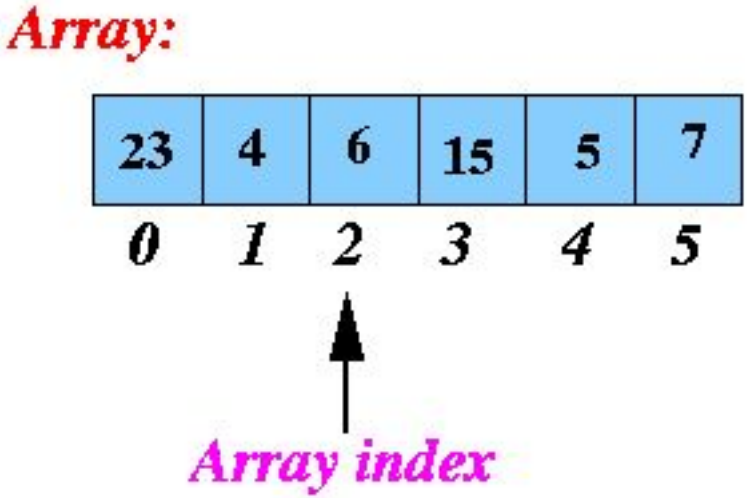
A Data Structure Bestiary

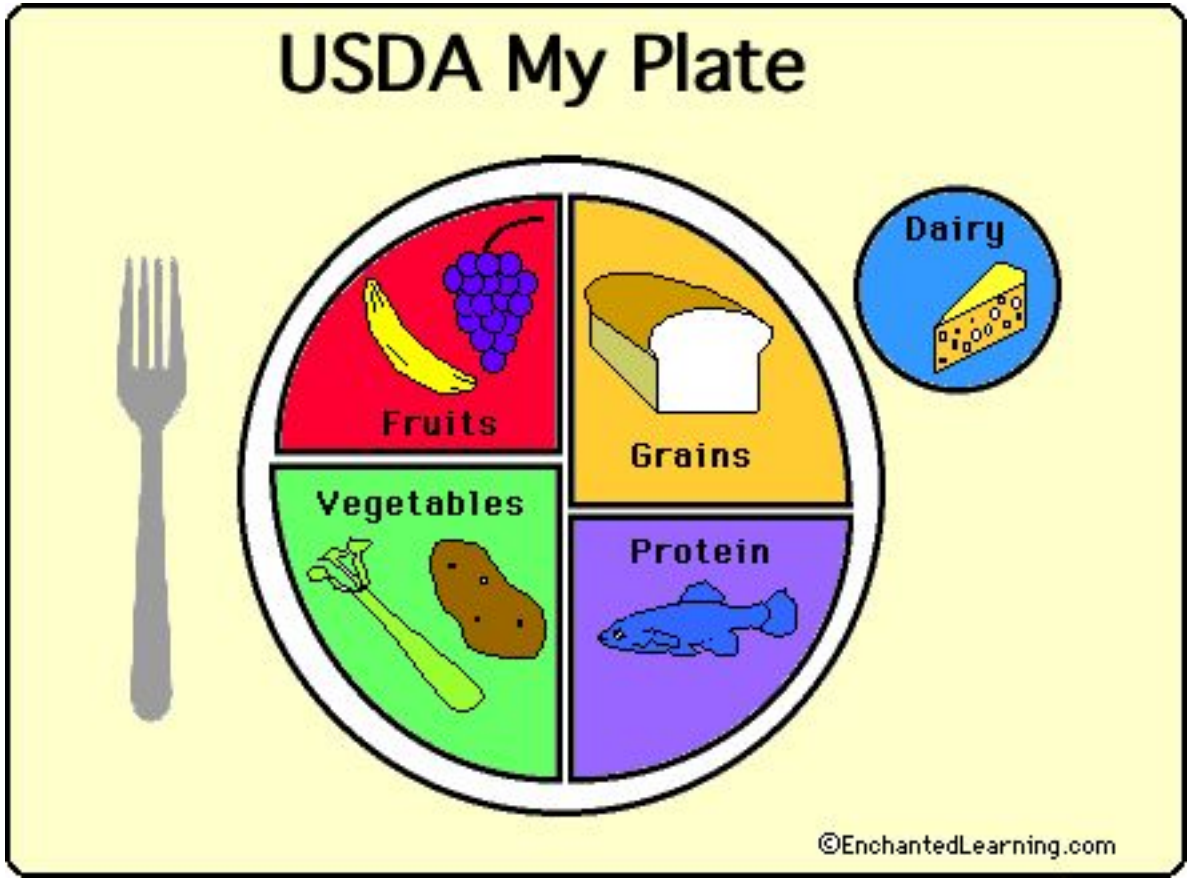
Slides adapted from David Matuszek, UPENN
https://www.cis.upenn.edu/~matuszek/cit594-2014/

Arrays

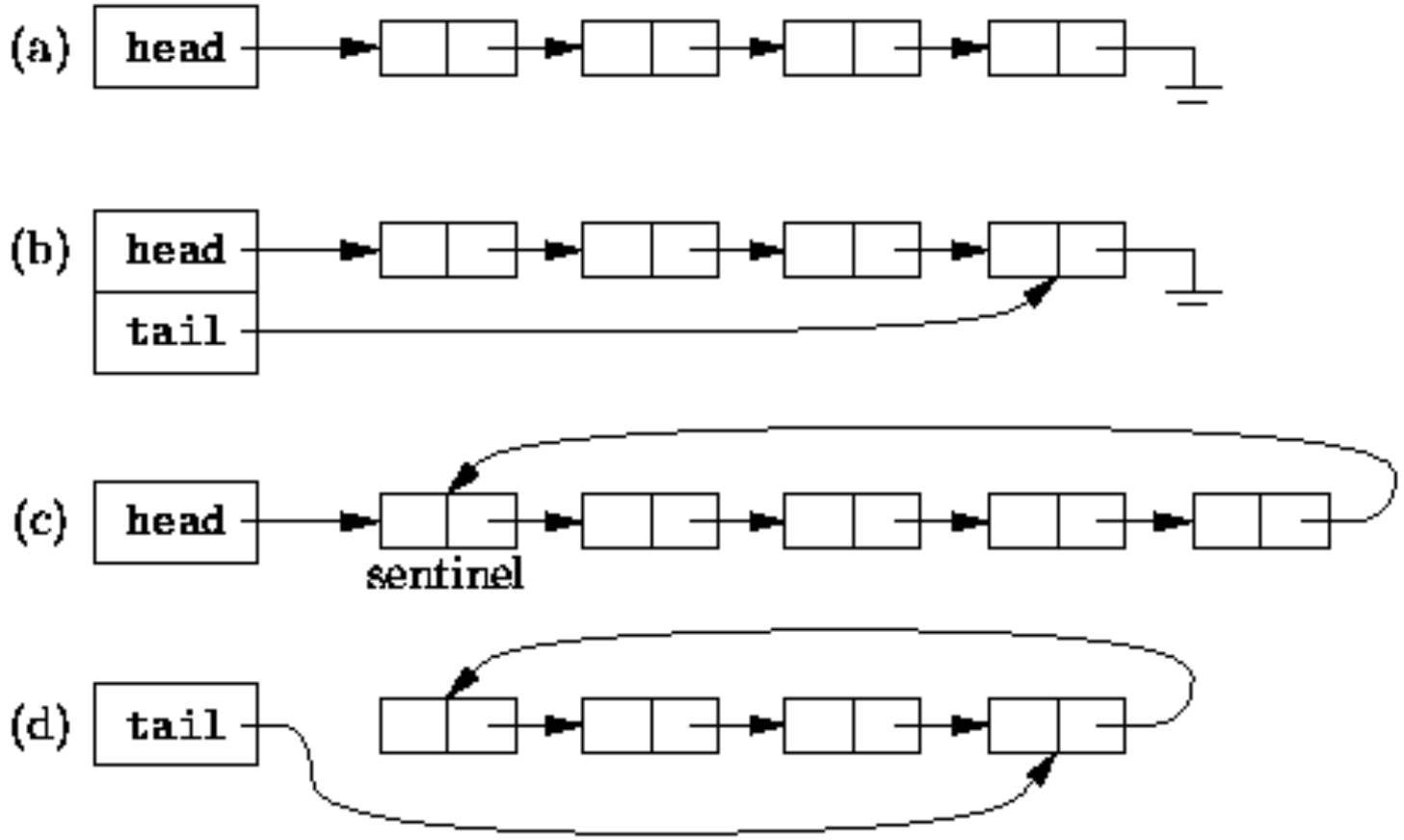


Strings

Tuples


USDA My Plate
Fruits
Grains
Dairy
Vegetables
Protein
©EnchantedLearning.com

Singly-linked lists



Simple list implementation

- Sample methods:
  - Test if list is empty
  - Look at the value in a node
  - Go to the next node (if any)
  - Add a new node to the beginning
  - Remove a node from the beginning

```
class Node<V> {
    V value;
    Node next;
    // constructor...
    // methods...
}
class List {
    Node<V> head = null;
    // methods...
}
```

Node

```
class Node<V> {
    private V value;
    private Node<V> next;

    Node(V value, Node next) {
        this.value = value;
        this.next = next;
    }

    V value() { return value; }

    boolean hasNext() { return next != null; }

    Node next() { return next; }
}
```

List

```
public class List<V> {
    private Node<V> head = null;

    boolean isEmpty() { return head == null; }

    Node<V> first() { return head; }

    void add(V val) { head = new Node<V>(val, head); }

    void remove() throws NullPointerException {
        head = head.next;
    }
}
```

Simple list implementation

- Sample methods:
  - Test if list is empty
  - Look at the value in a node
  - Go to the next node (if any)
  - Add a new node to the beginning
  - Remove a node from the beginning

```
class Node {
    int value;
    Node next;
    // constructor…
    // methods…
}
class List {
    Node head = null;
    // methods…
}
```

Node

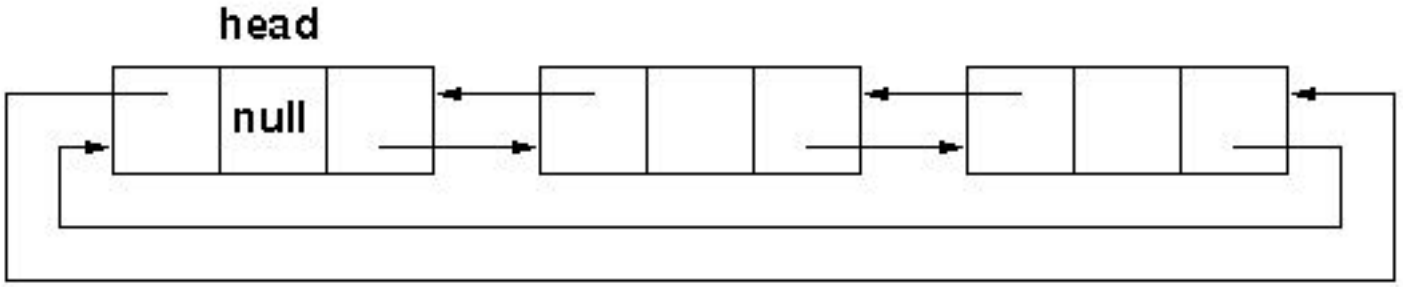
```
class Node {
    private int value;
    private Node next;

    Node(int value, Node next) {
        this.value = value;
        this.next = next;
    }

    int getVal() { return value; }

    boolean hasNext() { return next != null; }

    Node next() { return next; }
}
```

List

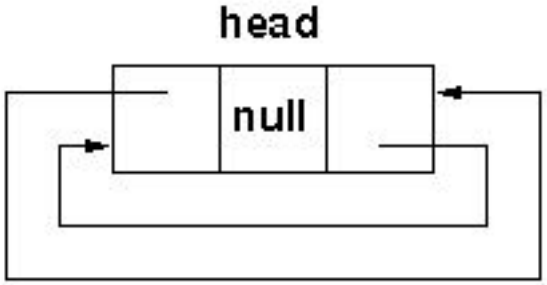
```
public class MyList {
 private Node head = null;

 boolean isEmpty() { return head == null; }

 Node first() { return head; }

 void add(int val) { head = new Node(val, head); }

 void remove() throws NullPointerException {
     head = head.next();
 }
}
```

Doubly-linked lists

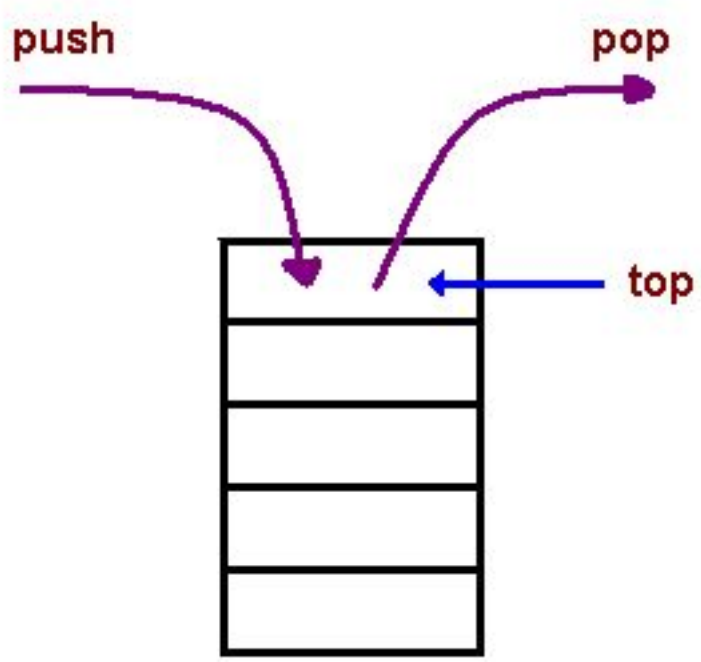
A doubly–linked list with 2 elements



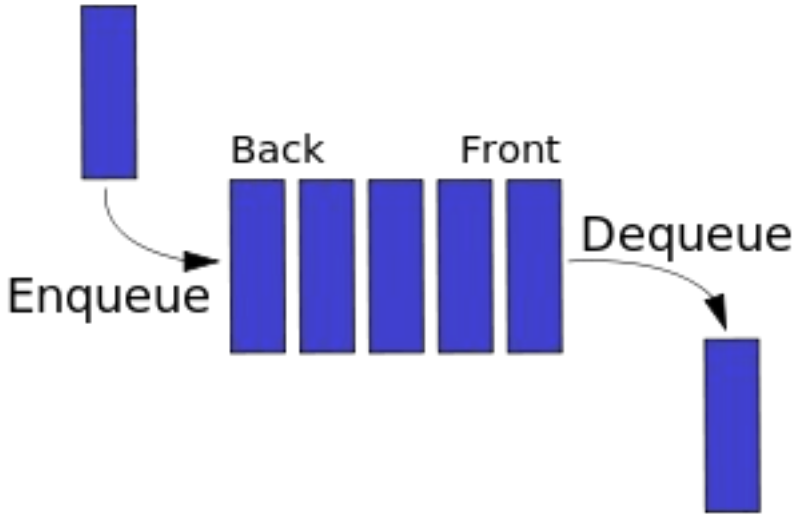
A doubly–linked list with no elements



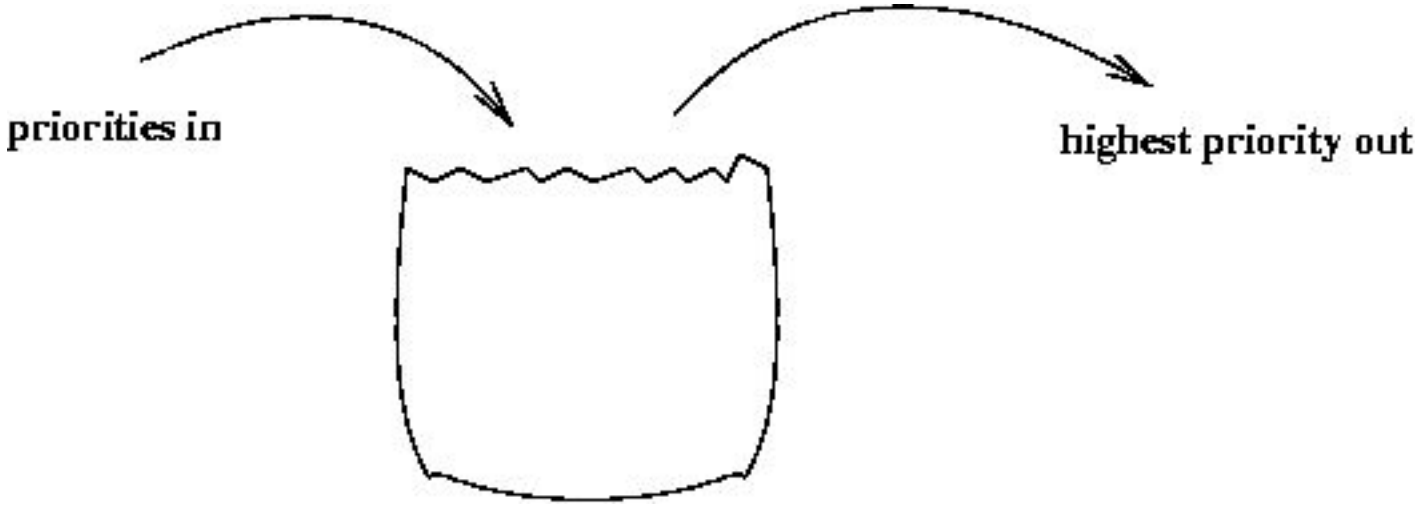
Stacks



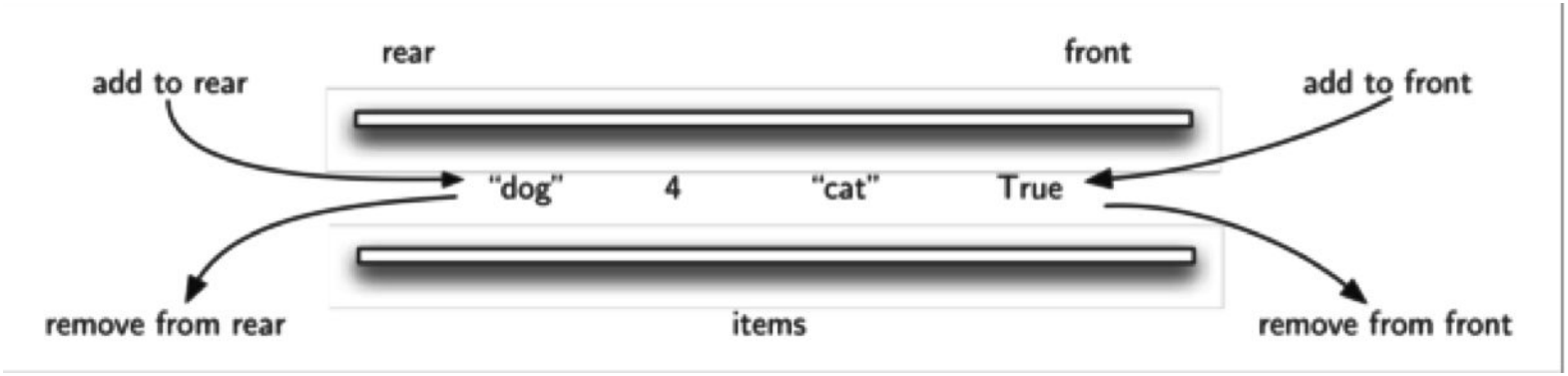
Queues



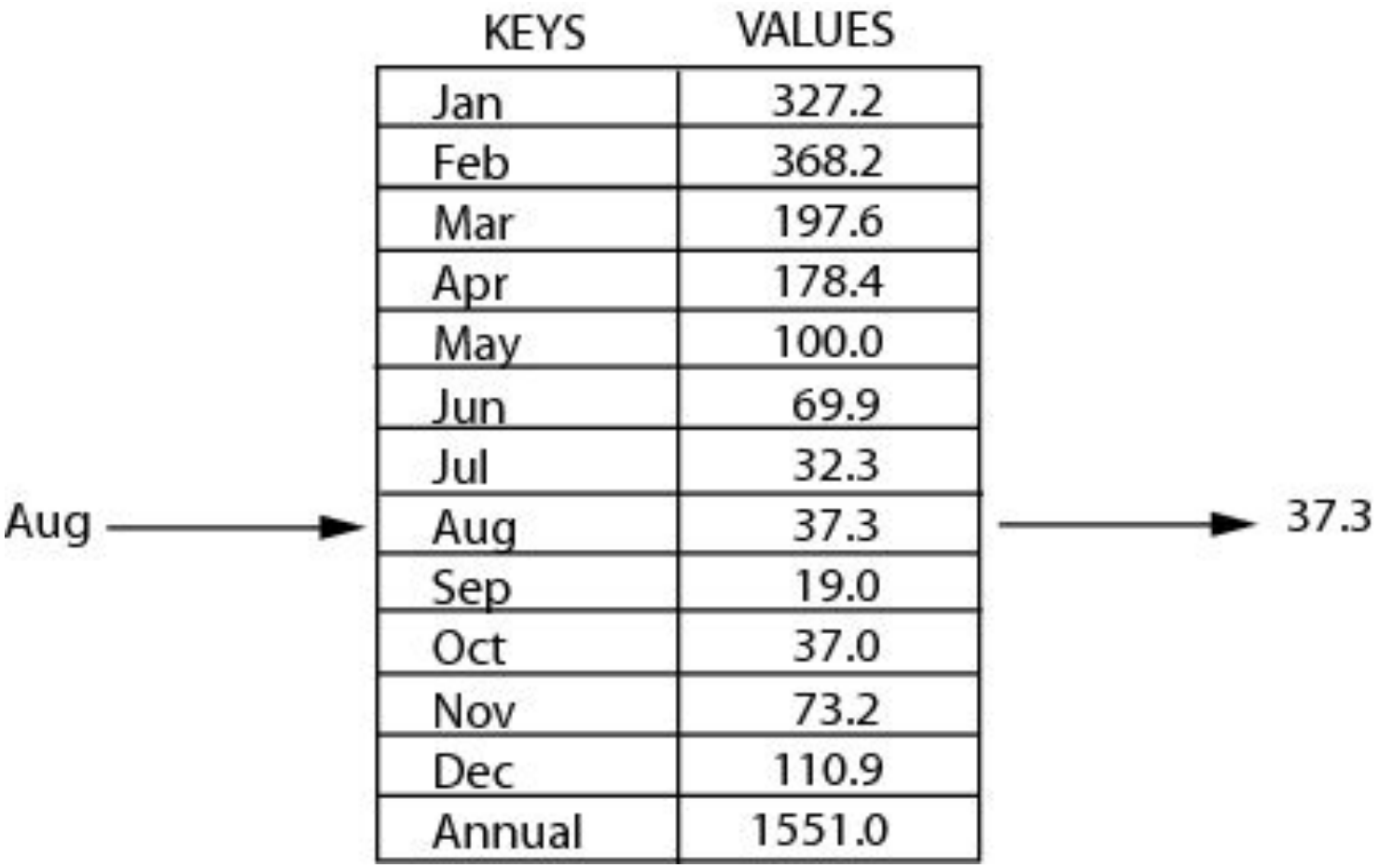
Priority queues



Deques


rear
front
add to rear
add to front
"dog"
4
"cat"
True
remove from rear
items
remove from front

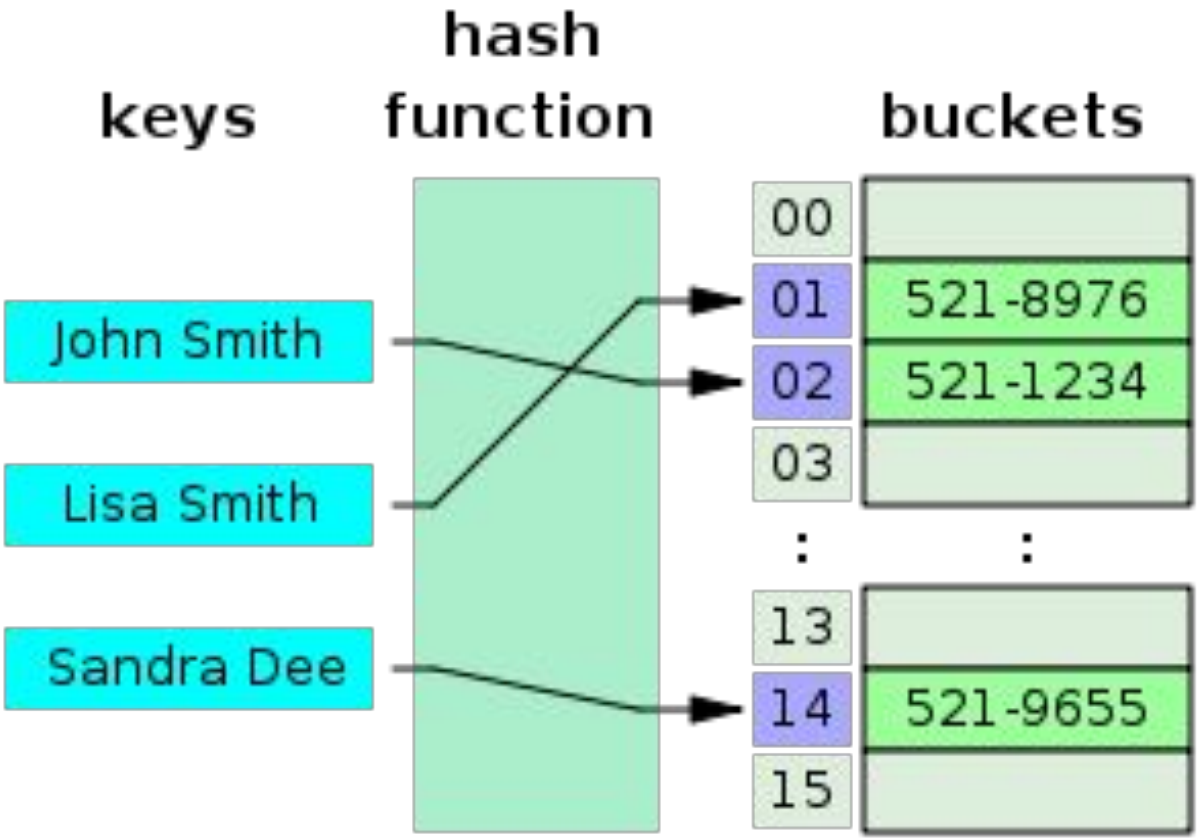
Maps


KEYS
VALUES
Jan 327.2
Feb 368.2
Mar 197.6
Apr 178.4
May 100.0
Jun 69.9
Jul 32.3
Aug 37.3
Sep 19.0
Oct 37.0
Nov 73.2
Dec 110.9
Annual 1551.0
Aug
37.3

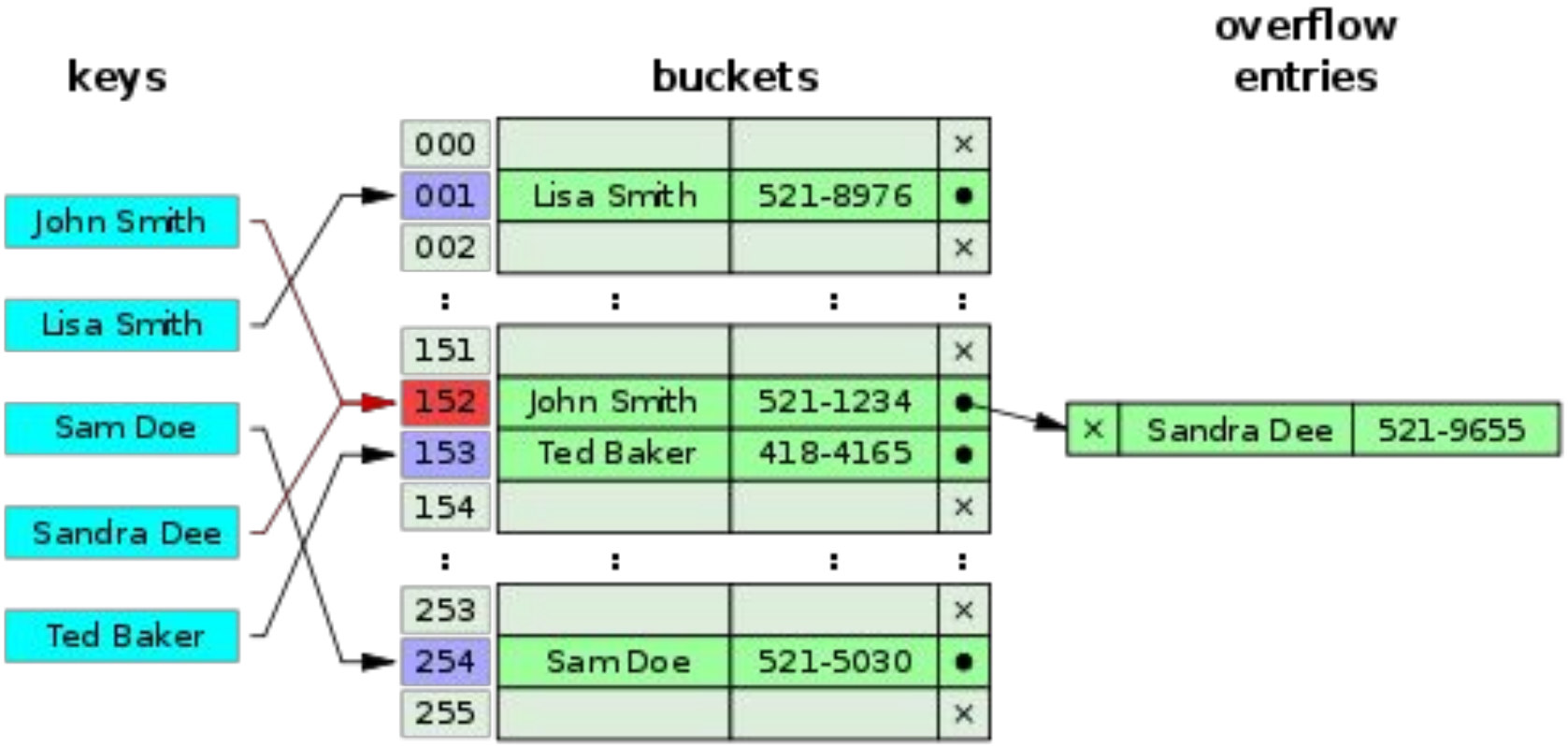
Sets


CUTIE Icon Set
20 HIGH QUALITY WEB ICONS
PNG: 48x48, 64x64, 128x128, 256x256, 512x512px Source File (AI) Included.
JUNE
16
News
created by: dryicons
exclusively for:

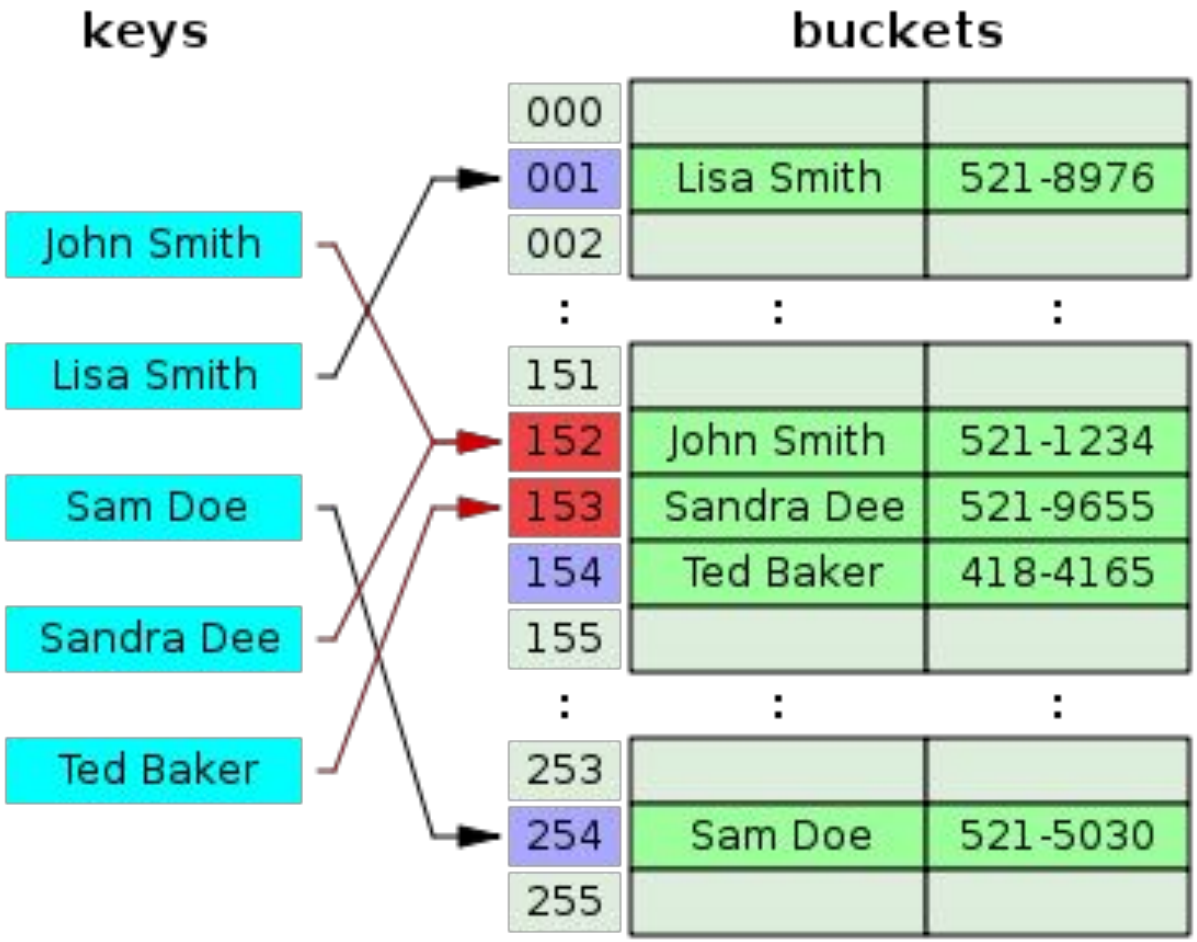
Hash tables (buckets)



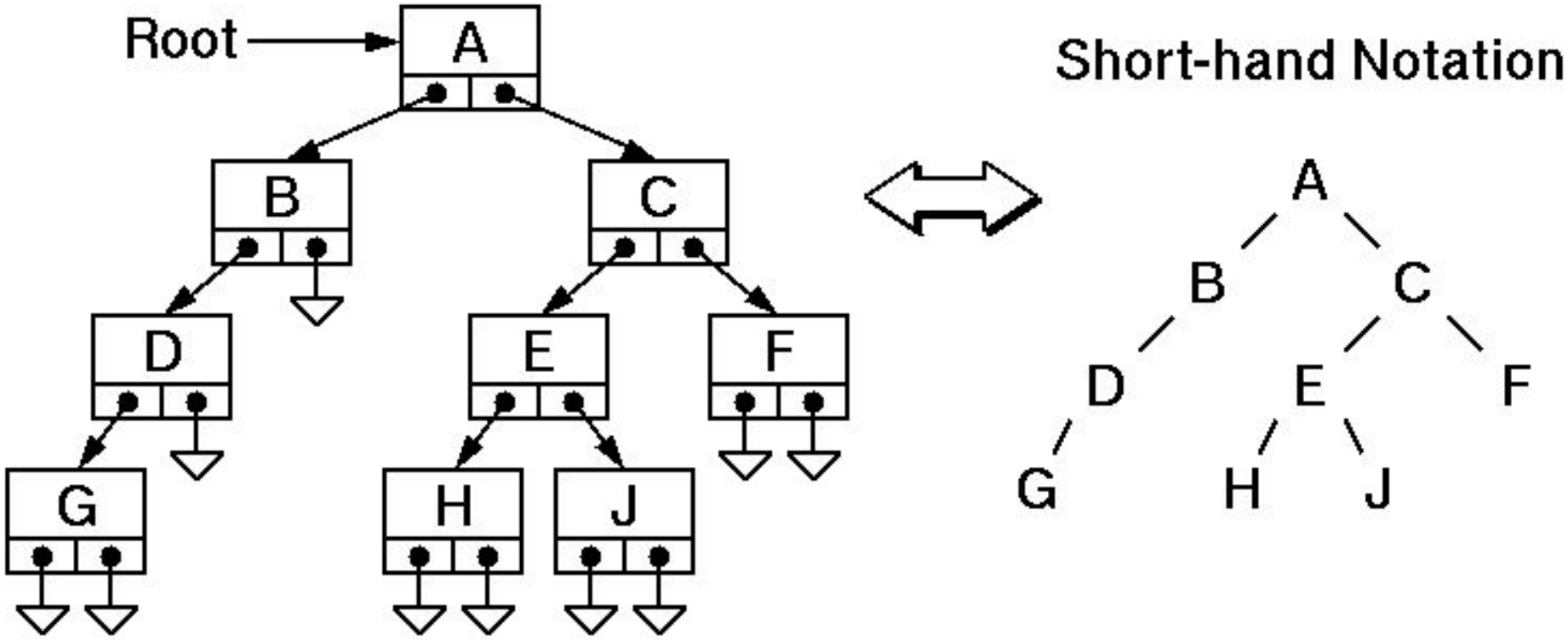
Hash maps (buckets)



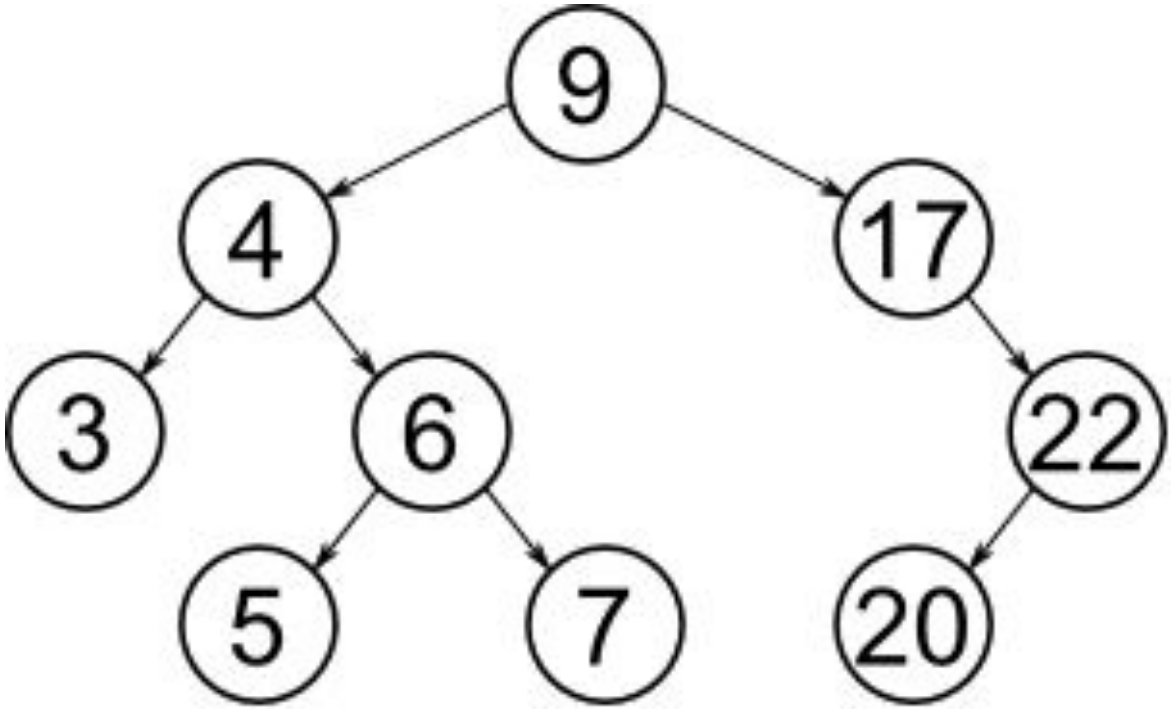
Hash maps (open addressing)



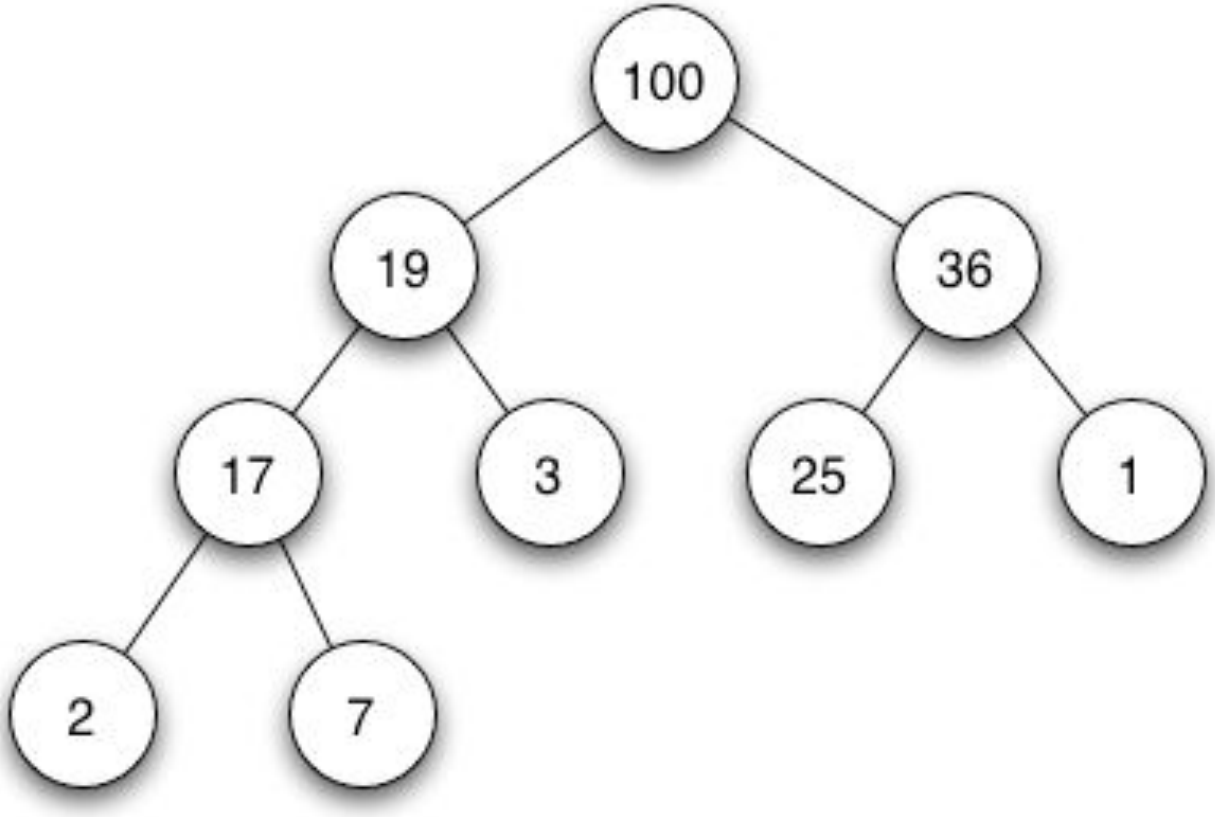
Binary trees



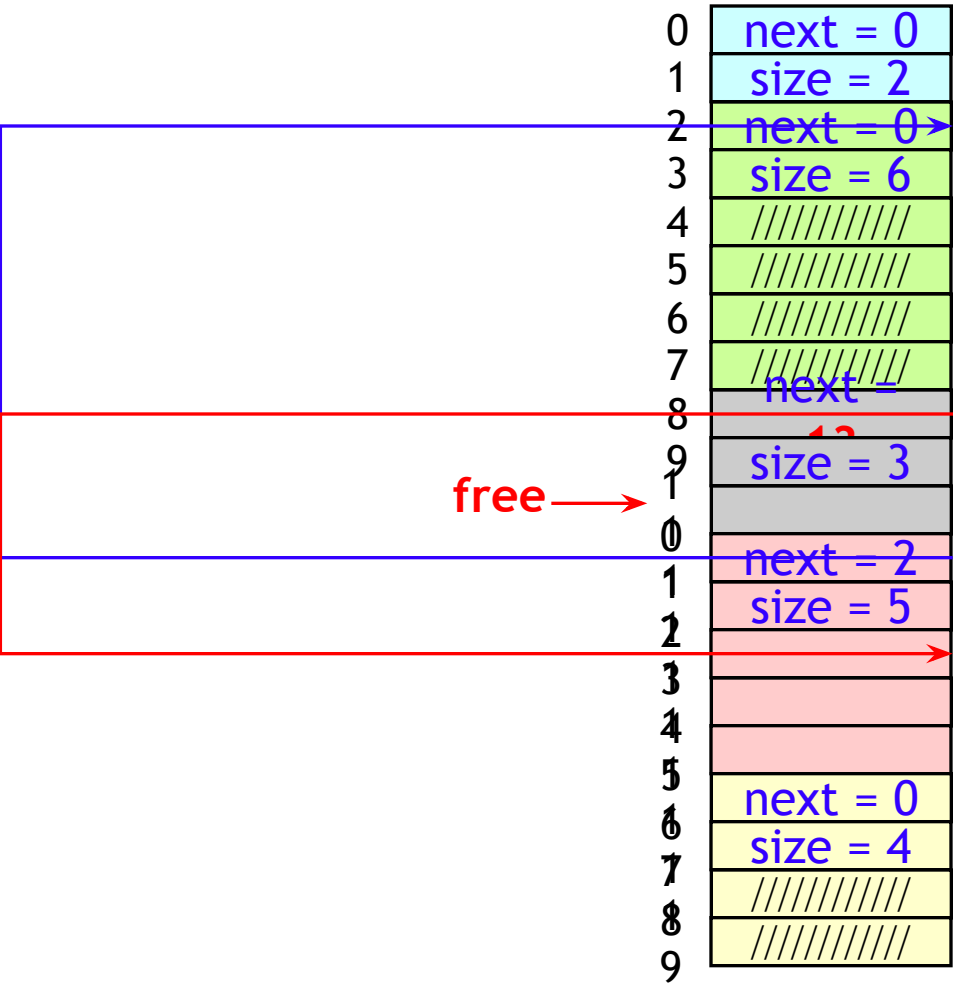
Search trees



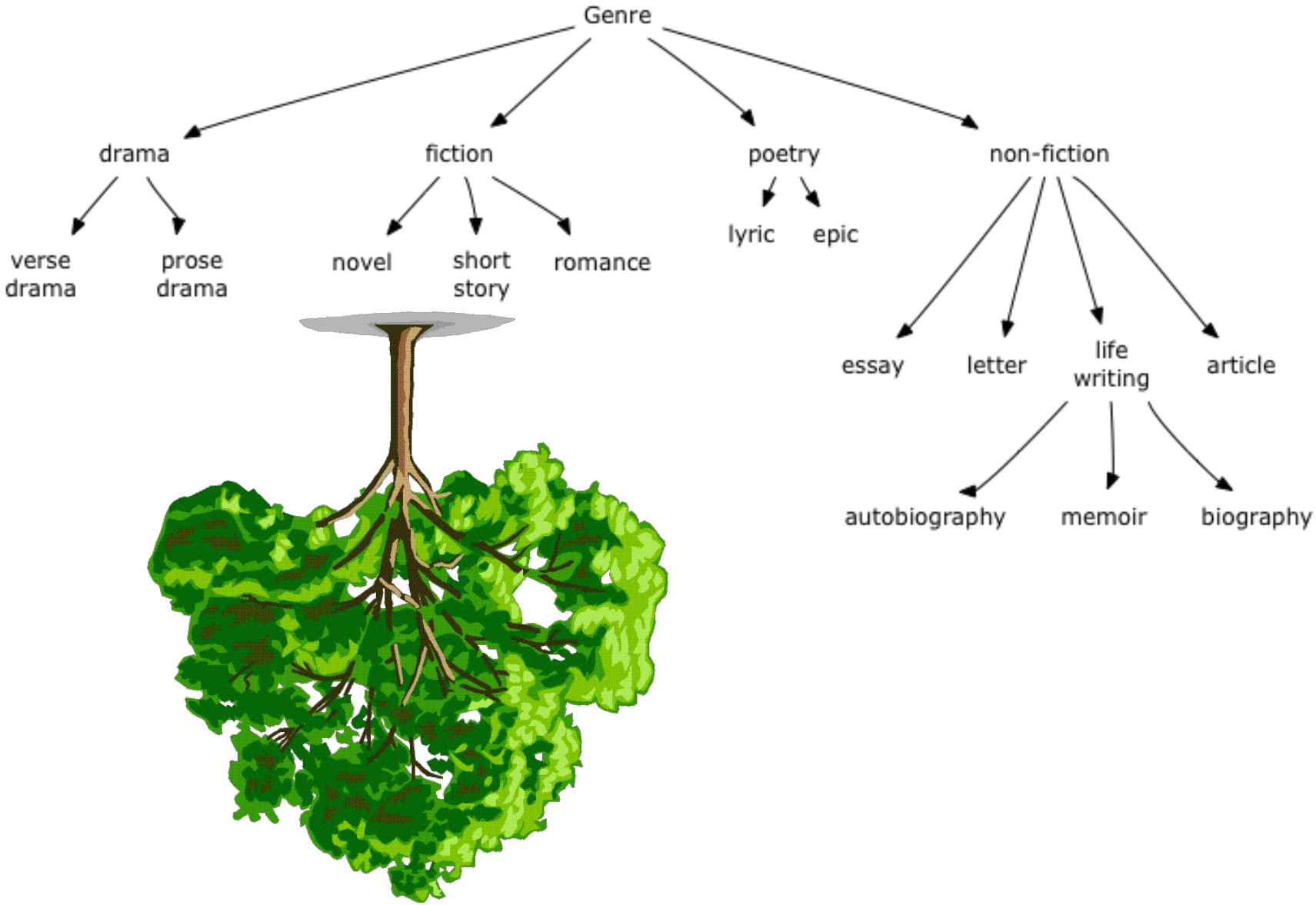
Heaps



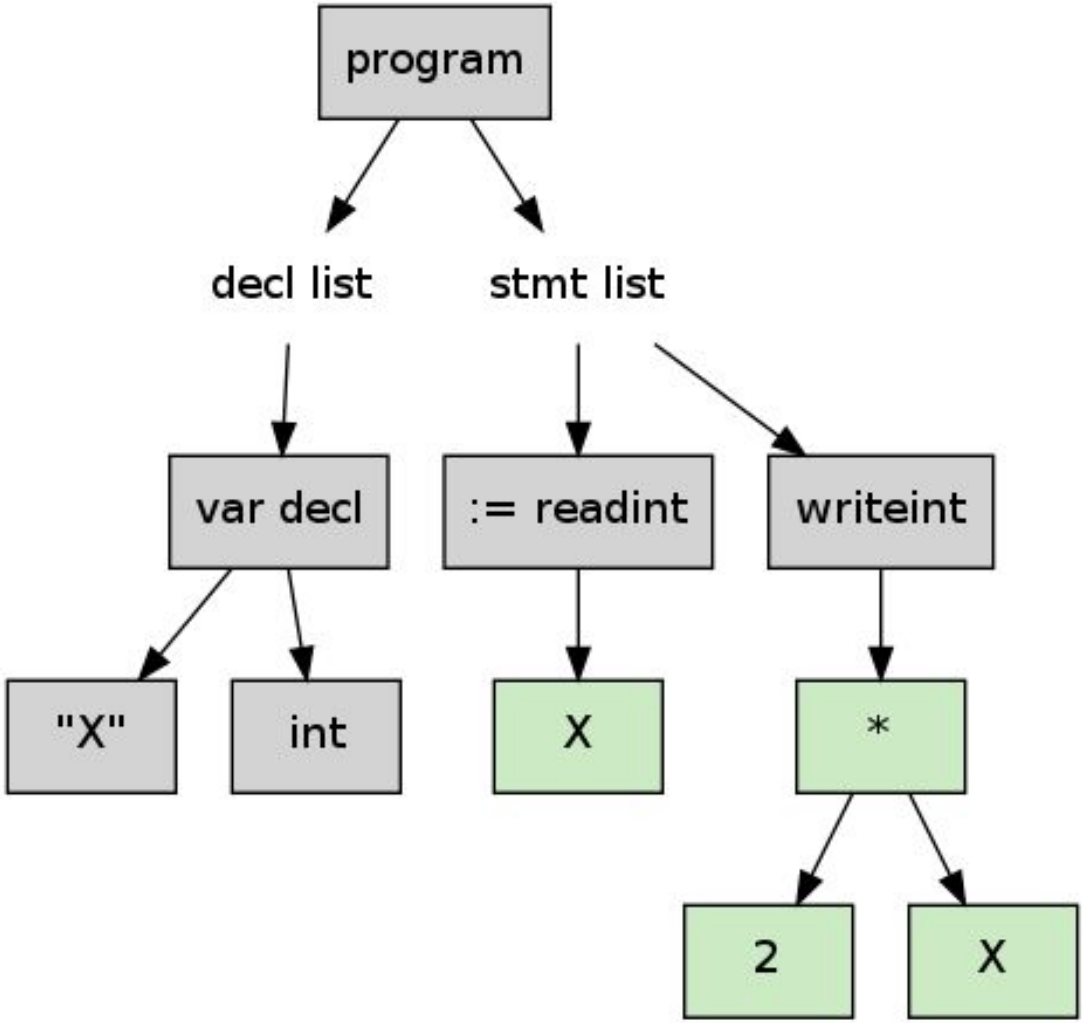
The other kind of heap



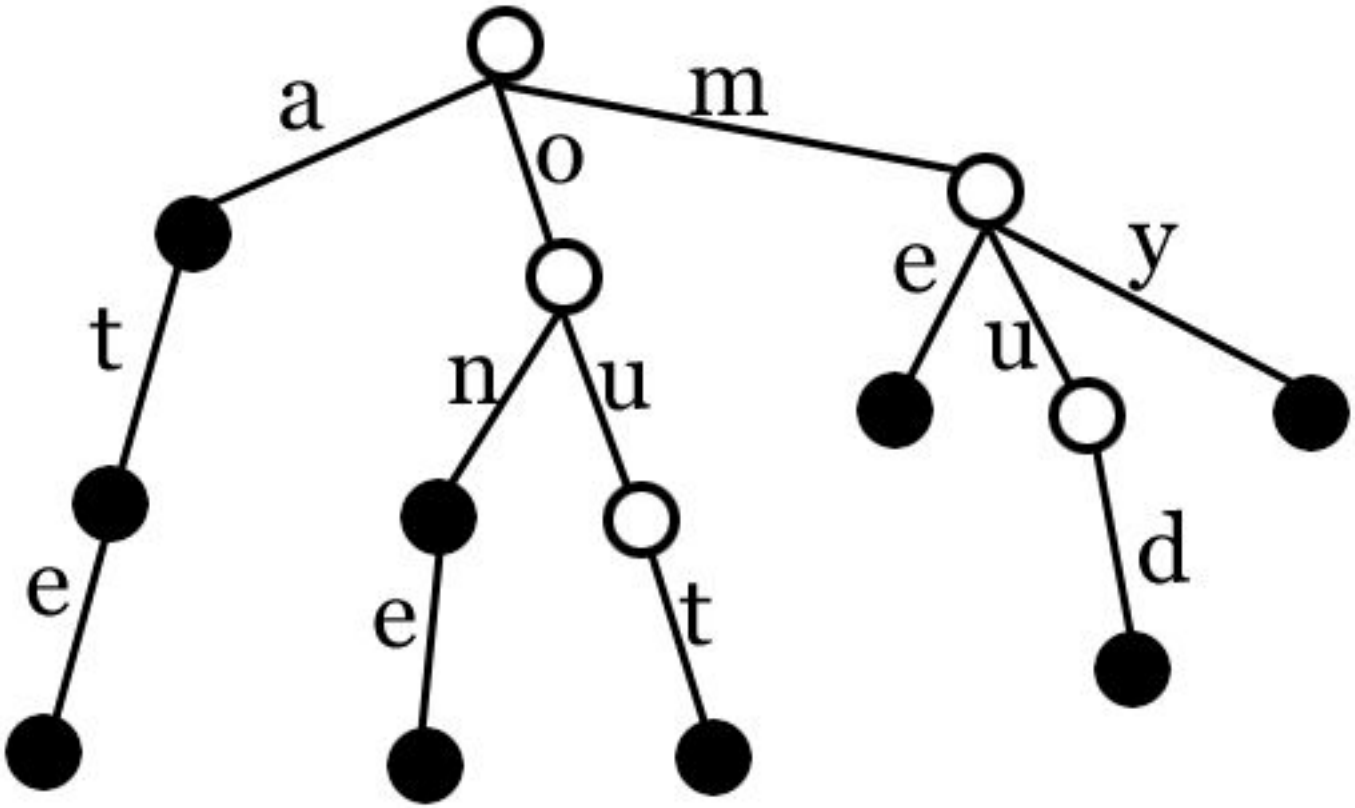
Trees


Genre
drama
fiction
poetry
non-fiction
verse drama
prose drama
novel
short story
romance
lyric
epic
essay
letter
life writing
article
autobiography
memoir
biography

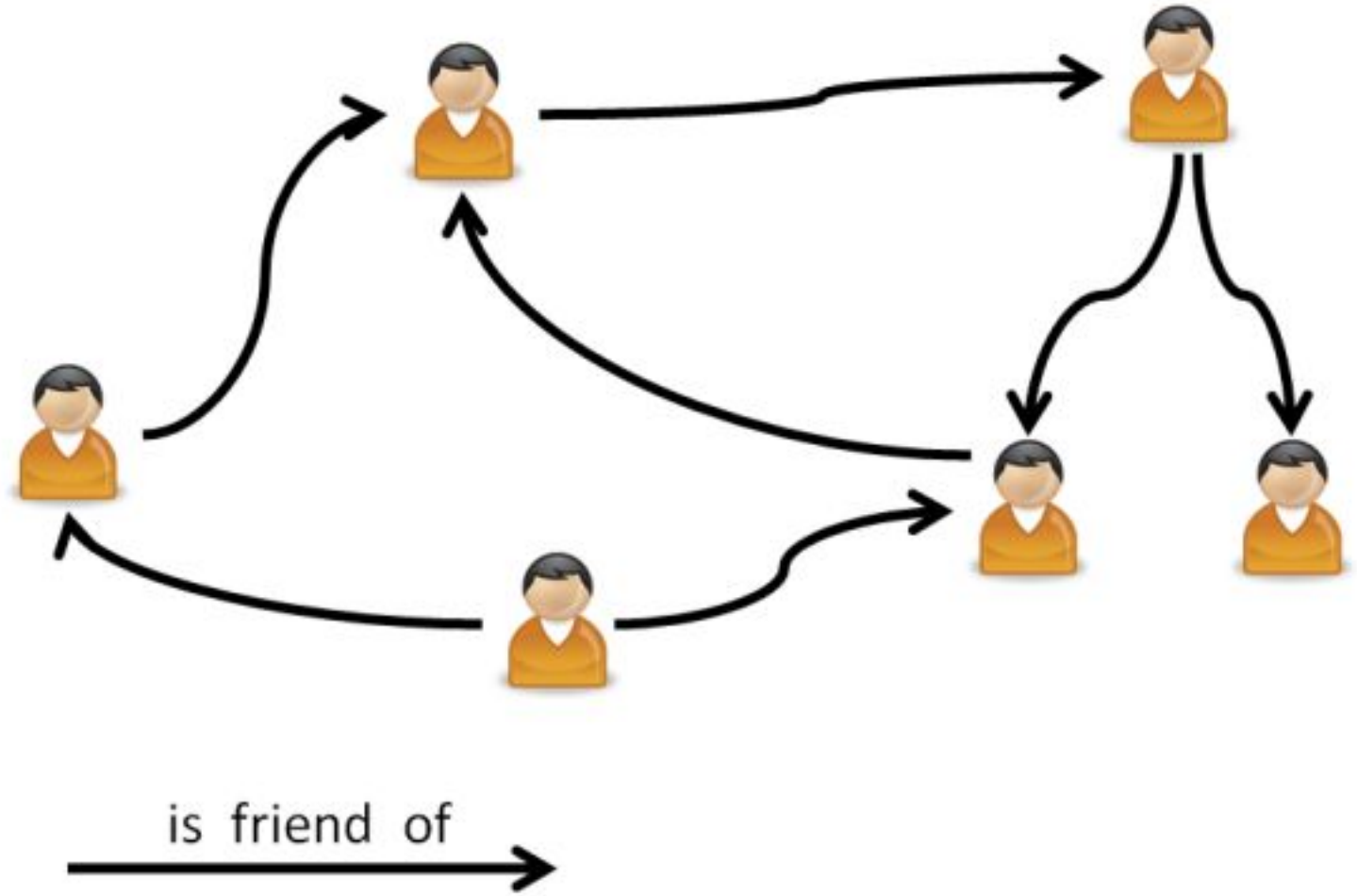
Parse trees


program
decl list
stmt list
var decl
:= readint
writeint
"X"
int
X
*
2
X

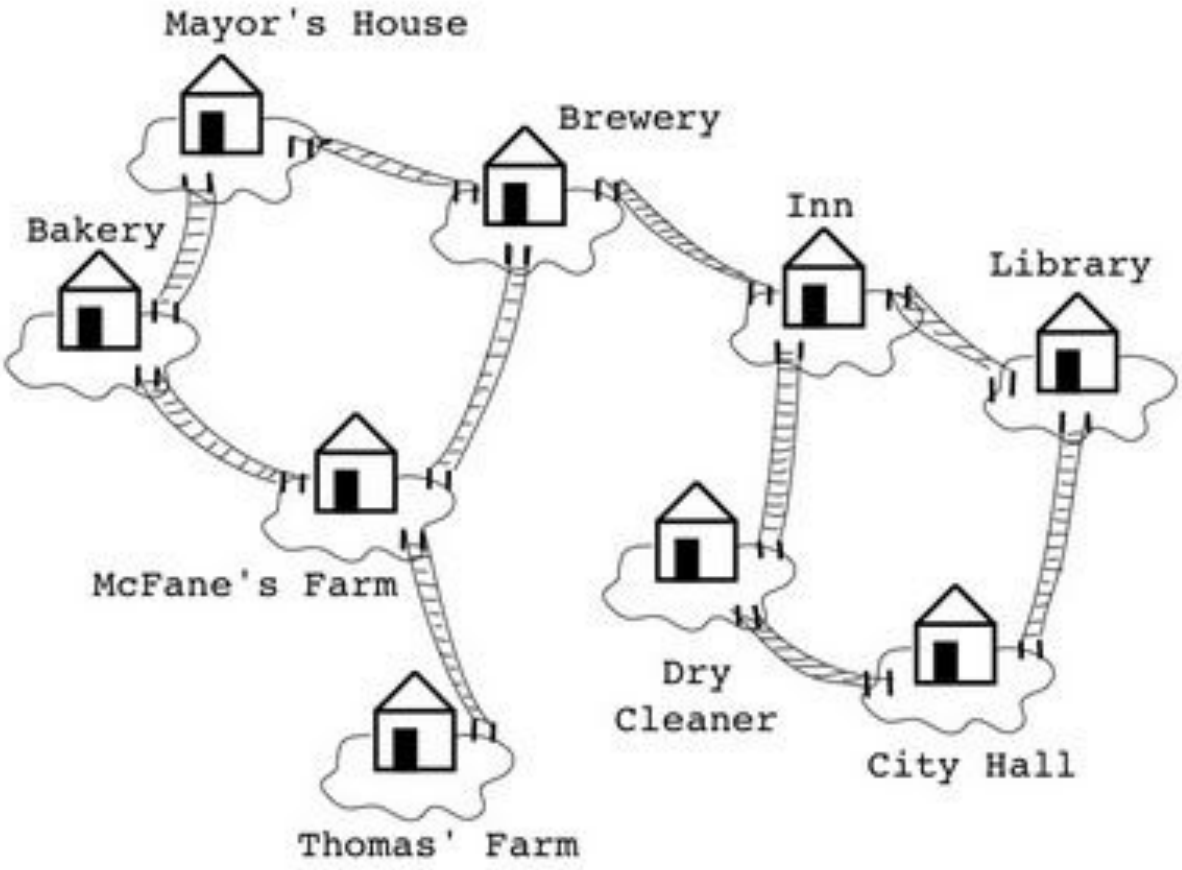
Tries



Diected graphs (Digraphs)

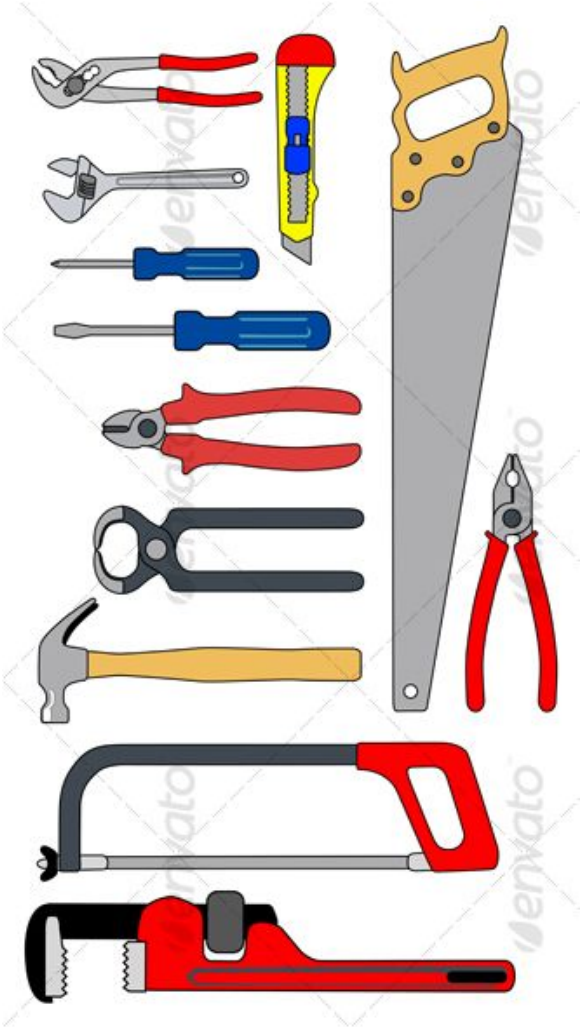


Undirected graphs



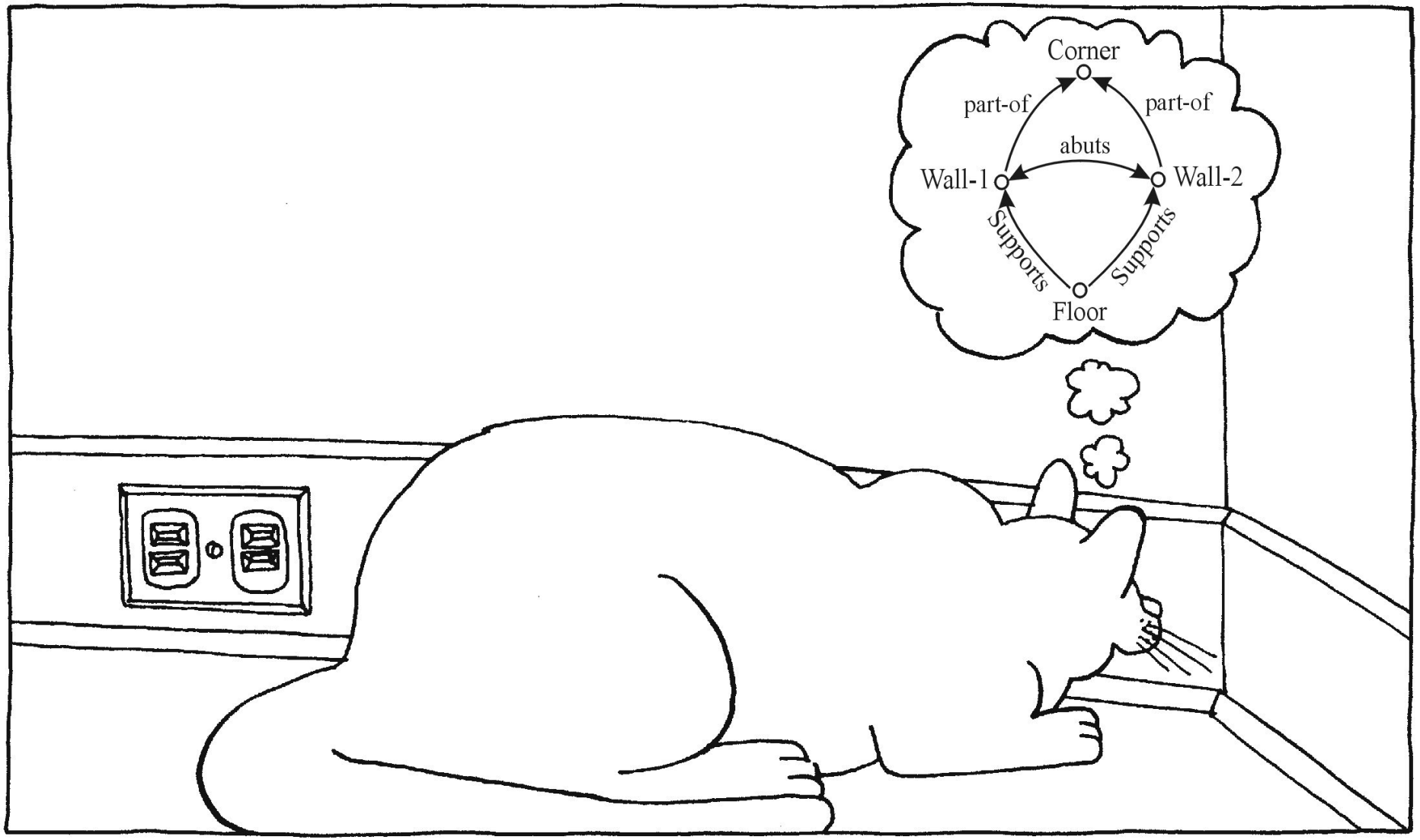
Data structures as tools

Arrays (or lists) alone



A more complete toolset

The End


Corner
part-of
part-of
abuts
Wall-1
Wall-2
Supports
Supports
Floor