

- Find best route

Data Structures ?

How to represent the data





so to perform the operations efficiently

Data Structures ?

Keep in mind the operations you need to perform

Choose the **best** structure for your data

Study different data structures

How to understand if a data structure is good

Objectives of the course

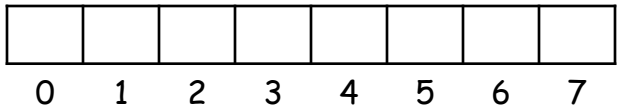
Present in a systematic fashion the most commonly used data structures, emphasizing their *abstract properties*.

Discuss typical algorithms that operate on each kind of data structure, and analyze their performance.

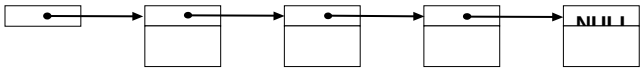
Compare different Data Structures for solving the same problem, and choose the best.

Review

- Arrays

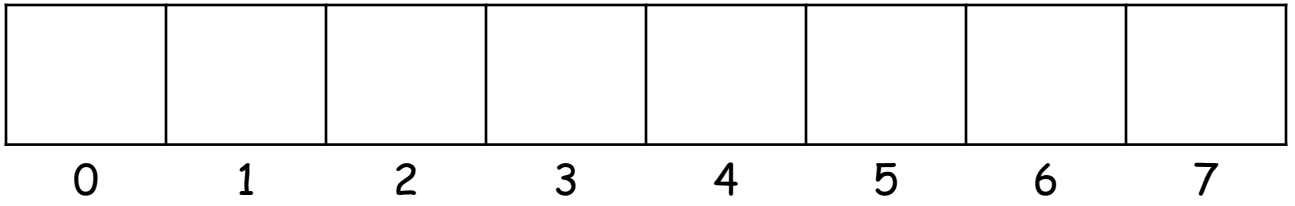


- Linked Structures



Array

A:

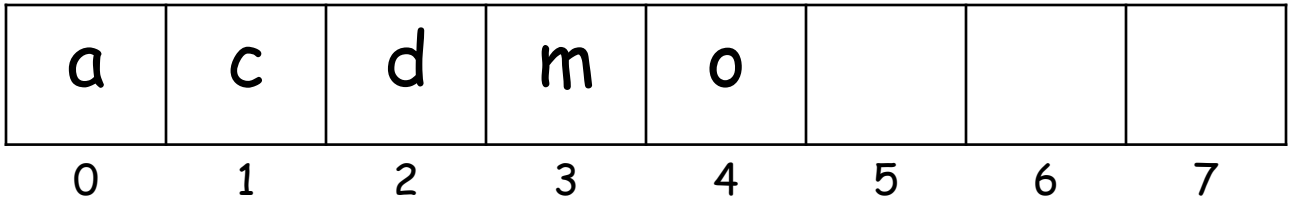


Numbered collection of variables of the same type. Fixed length.

- **Static** structure
- Direct access

Insertion ?
Deletion ?

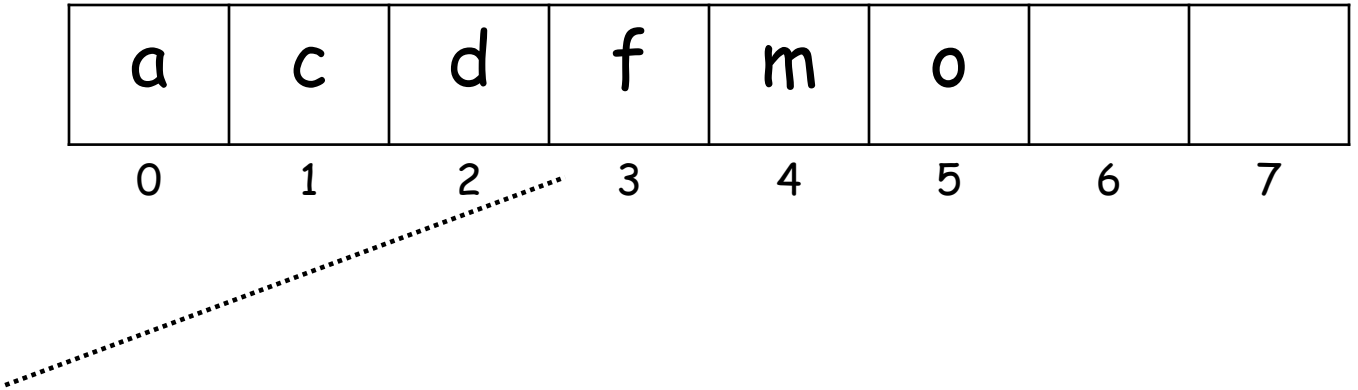
Array



f

example of insertion in a sorted array

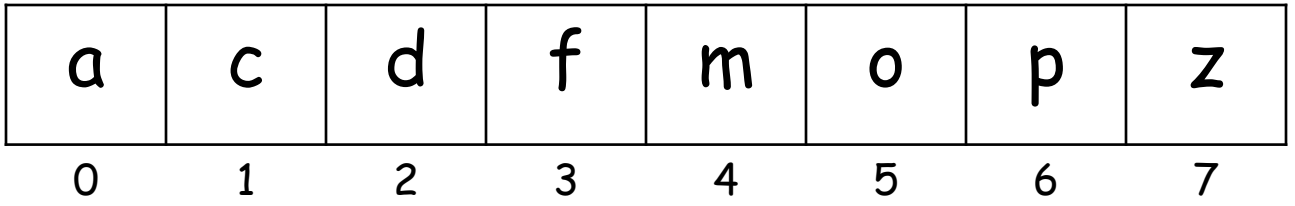
Array



move “m” and “o”
to make room for “f”

example of insertion in a sorted array

Array



1) For insertions and deletions
elements MUST BE MOVED

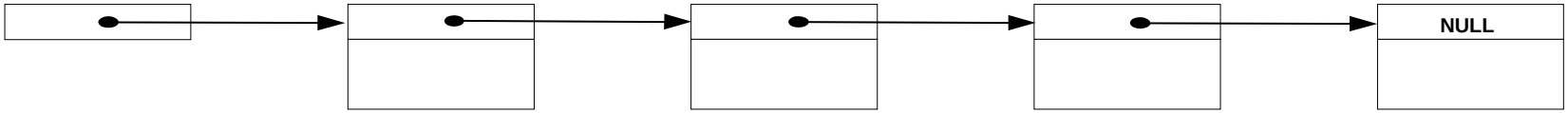
2) What happens when the array is FULL ?

Operations on Arrays

```
void        addElement(int index, Element e)   // insert
Element     setElement(int index, Element e)
Element     getElement(int index )
Element     remove(int index )
int         size()
```

Supported set of operations on a data structure define the interface of a data structure

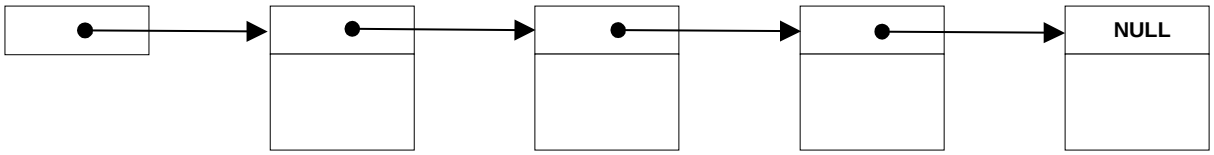
Linked Structures



- **Dynamic** structure Size kavramı yoktur. Ne kadar eklersen o kadar alır
- Sequential access Erişme işlemleri için daima 1. elemandan başlamak gerekir
- Insertion and deletion occur without moving elements

Single Linked Lists

Tek yönlü bağlantı olduğu için

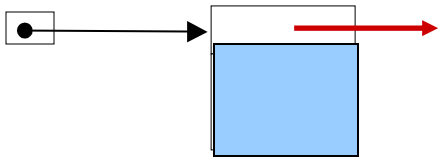


Node v

Object element

Node next

v.next



v.element

Java Implementation – Singly Linked List

you will review it in the Lab

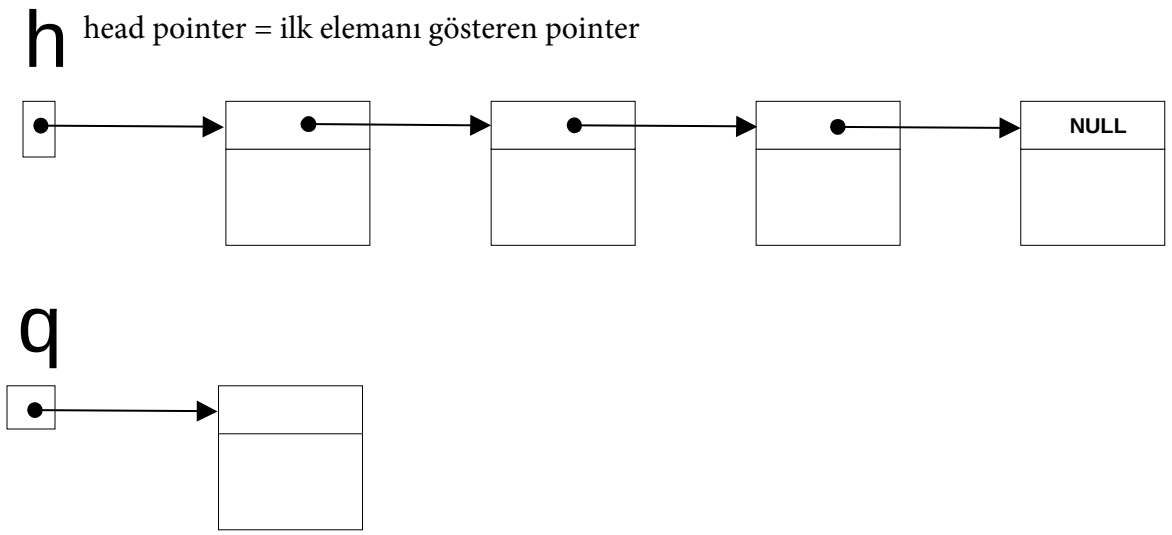
Usual Methods (see textbook):

- void setElement(Object e) Herhangi bir node'nin element'ini setler. v.setElement(4)
- void setNext(Node newNext) Herhangi bir node'nin bir sonraki tanıdığı nodeyi setler. v1.setNext(v2)
- Object getElement() Herhangi bir node'nin elemanını verir. v nodesinin elementi 4 ise v.getElement() 4 verir
- Node getNext() Herhangi bir node'nin tanıdığı sonraki node'yi verir. v1 -> v2 şeklinde olsun, v1.getNext() v2'yi verir

Örnek
v1->v2->v3->null
Yukarıdaki yapıya v4 nodesini ekle. v4 nodesi oluşturulmuş olduğunu kabul et
v4'ün element'i X, bir sonraki elemanı ise null olsun.
v4.setElement(X)
v4.setNext(null)
v3.setNext(v4) = v1->v2->v3->v4->null

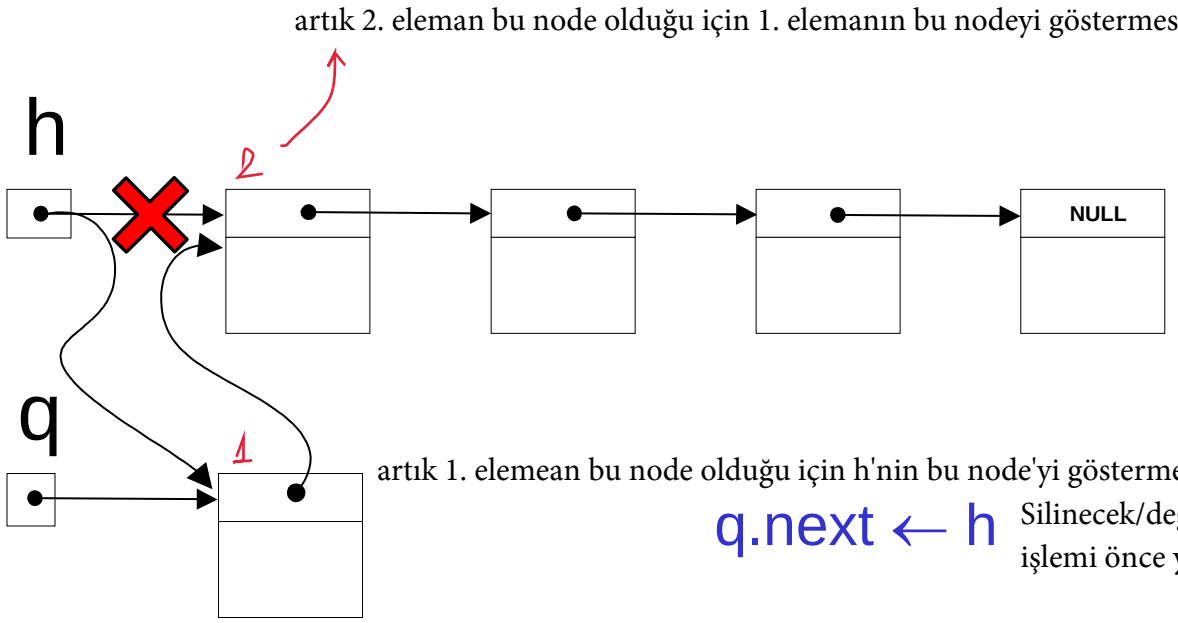
Insertion

Original configuration:



Goal: to insert the element q into list h.

Insertion at the beginning



$q.next \leftarrow h$ Silinecek/değiştirilecek bağlantının işlemi önce yapılır

$h \leftarrow q$

(easy)

… we are using pseudocode …

pseudocode

q.next ← h

variable **q.next** gets the value of variable **h**

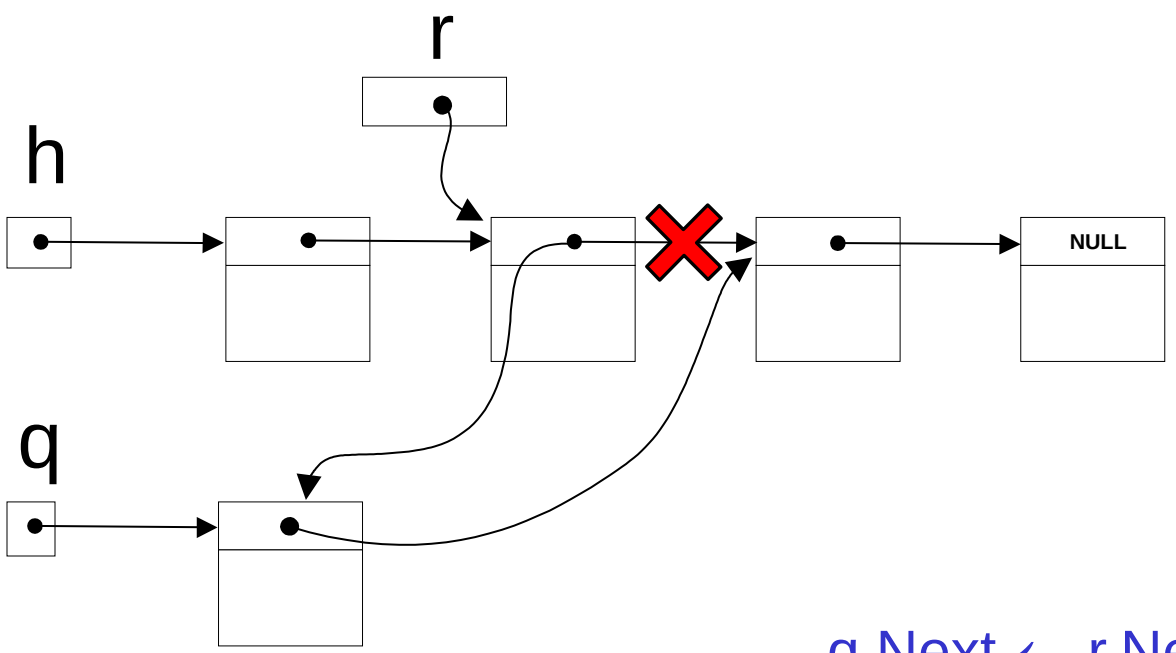
(q.next:= h)

pseudocode

$$h \leftarrow q$$

variable **head** gets the value of variable **q**

Insertion after r

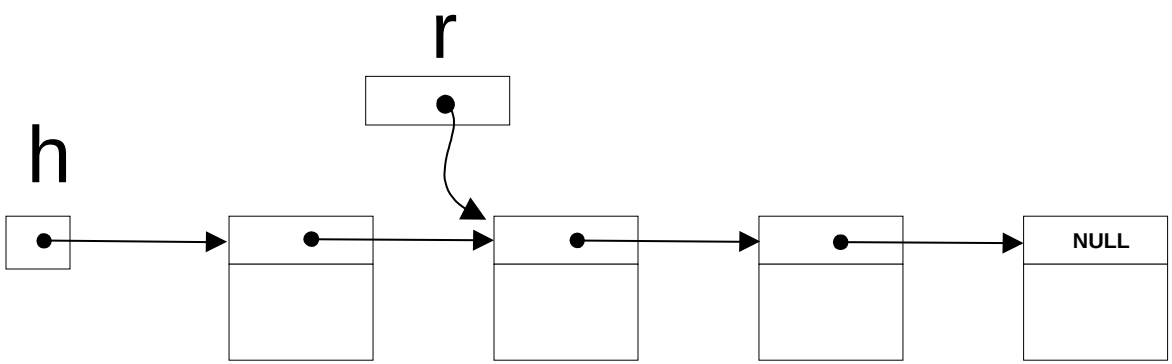


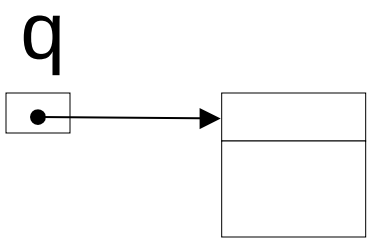
q.Next ← r.Next

r.Next ← q

(easy)

Insertion before r





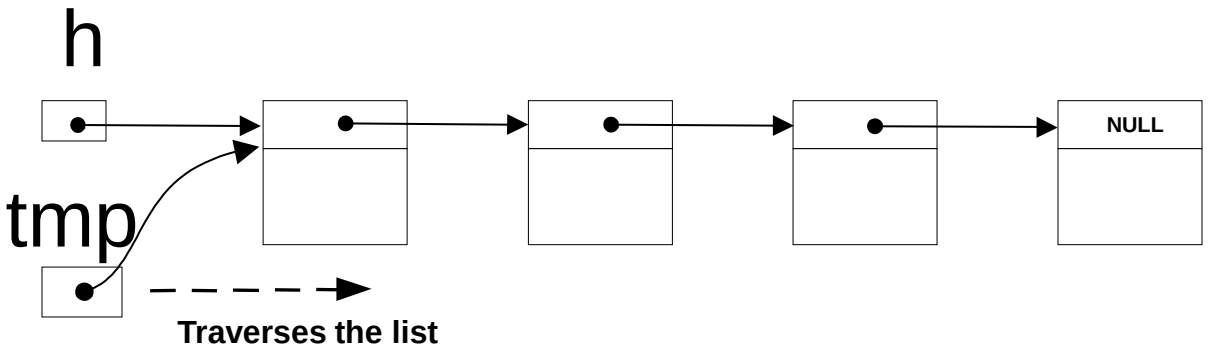
(more difficult)

- Must maintain a pointer to the preceding element

or

- Exchange the contents pointed to by r and q, and insert q after r.

Search

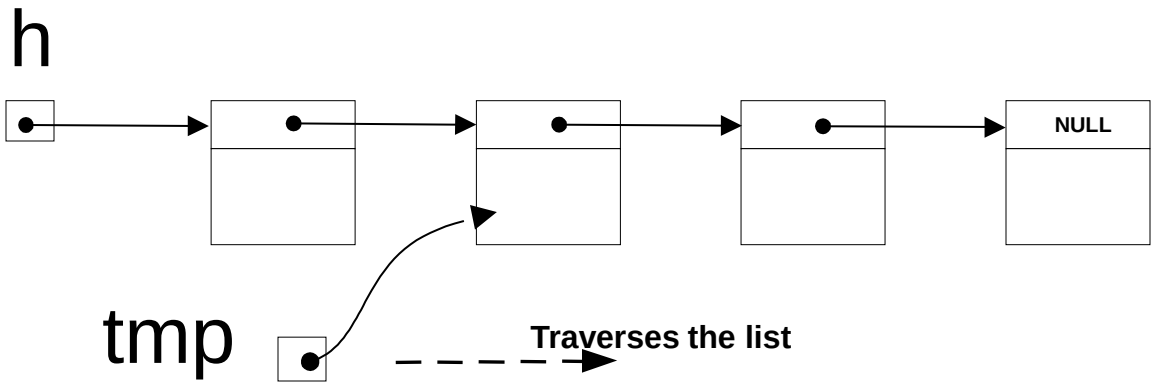



```
Node tmp;
tmp ← p;
while (tmp != null ) {
    if tmp .element  is  ce-que-je-recherche {
        return tmp ; }
    else
```

Listenin sonuna gelene kadar

aradığımız elemanı bulduysak elemanı tutan node'yi return et

Search

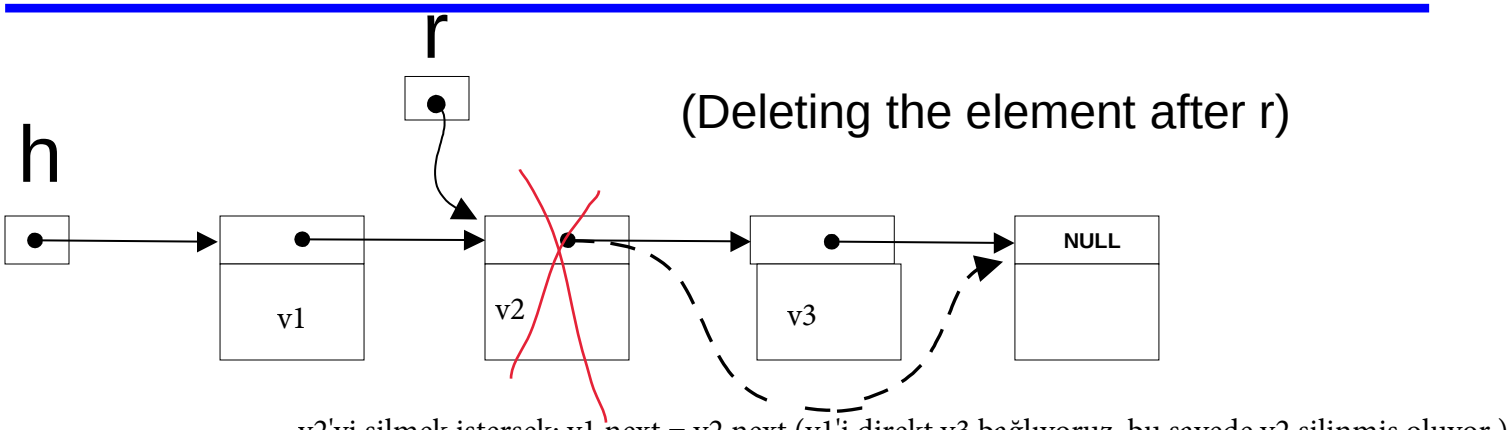



```
Node tmp;
tmp ← firstnode;
while (tmp != null ) {
     if tmp .element  is  ce-que-je-recherche {
        return tmp n; }
    else {tmp ← tmp .next; }
  }
  return tmp;
```

tmp.next bir sonraki node'yi verir.

buradaki return tmp çalışırsa listede eleman bulunamamıştır, tmp null olur bu sebeple null döner

Deletion

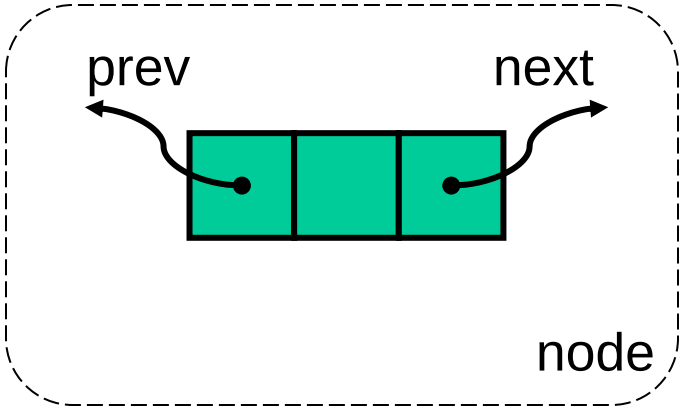


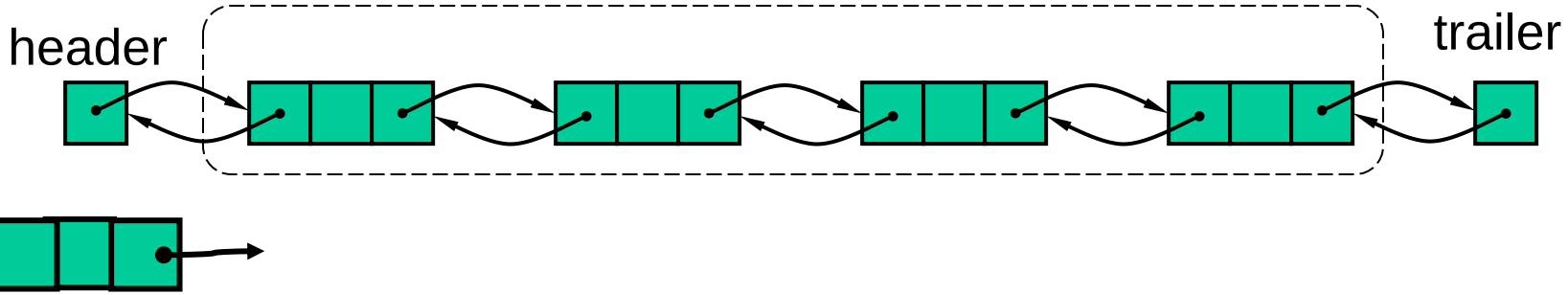
First element (easy)	h ← h.Next
Element after r (easy)	r.Next ← r.Next.Next
Element at r (difficult)	• Use a pointer to the preceding element, or • Exchange the contents of the element at r with the contents of the element following r, and delete The element after r. **Very difficult if r points to the last element!

çift yönlü bağlı: hem ileri hem geri

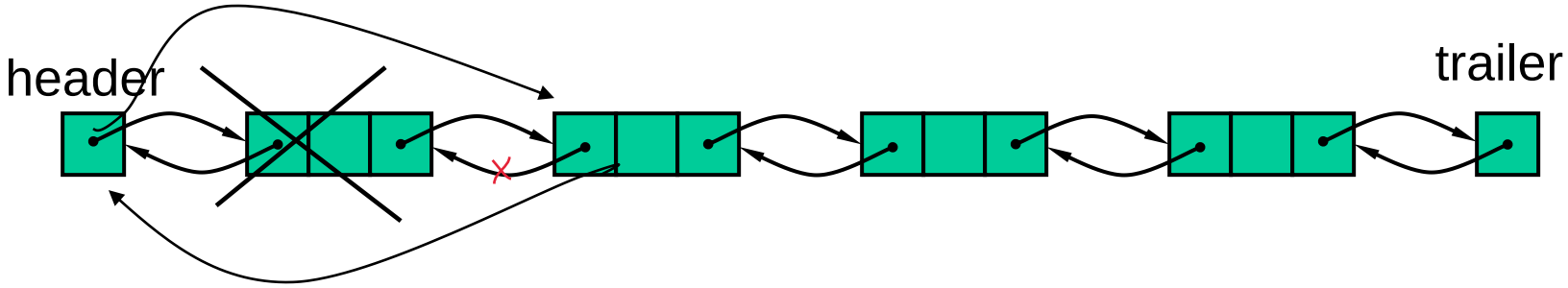
Doubly Linked List

- Nodes store:
  - element
  - link to the previous node
  - link to the next node
- Special trailer and header nodes





Deletion (first element)



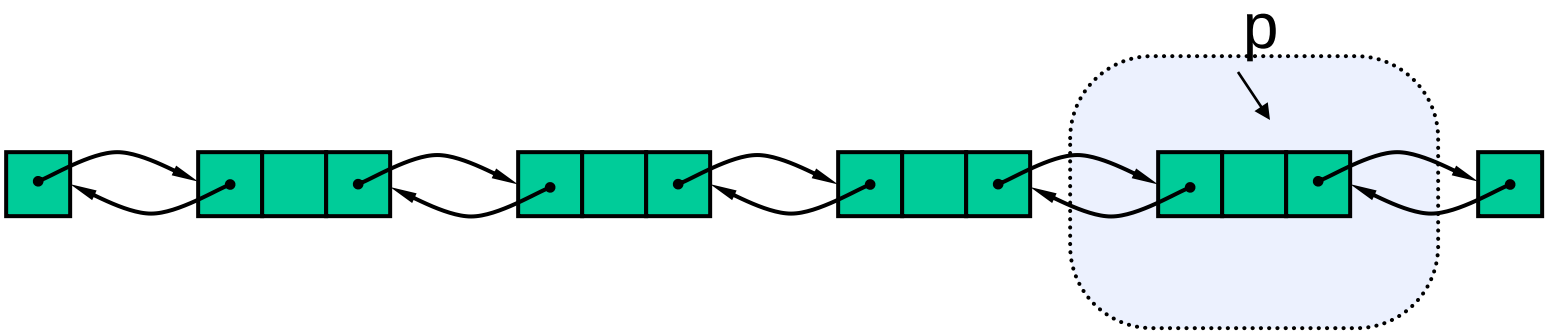
Singly Linked List'ten farklı olarak header kutunun kendisini temsil eder

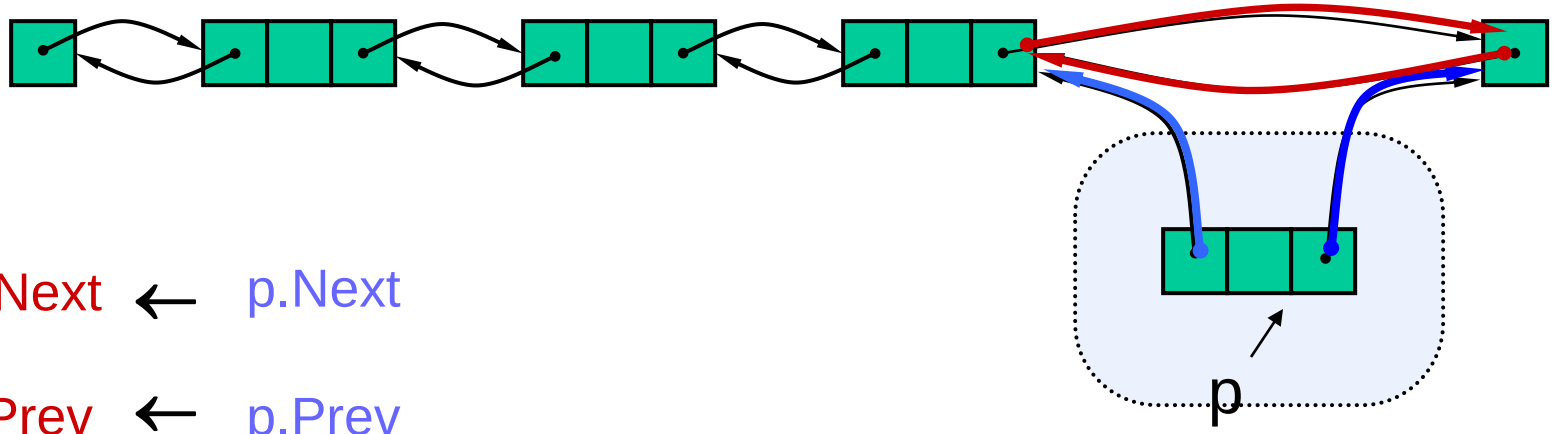
header.next.next.prev ← header

header.next ile bir sonraki kutuya gidilir

header.next ← header.next.next

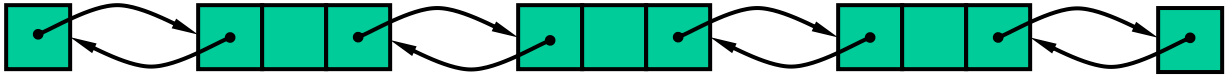
Deletion (element p)



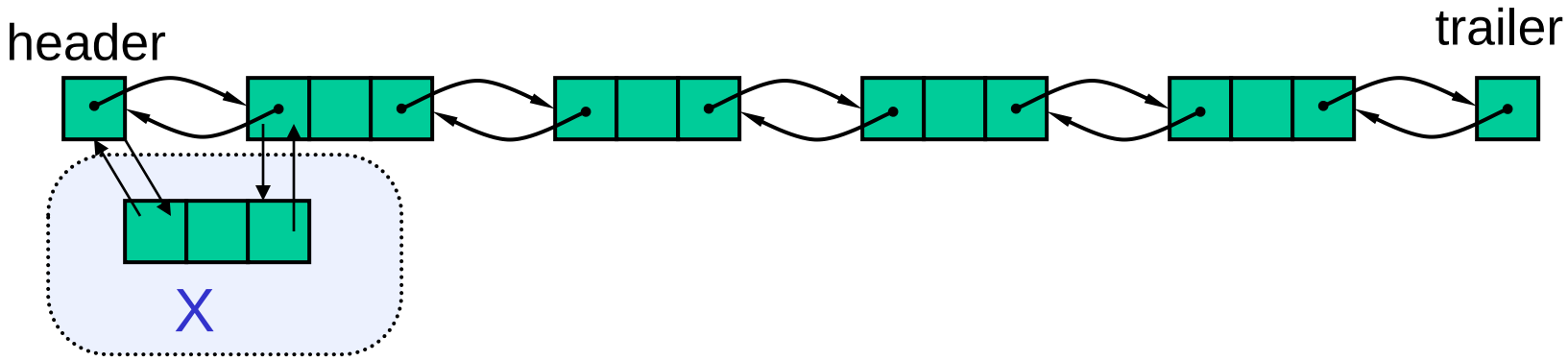


p.Prev.Next ← p.Next

p.Next.Prev ← p.Prev



Insertion (beginning)



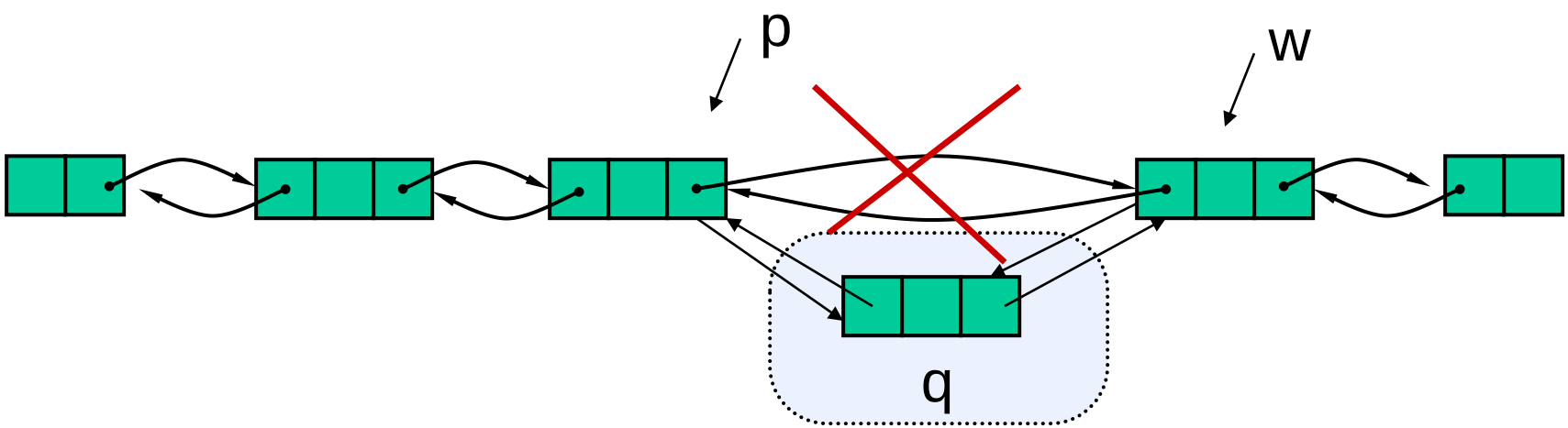
X.next ← header.next

header.next ← X

X.prev ← header

X.next.prev ← X

Insertion (after p)



addAfter(p,q)

w ← p.getNext()

q.setPrev(p)

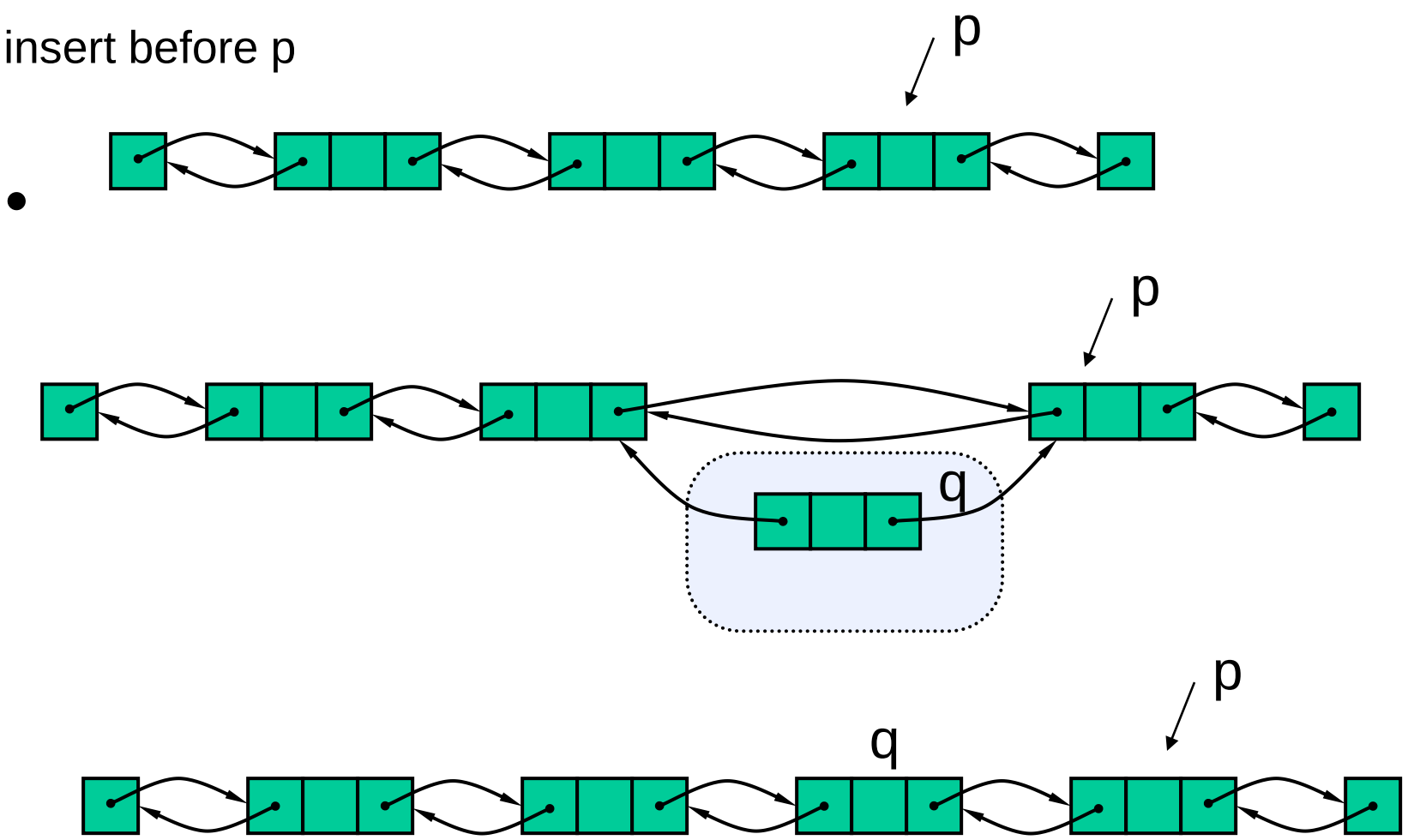
q.setNext(w)

w.setPrev(q)

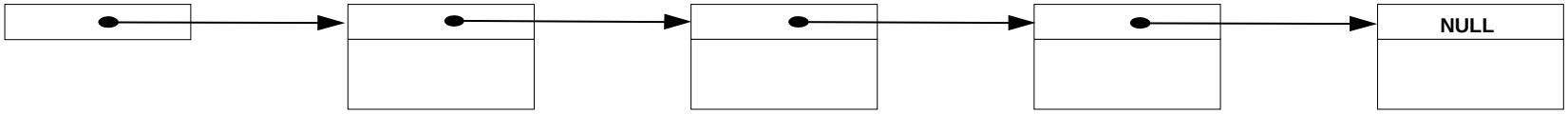
p.setNext(q)

Insertion (before p)

insert before p



Linked Structures



Dynamic structure: it is never full

No movements of elements

but

There is no DIRECT ACCESS to an element
the list has to be traversed

Java implementation - you will see it in the Labs

A node of a doubly linked list has a next and a prev link.

The doubly linked list supports methods like these:

- setElement(Object e)
- setNext(Object newNext)
- setPrev(Object newPrev)
- getElement()
- getNext()
- getPrev()