Sorting

Presented by :
Hariom Rauniyar
Nikita Silwal
Bipin Poudel

Sorting

The process of rearranging the items in a list according to some linear order is termed as sorting. In other words, Sorting is the process of ordering elements in an array in specific order e.g. ascending or descending order. Sorting can be done based on key references which can be numbers, alphabets, etc. Some examples of sorting are :-

1. Words in a dictionary are sorted.
2. A list of cities could be sorted by population, by area, or by zip code.
3. Index of a book.
4. A listing of course offerings at a university sorted, etc.

Need of sorting

We know that searching a sorted array is much easier than searching an unsorted array. This is especially true for people i.e. finding a person's name in a phone book is easy, but finding a number without knowing the person's name is virtually impossible. As a result, any significant amount of computer output is generally arranged in some sorted order so that it can be interpreted.

Types of Sorting

- Internal Sorting
- External Sorting

Internal Sorting

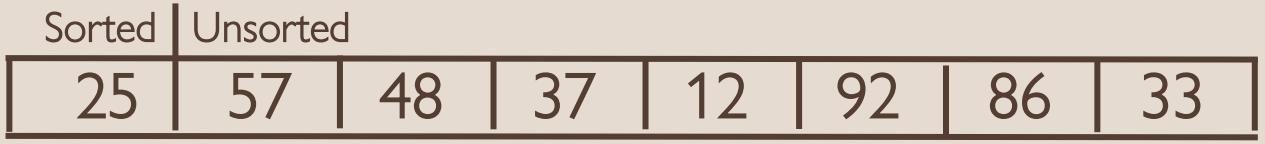
- All the data to sort is stored in main memory at all times while sorting is in progress.
- It is performed when the data to be sorted is small enough to fit in the memory.
- In other words, it is used when size of input is small.
- The storage device used is RAM.
- Insertion sort, Quick sort, Bubble sort etc.

External Sorting

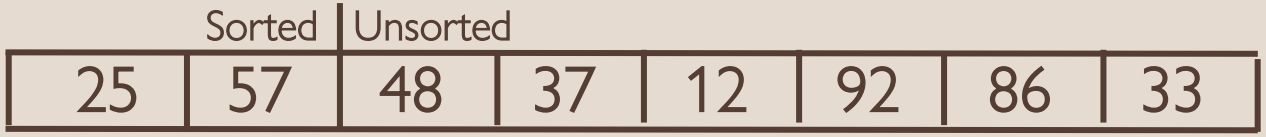
- Data is stored outside memory(like on disk) and only loaded into memory is small chunks.
- External sorting is usually applied in cases when data can't fit into main memory entirely.
- In other words, it is used when size of input is large.
- Storage devices are both secondary memory(hard disk) and main memory(RAM).
- External merge sort, external radix sort, four tape sort, etc.

❑ Insertion Sort (25,57,48,37,12,92,86,33)

Pass 1 :



Pass 2 :



Pass 3 :

Pass 4 :

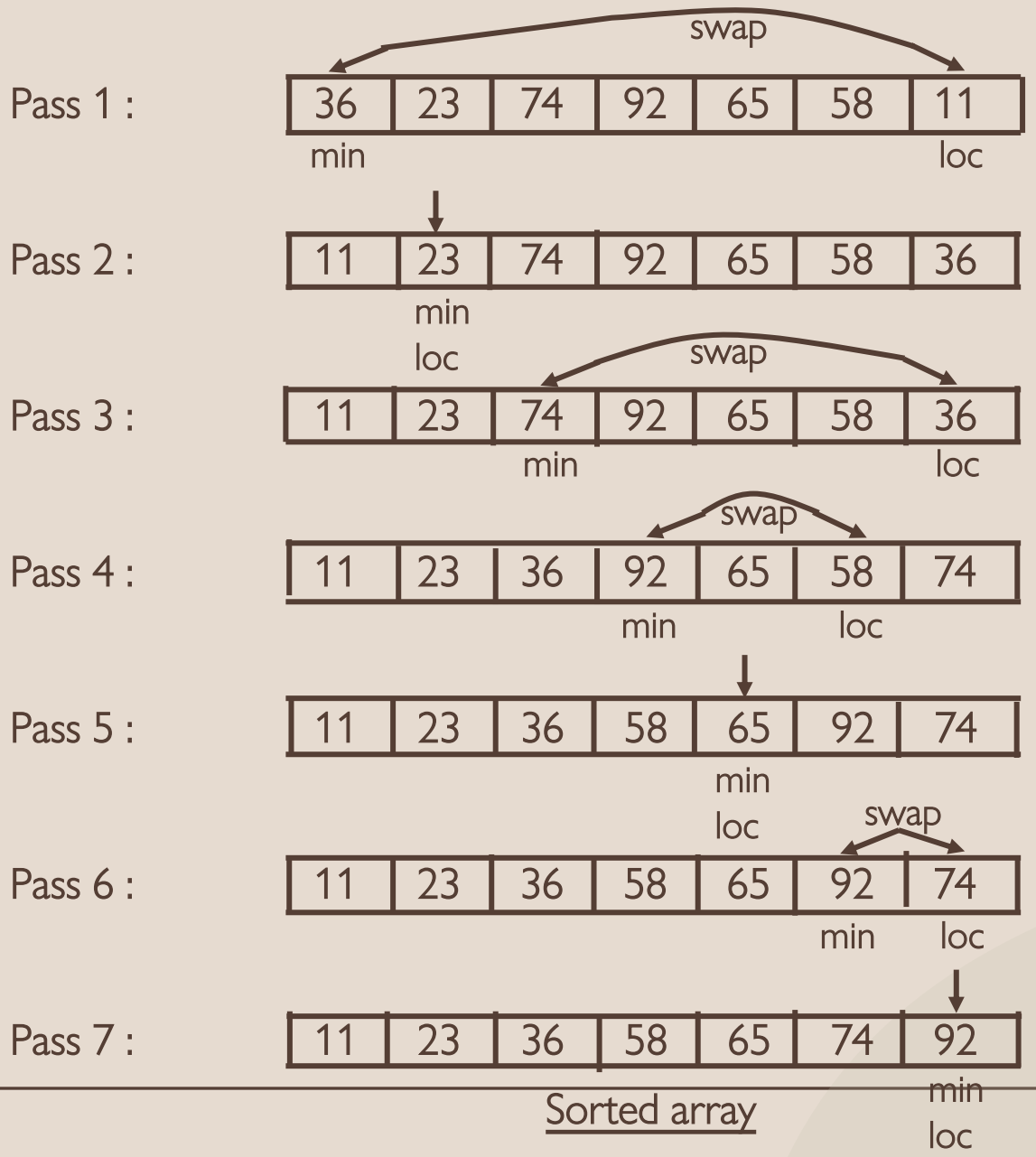
Pass 5 :

Pass 6 :

Pass 7 :

Pass 8 :

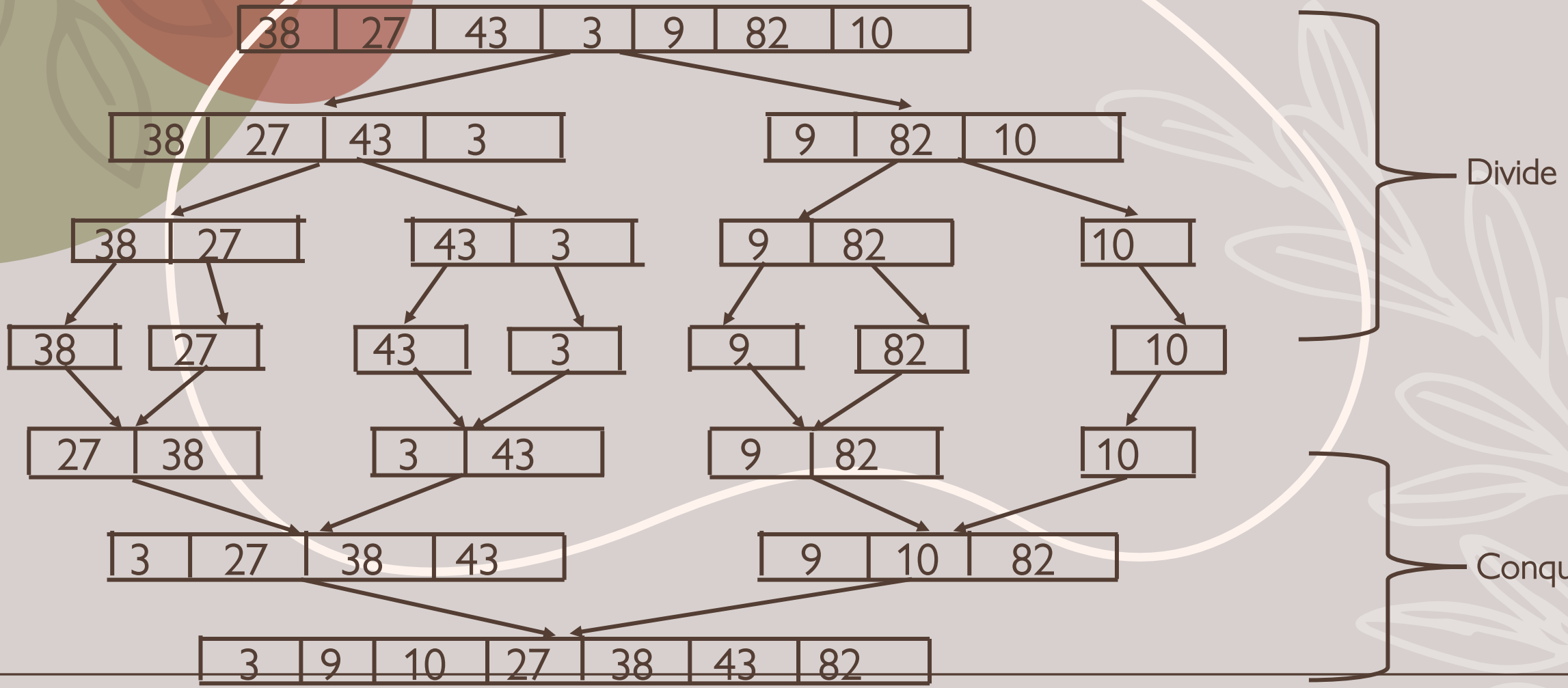
❑Selection Sort (36,23,74,92,65,58,11)



here,
loc = location(smallest value)

❑ Merge Sort (based on divide and conquer strategy)

Sort :- 38,27,43,3,9,82,10



❑Shell Sort (33,31,40,8,12,17,25,42)

N = 8

gap = [N/2] = [8/2] = 4 (floor value)

Pass 1 :

12,17,25,8,33,31,40,42

Pass 2 : gap = [4/2] = 2 (floor value)

12,17,25,8,33,31,40,42

12,17,25,8,33,31,40,42

12,8,25,17,33,31,40,42

12,8,25,17,33,31,40,42

12,8,25,17,33,31,40,42

12,8,25,14,33,31,40,42

Pass 3 : gap = [2/2] = 1 (floor value)

12,8,25,17,33,31,40,42

8,12,25,17,33,31,40,42

8,12,25,17,33,31,40,42

8,12,17,25,33,31,40,42

8,12,17,25,33,31,40,42

8,12,17,25,31,33,40,42

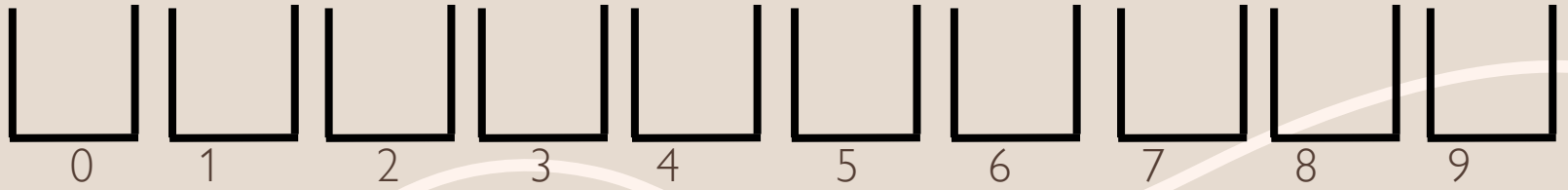
8,12,17,25,31,33,40,42

8,12,17,25,31,33,40,42 (Sorted array)

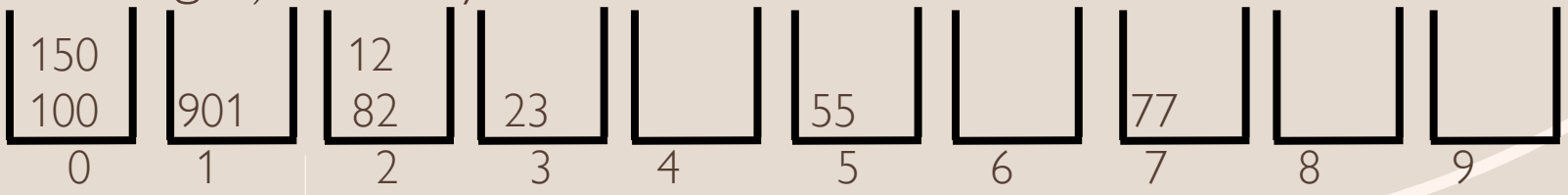
❑Radix Sort

(82,901,100,12,150,77,55,23)

Step 1 : Define 10 queues each representing a bucket for digits from 0 to 9.



Step 2 : Insert all numbers of the list into respective queue based on the least significant digits (one place digits) of every number.

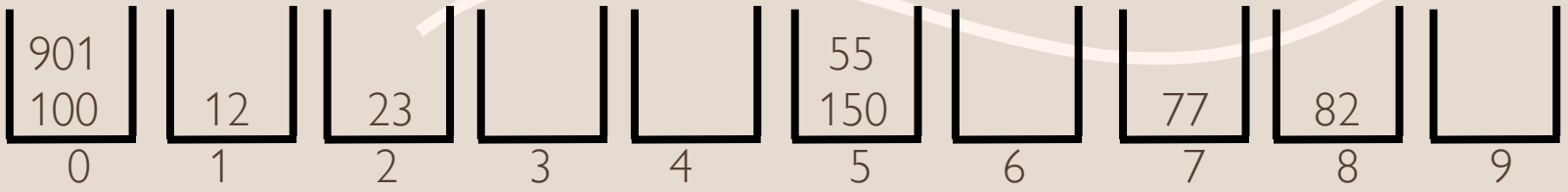


Group all the numbers from queue 0 to queue 9 in the order they have inserted.

(100,150,901,82,12,23,55,77)

Step 3 : Insert all the numbers of the list into respective queue based on the next least significant digits(Tens placed digits) of every number.

(100,150,901,82,12,23,55,77)

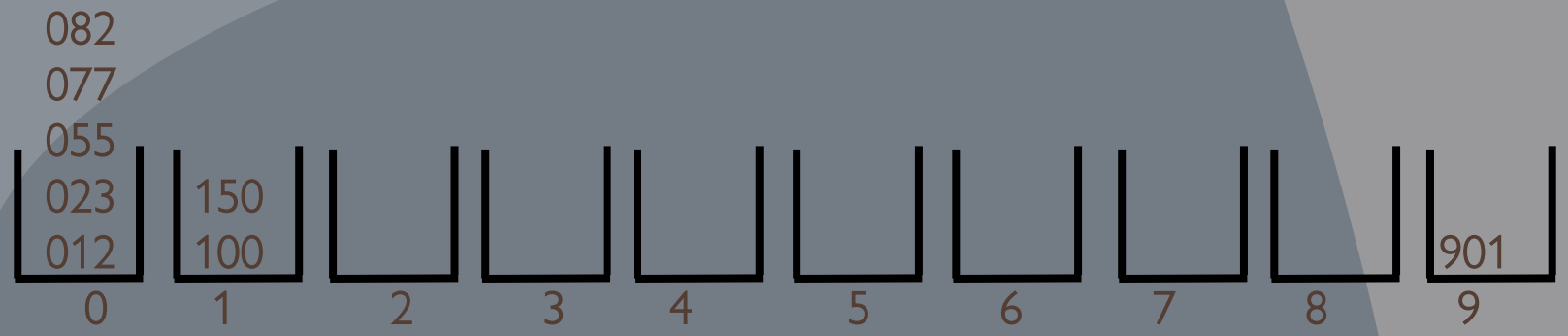


Group the numbers from queue-0 to queue-9 into the order they have inserted and consider the list for next step.

(100,901,12,23,150,55,77,82)

Step 4 : Insert all the numbers of the list into respective queue based on the next significant digits (Hundreds placed digit) of every number.

(100,901,12,23,150,55,77,82)



Group all the numbers from queue 0 to queue 9 in the order they have inserted & consider the list for next step as input.

(12,23,55,77,82,100,150,901)

List is sorted.

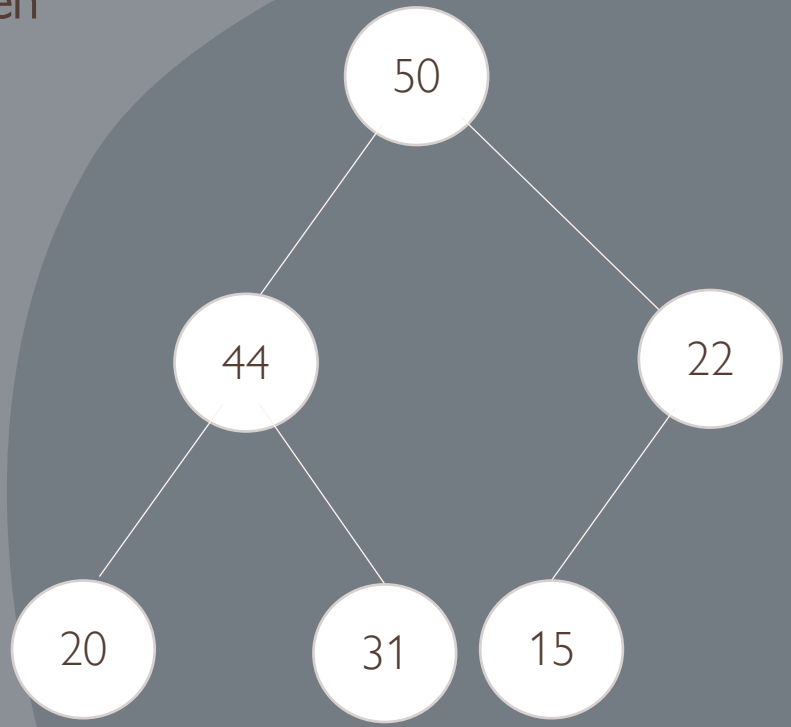
❑Binary heap(heap)

Heap is a tree structure (almost complete binary tree) data structure in which all the tree nodes are in particular order, such that tree satisfies the heap properties (i.e. a specific parent-child relationship)

❑Types of heap

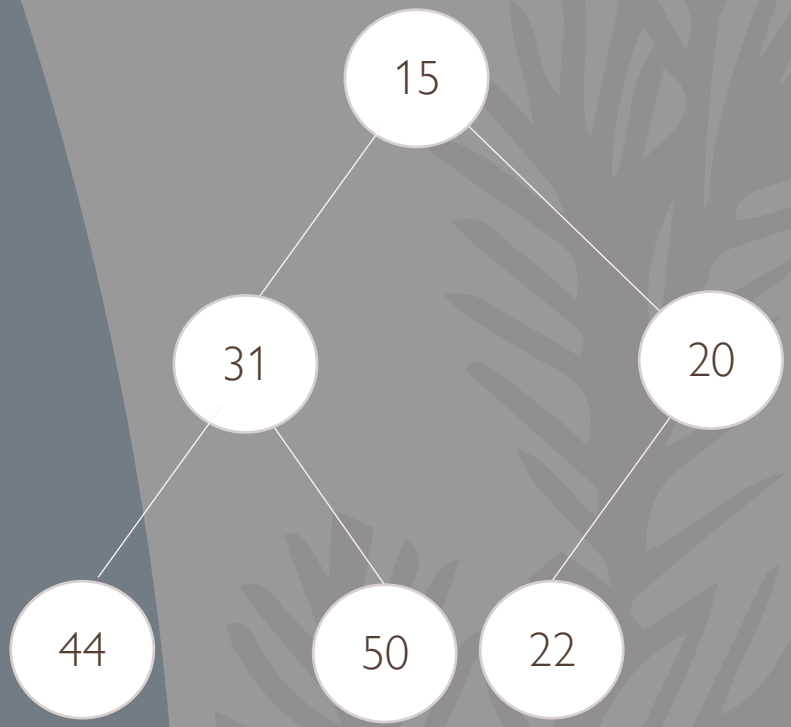
Max heap :

Value of node is always >= the value of each of its children

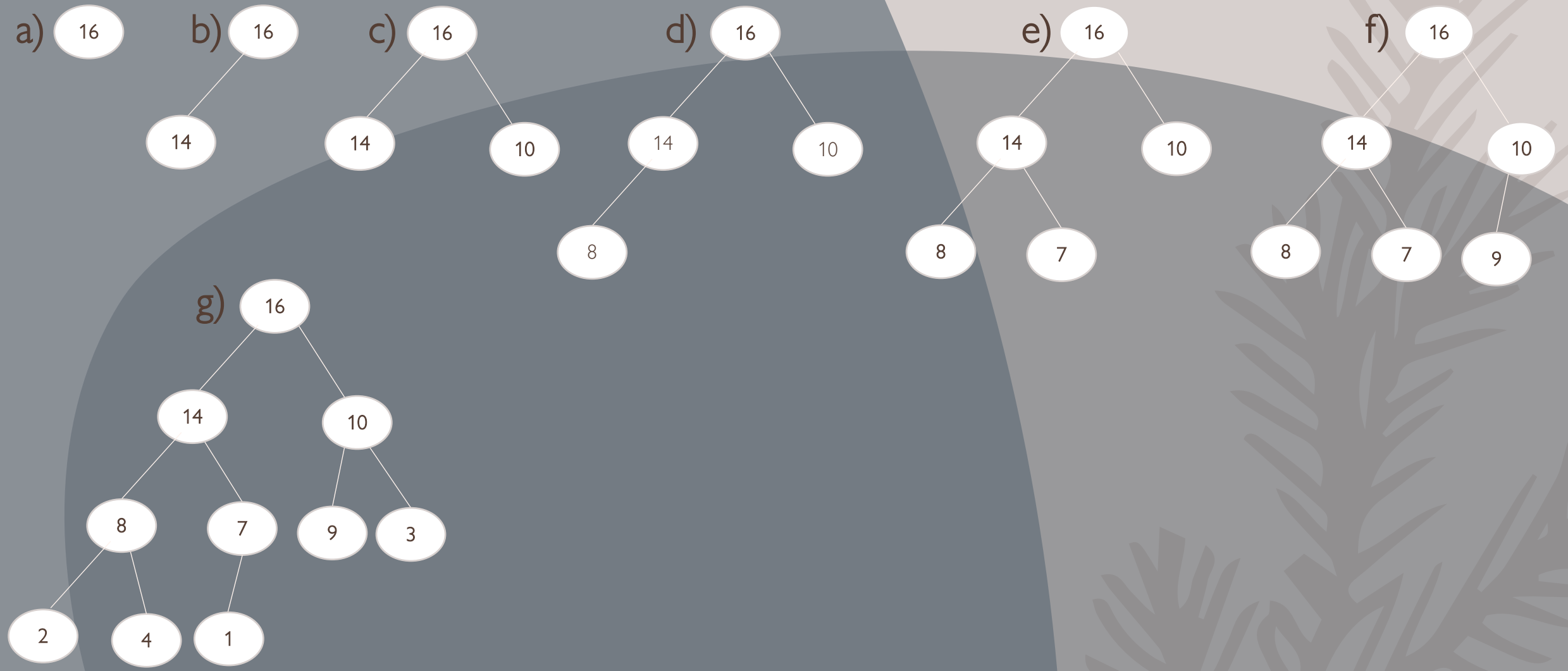


Min heap :

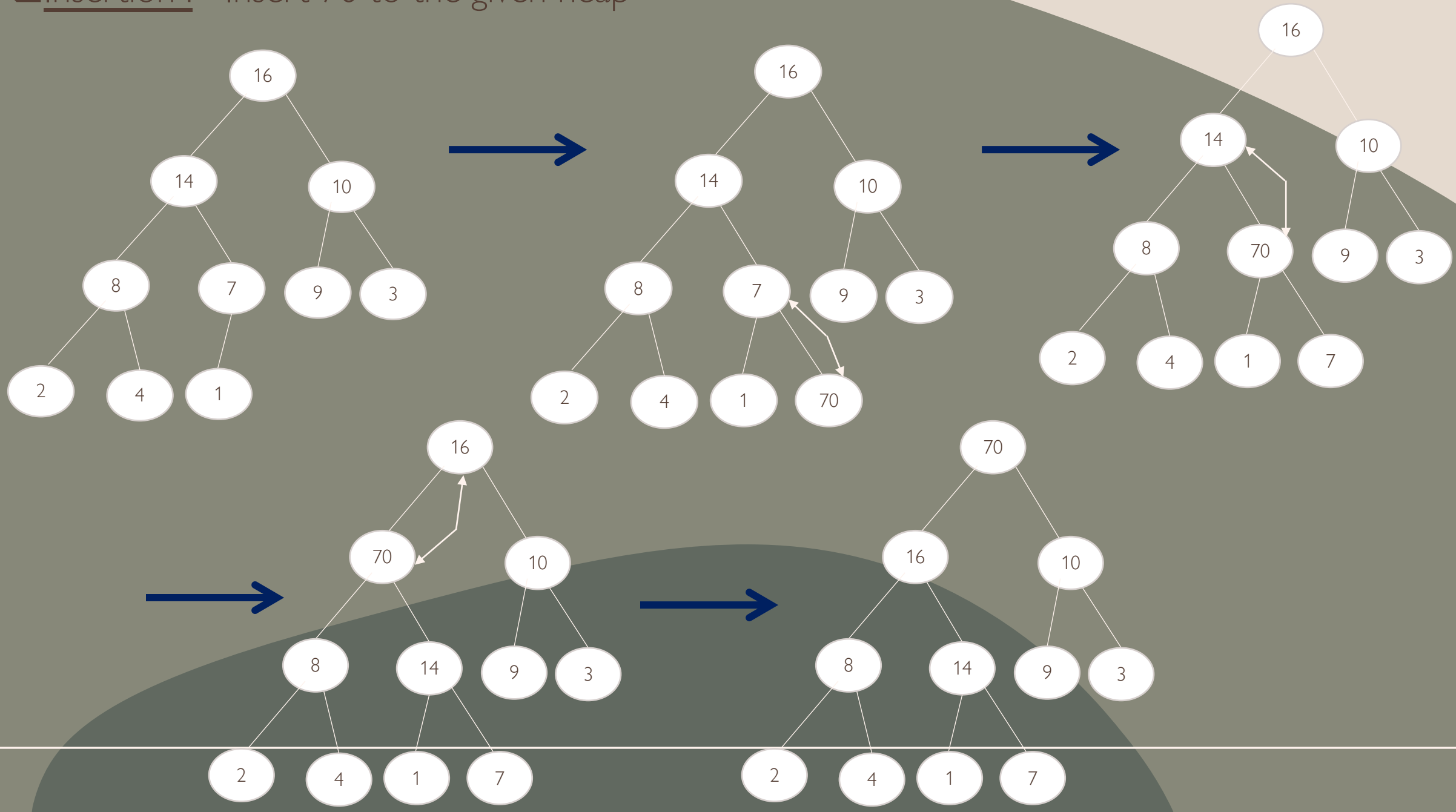
Value of node is always <= the value of each of its children..



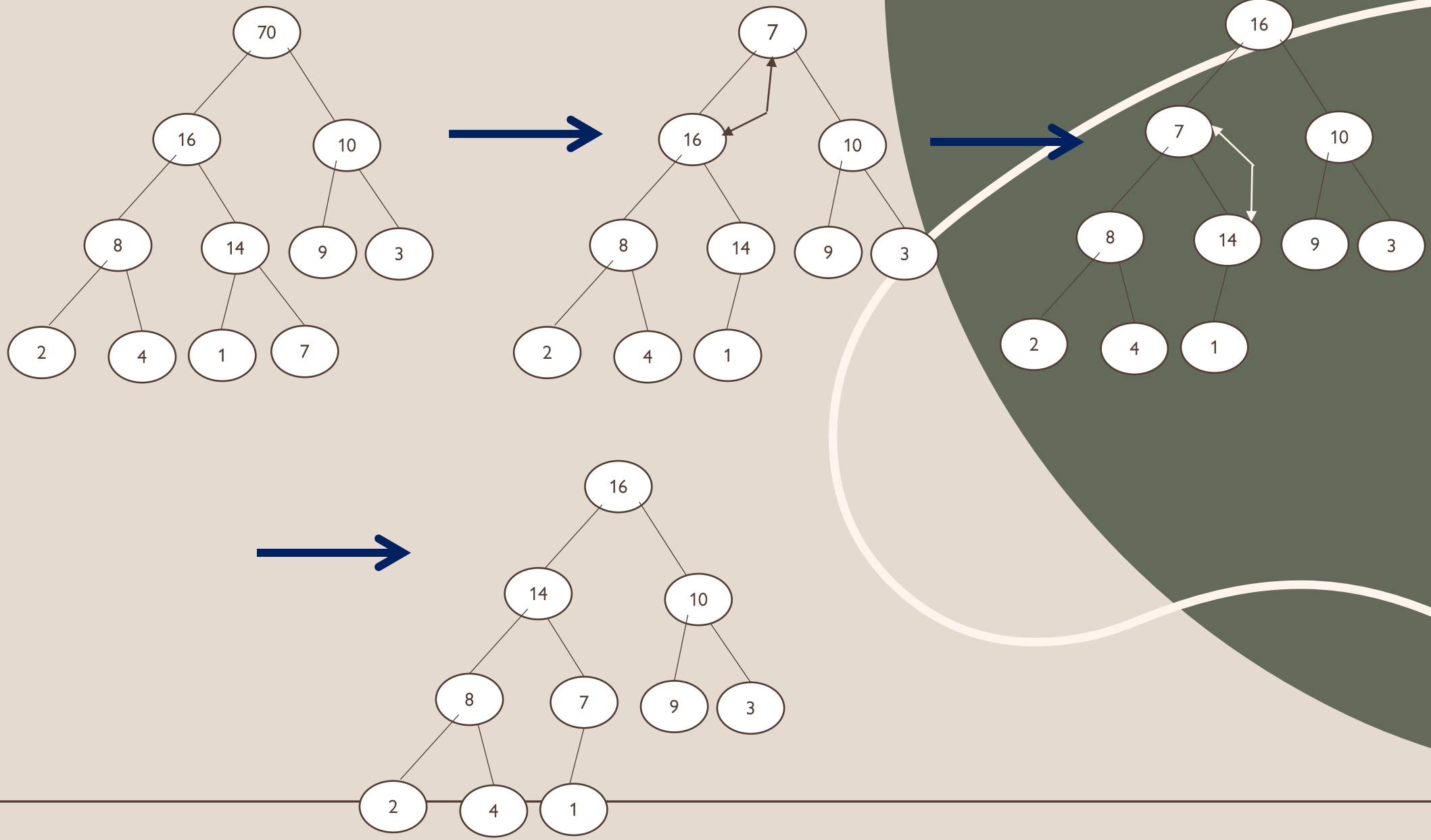
- Make a Max heap: (16,14,10,8,7,9,3,2,4,1)



❑Insertion : Insert 70 to the given heap



❑Deleting : Delete 70 from root



Thank you!