Algorithms and Data Structures

Andrzej Pisarski

Plan of the lecture

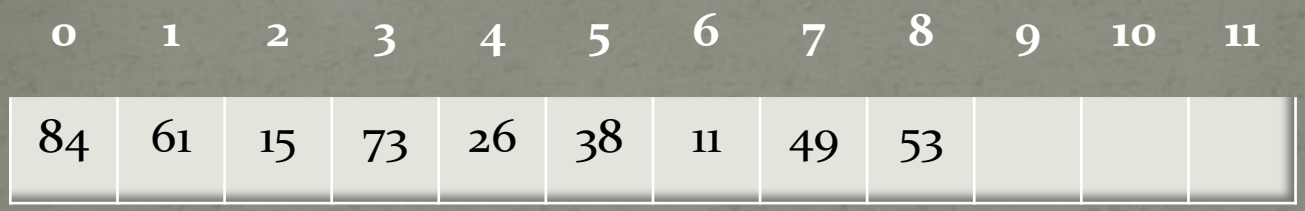
- Review of data structures
- Rapid prototyping - JavaBlock

Review of data structures

1. Arrays
2. Ordered Arrays
3. Stacks (LIFO)
4. Queses and Priority Queues (FIFO)
5. Linked Lists
   a) A (Singly-) Linked Lists
   b) Doubly Linked Lists
6. Trees
   a) Binary Trees
   b) AVL Trees
   c) Red-Black Tree
7. Hash Tables

Review of data structures

1. Arrays



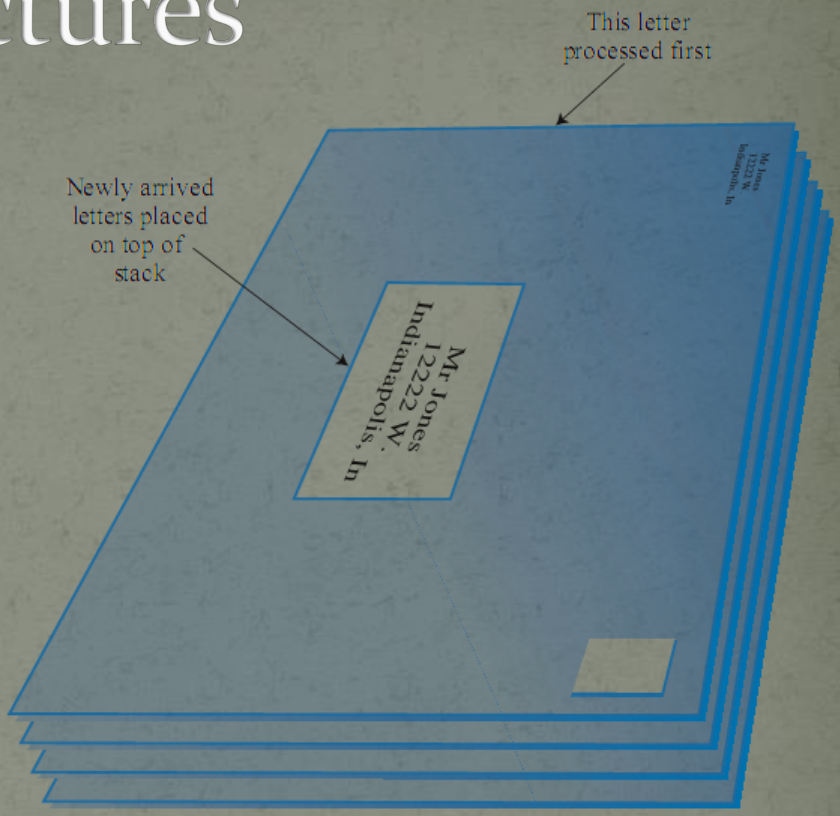
Review of data structures

2. Ordered Arrays

Review of data structures

3. Stacks (LIFO)

LIFO – last item inserted is first to be removed.

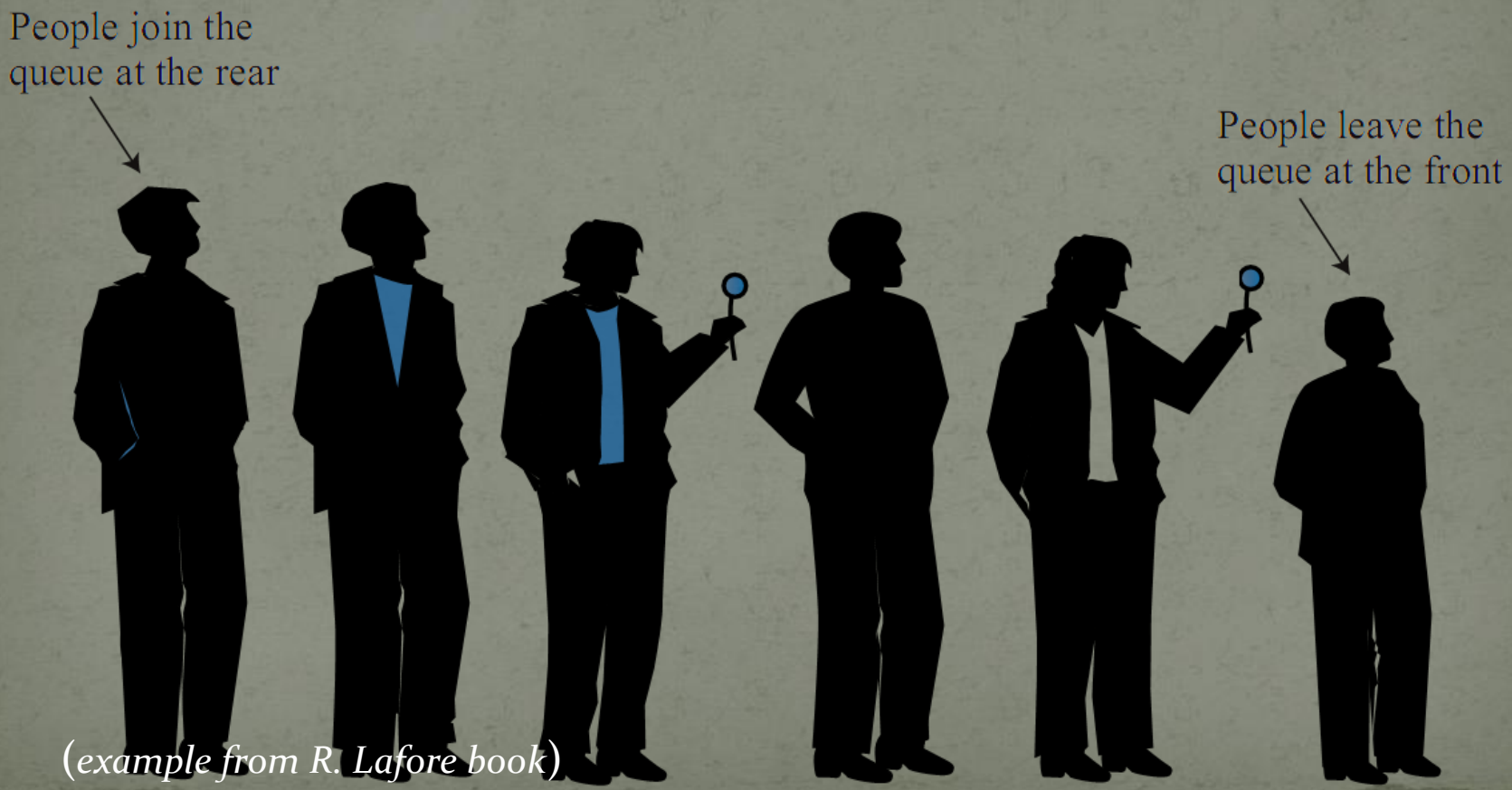


(*example from R. Lafore book*)

Review of data structures

4. a) Queses

FIFO – first item inserted is first to be removed.

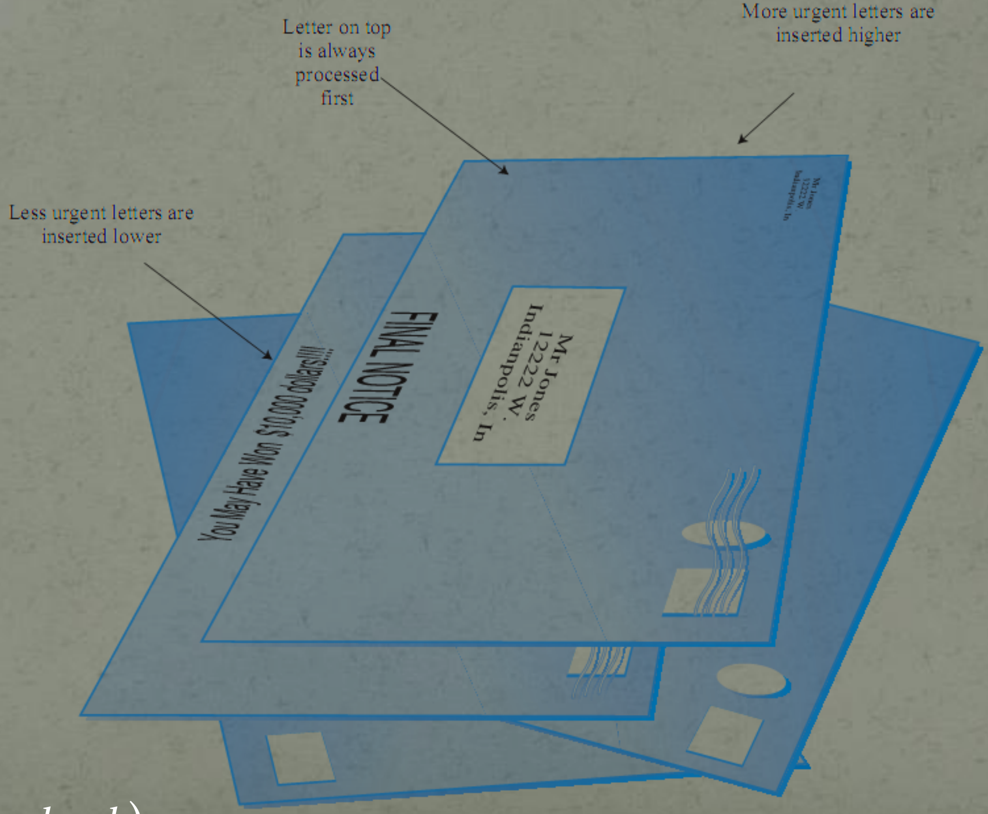


(*example from R. Lafore book*)

Review of data structures

4. b) Priority Queues

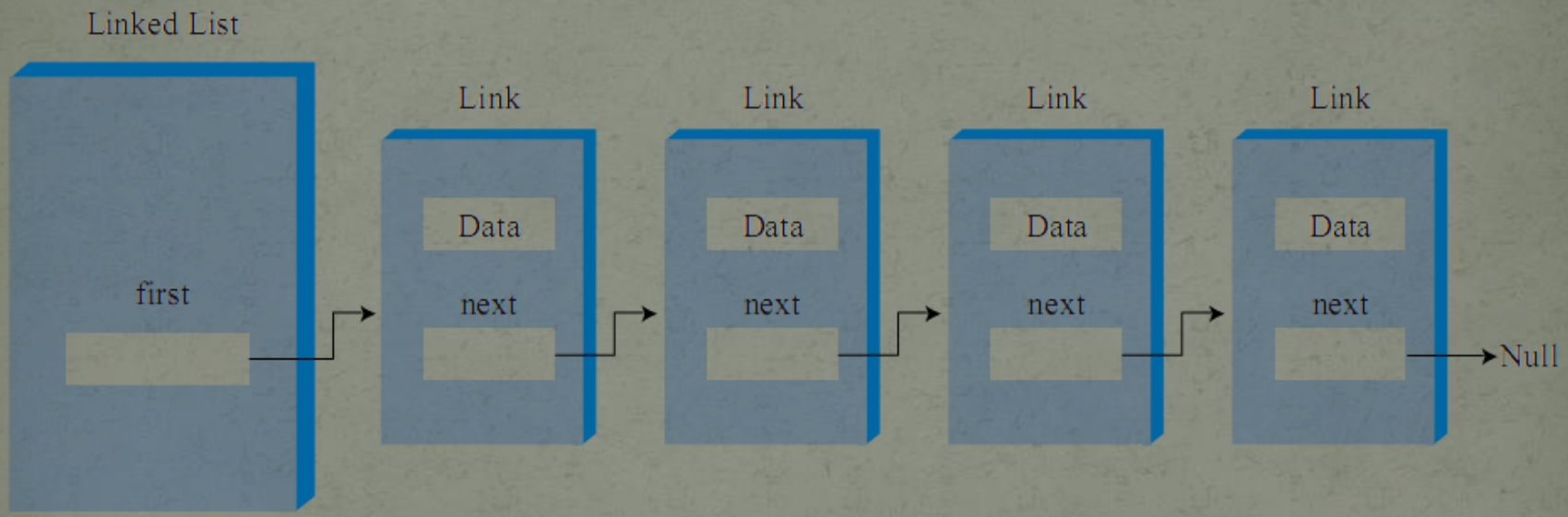
FIFO – first item inserted is first to be removed.



(*example from R. Lafore book*)

Review of data structures

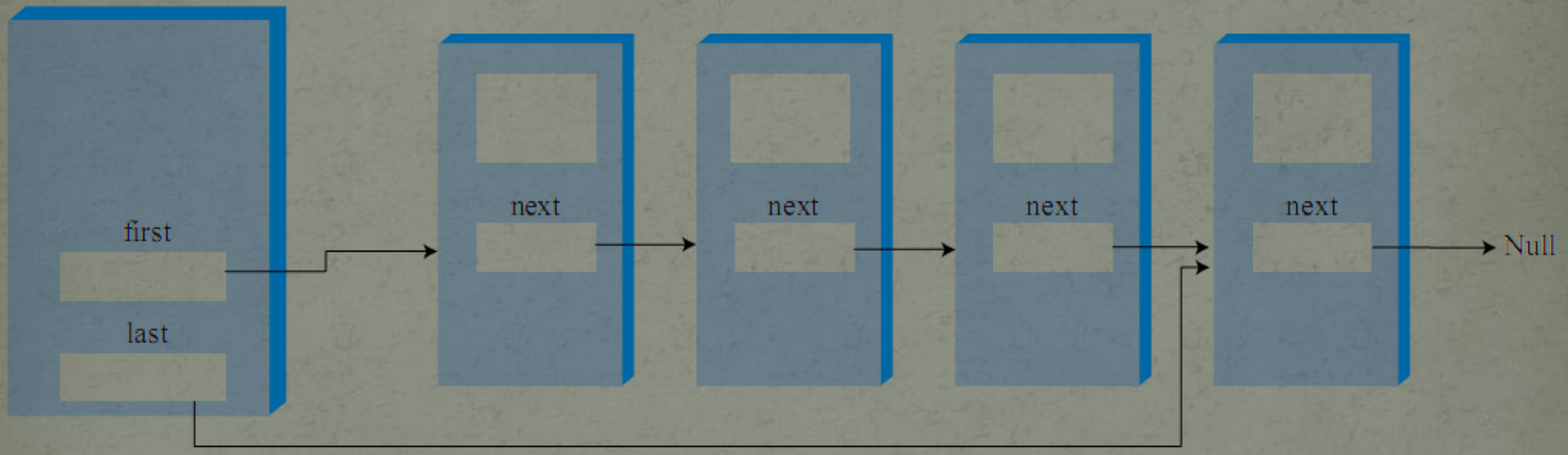
5. Linked Lists
   a) A (Singly-) Linked List: Single-ended List



(*example from R. Lafore book*)

Review of data structures

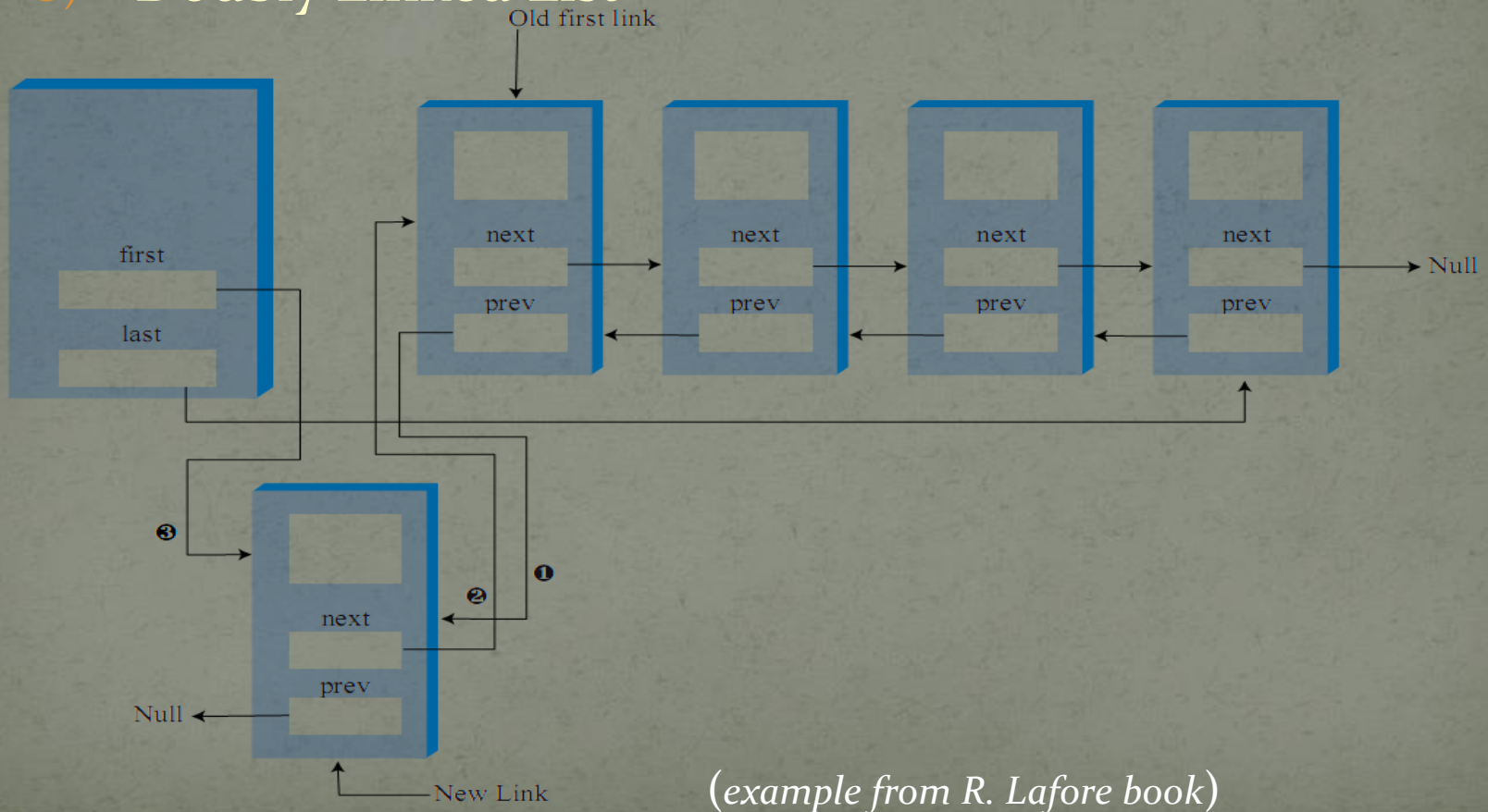
5. Linked Lists
   a) A (Singly-) Linked List: Double-ended List



(*example from R. Lafore book*)

Review of data structures

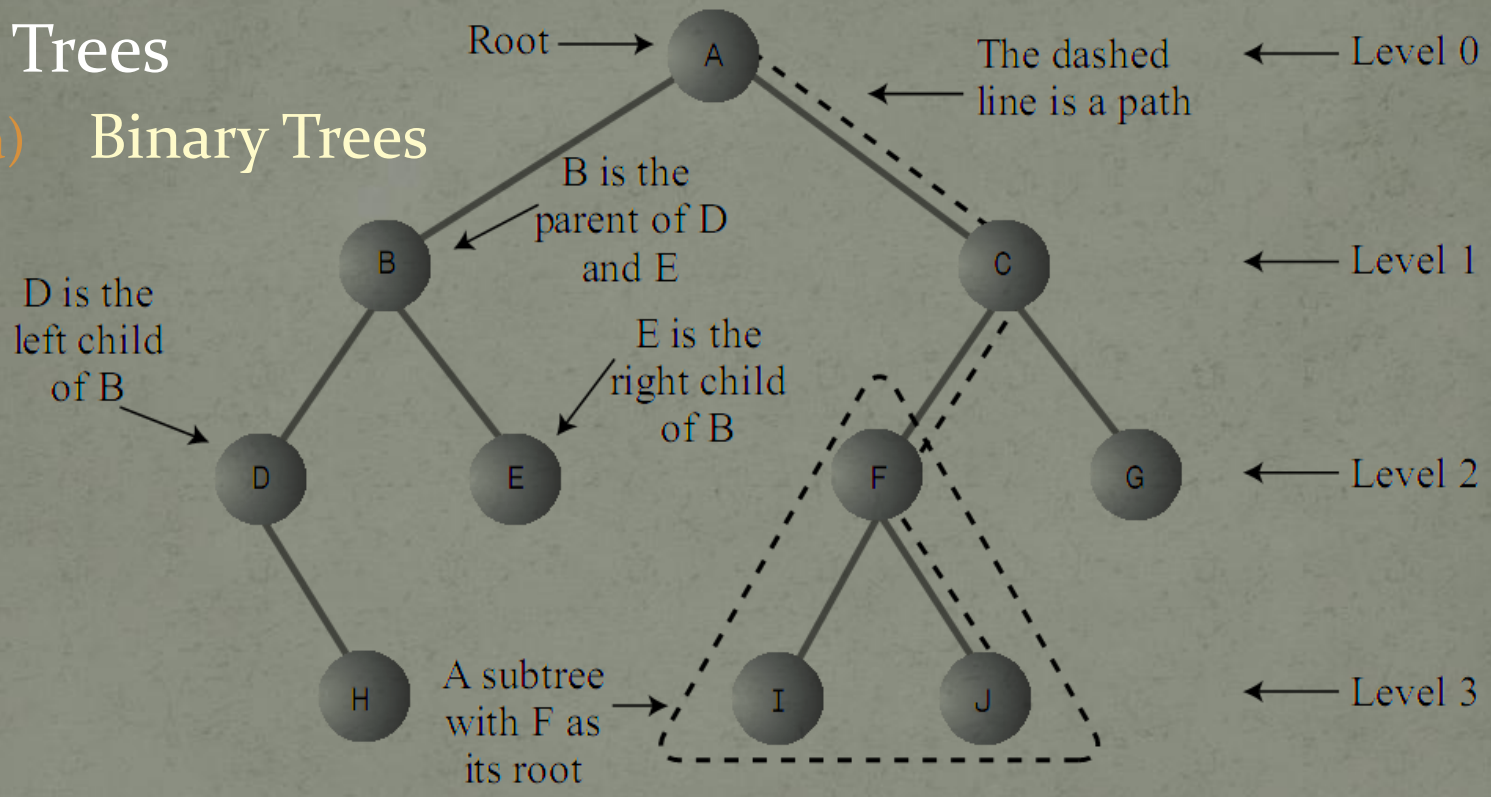
5. Linked Lists
   b) Doubly Linked List



(*example from R. Lafore book*)

Review of data structures

6. Trees
   a) Binary Trees

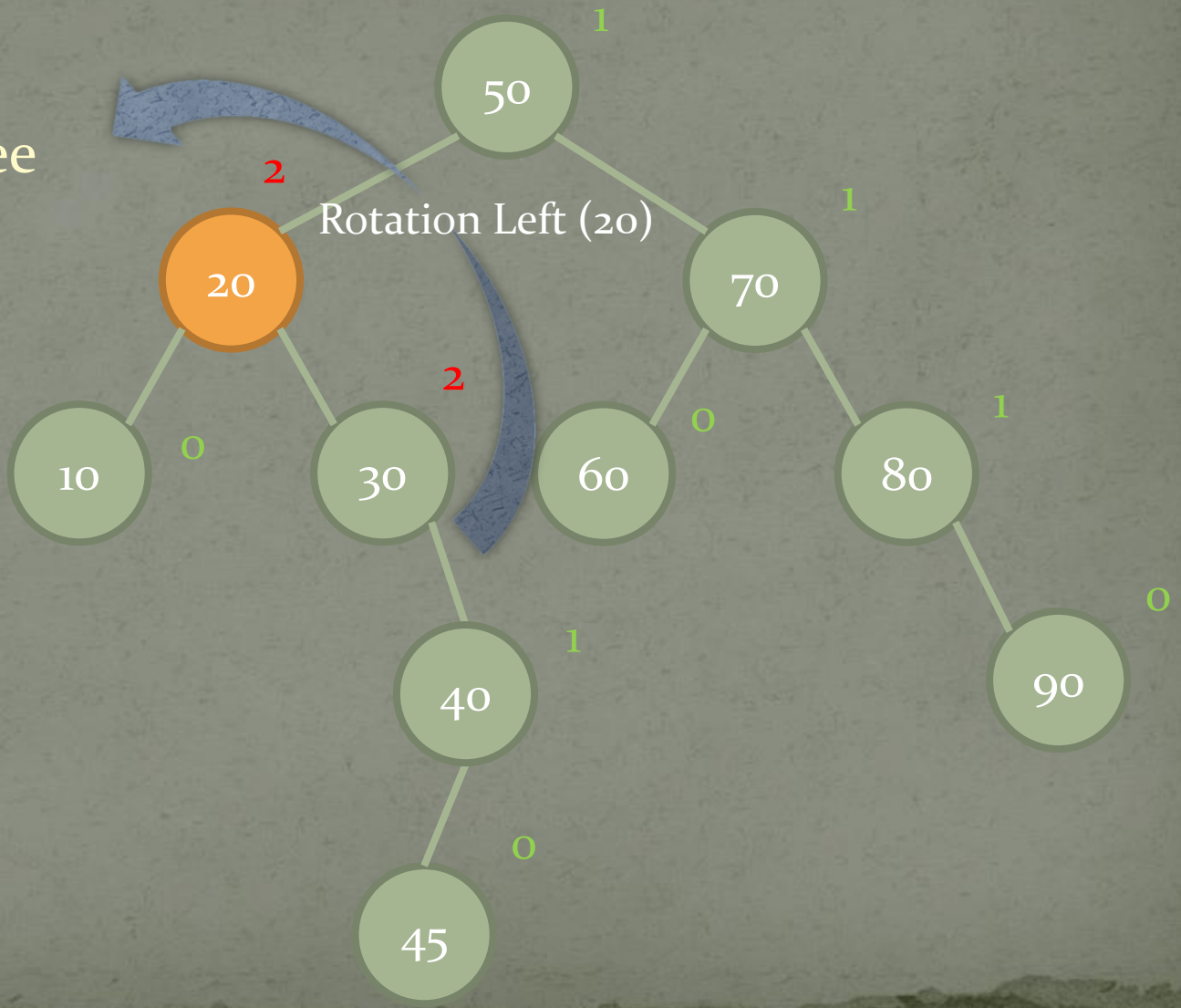


H, E, I, J, and G are leaf nodes

(*example from R. Lafore book*)

Review of data structures

6. Trees
   b) AVL Tree



Review of data structures

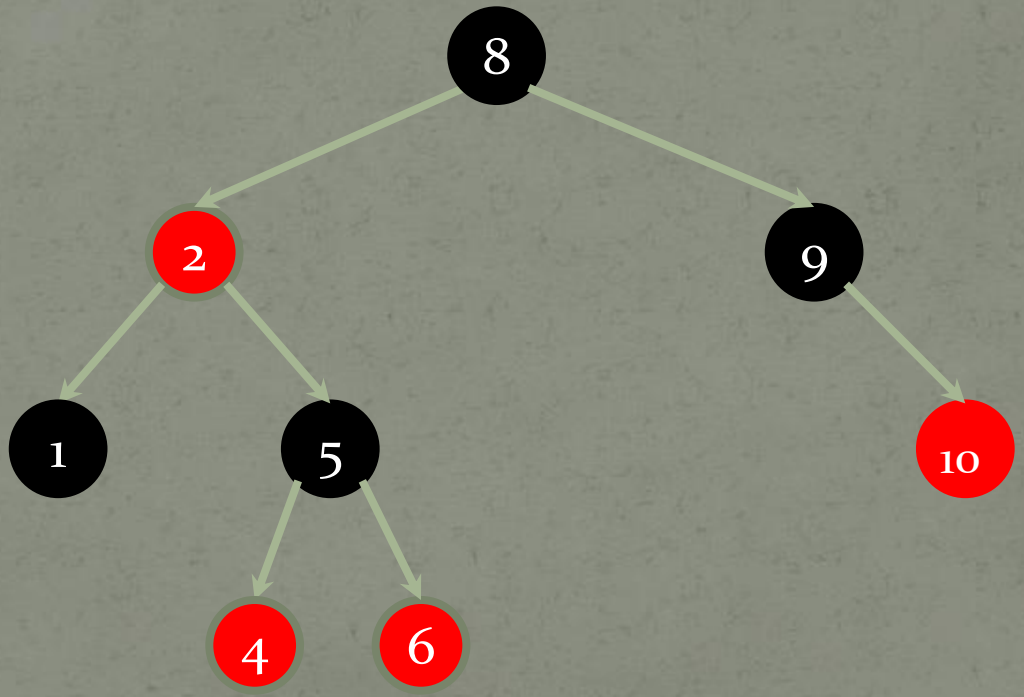
6. Trees
   c) Red-Black Tree

Node insertion or delation:

1. Each node is black or red,
2. The root is always black ,
3. Each leaf is always black ,
4. Children of red node must be black,
5. Each path from the root to leaf must contain the same number of black nodes.

Review of data structures

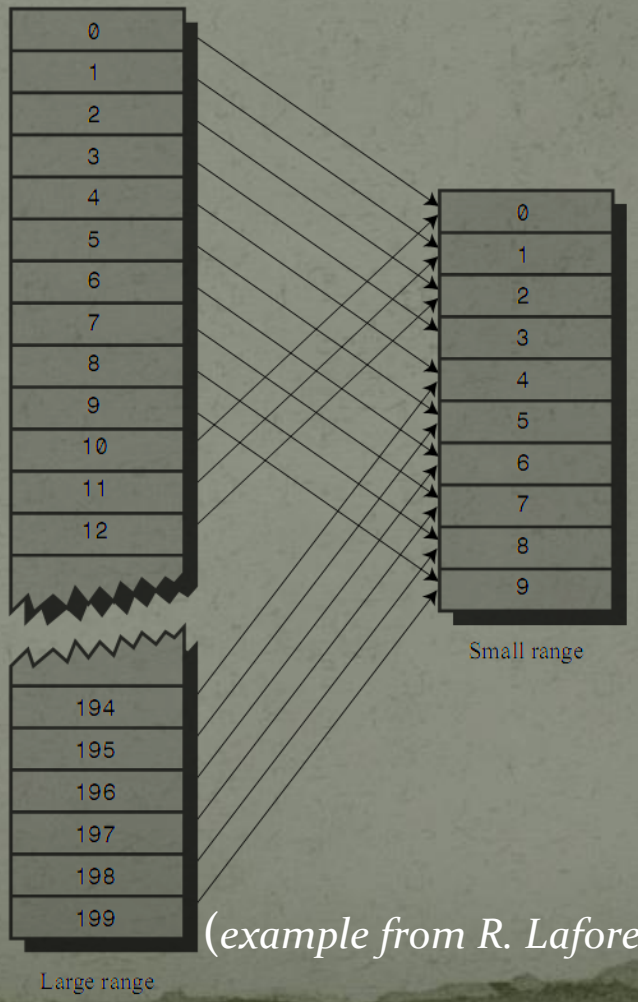
6. Trees
   c) Red-Black Tree



Review of data structures

7. Hash Tables

hugeNumber = 26*27^9 + 26*27^8 + …
+ 26*27^1 + 26*27^0
arraySize = numberWords * 2;
arrayIndex = hugeNumber % arraySize;
(hash function)



(*example from R. Lafore book*)

Rapid prototyping - JavaBlock

- Why to use a JavaBlock?

Rapid prototyping - JavaBlock

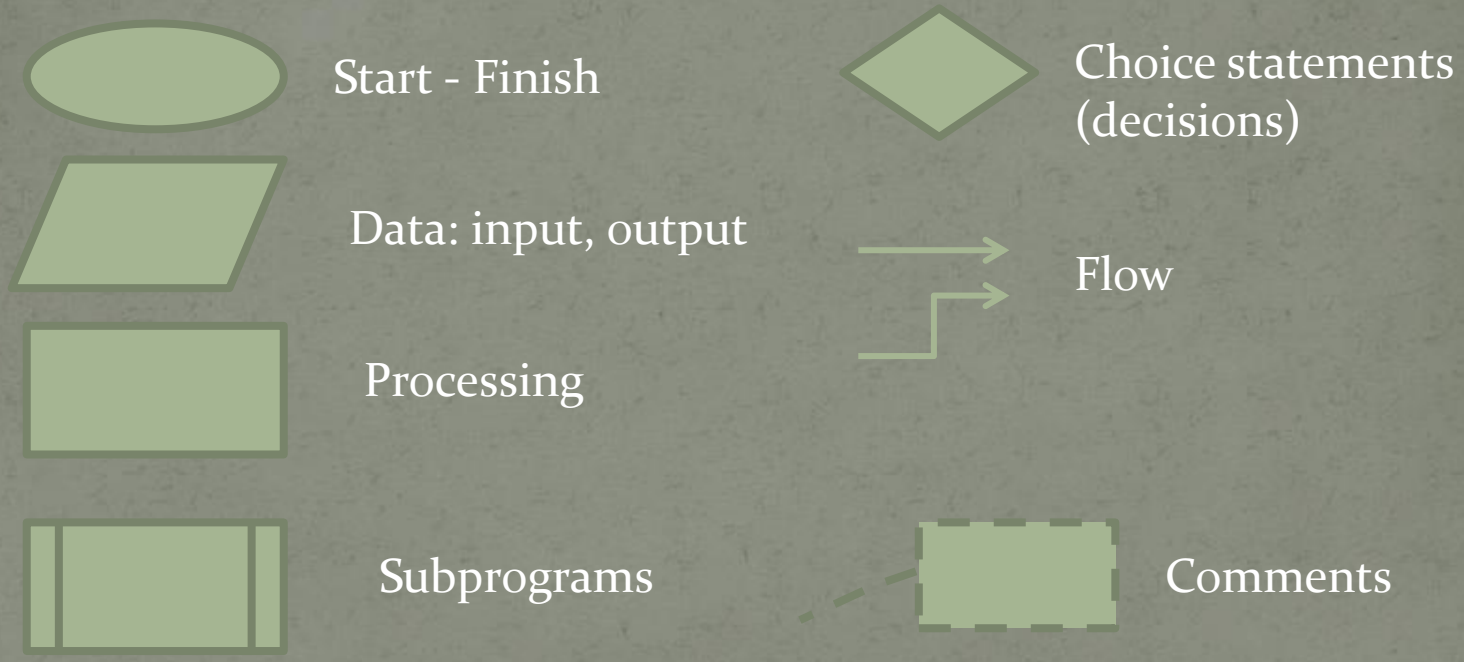
- Why to use a JavaBlock?
  - JavaBlock allow to presenting algorithms in a graphical way. This is the simplest method of programming for beginners, because you do not need to know language syntax, and the developer is not limited by the writing code.

Rapid prototyping - JavaBlock

- Why to use a JavaBlock?
  - JavaBlock allow to presenting algorithms in a graphical way. This is the simplest method of programming for beginners, because you do not need to know language syntax, and the developer is not limited by the writing code.
  - Block diagrams also allow to see what happens when the program is in the design phase.

Rapid prototyping - JavaBlock

- Flowchart – short remind



Rapid prototyping - JavaBlock