Data Management and File Organization

External Sorting

Why Sorting?

- Record access in sorted files is much faster than pile files
- In large files, exhaustive read in the order of an attribute may take several days but in a sorted file it only needs a few seconds

Internal and External Sorting

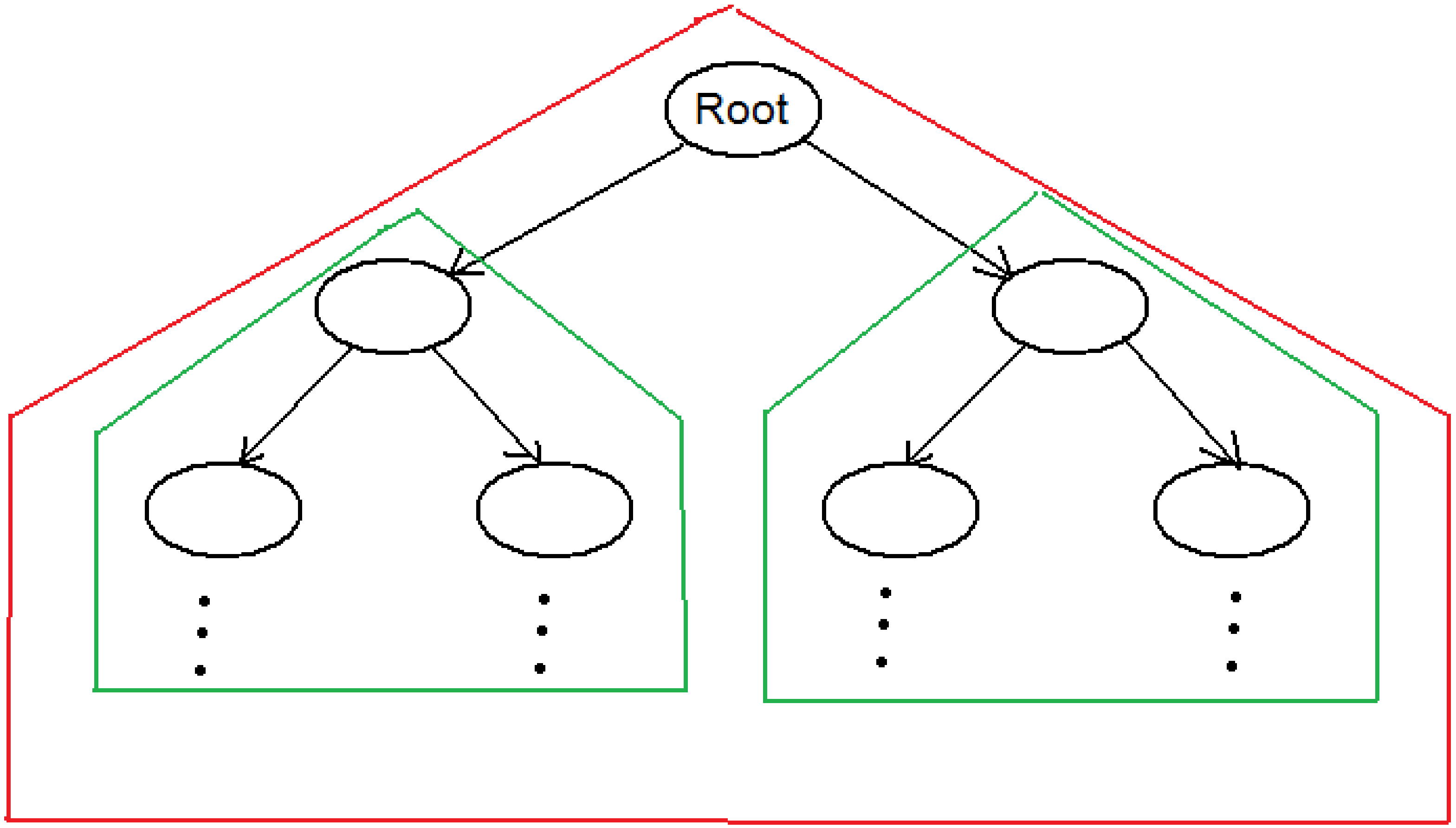
- Internal sorting can only sort data in memory
  - Example: Quick sort, Merge sort

- External sorting is used when the data is larger than the memory

Tree Data Structure

- A tree is a 2D data structure
- Tree has a node called root and some nodes called children
- Each child is the root of a sub-tree

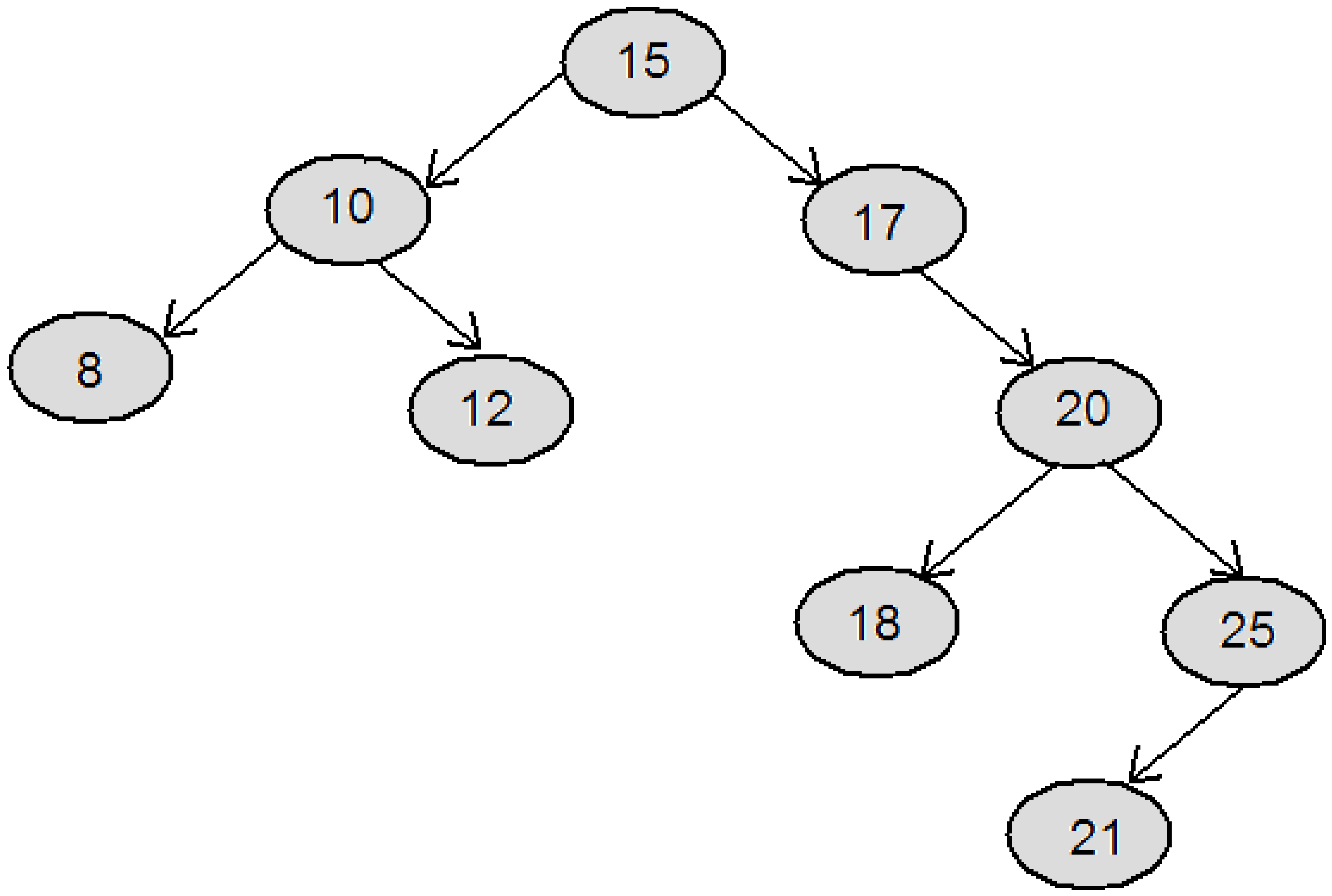
Tree Data Structure



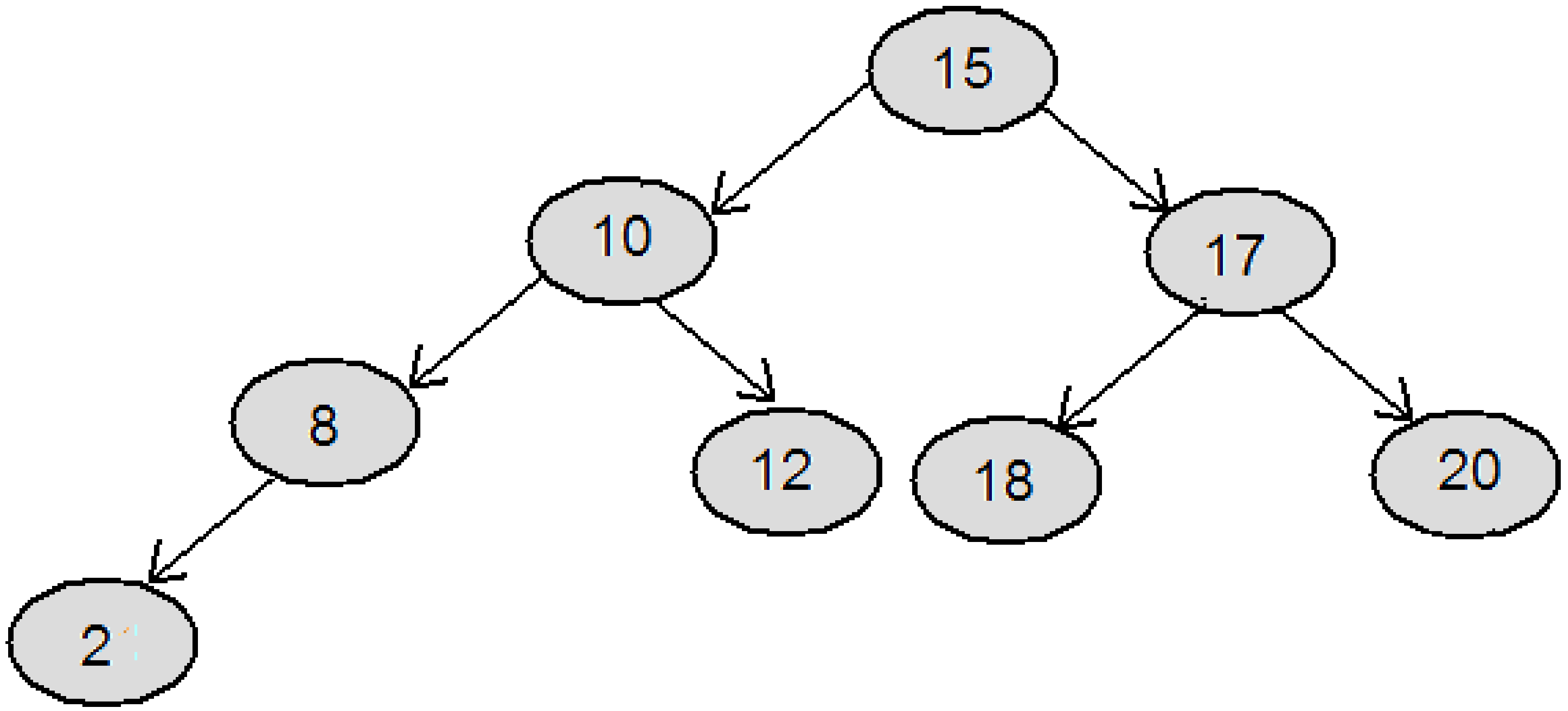
Binary Trees

- A Tree with at most two children at each node is called a binary tree
- A complete binary tree is
  - A *binary tree* in which every level, except possibly the deepest, is completely filled.
  - At the deepest level, all *nodes* must be as far left as possible

Example: Binary tree



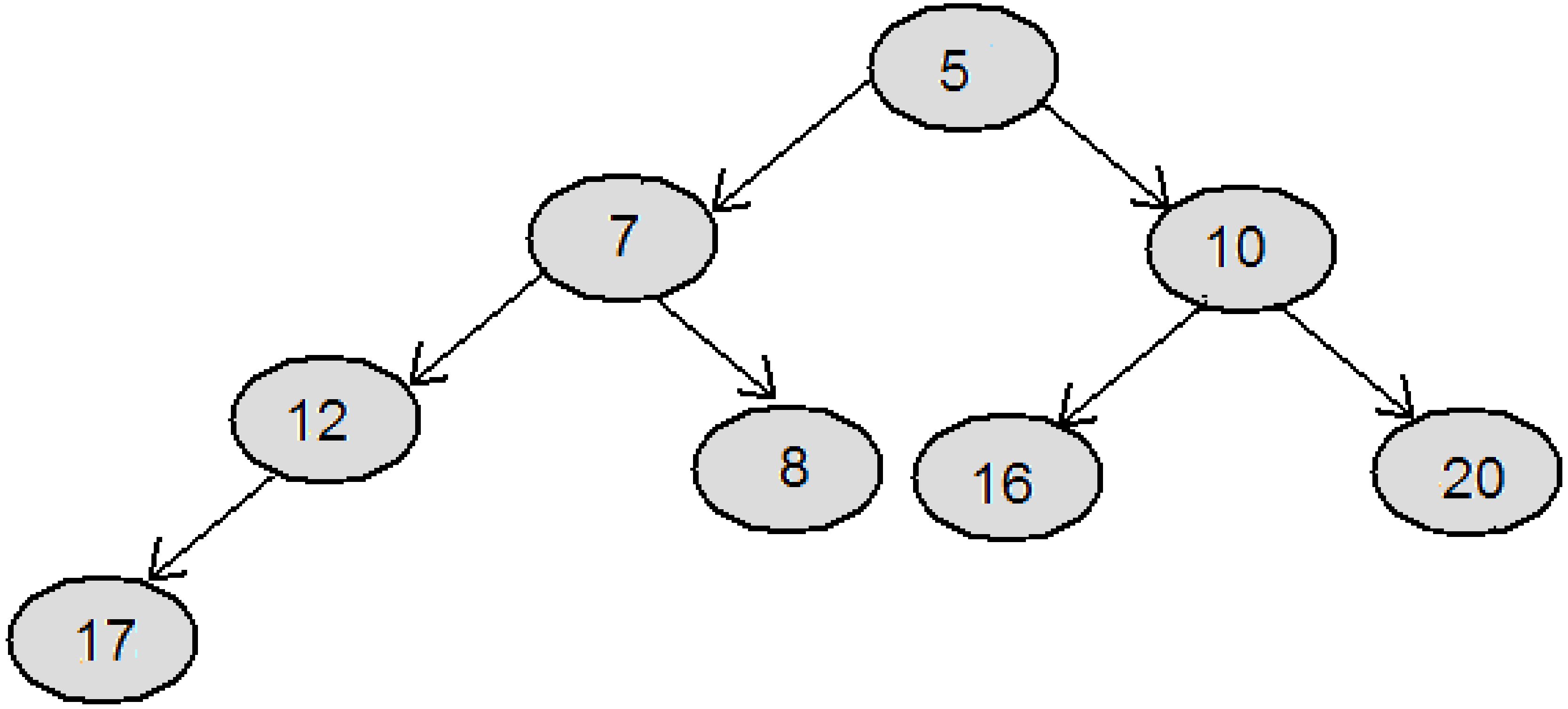
Example: Complete Binary Tree



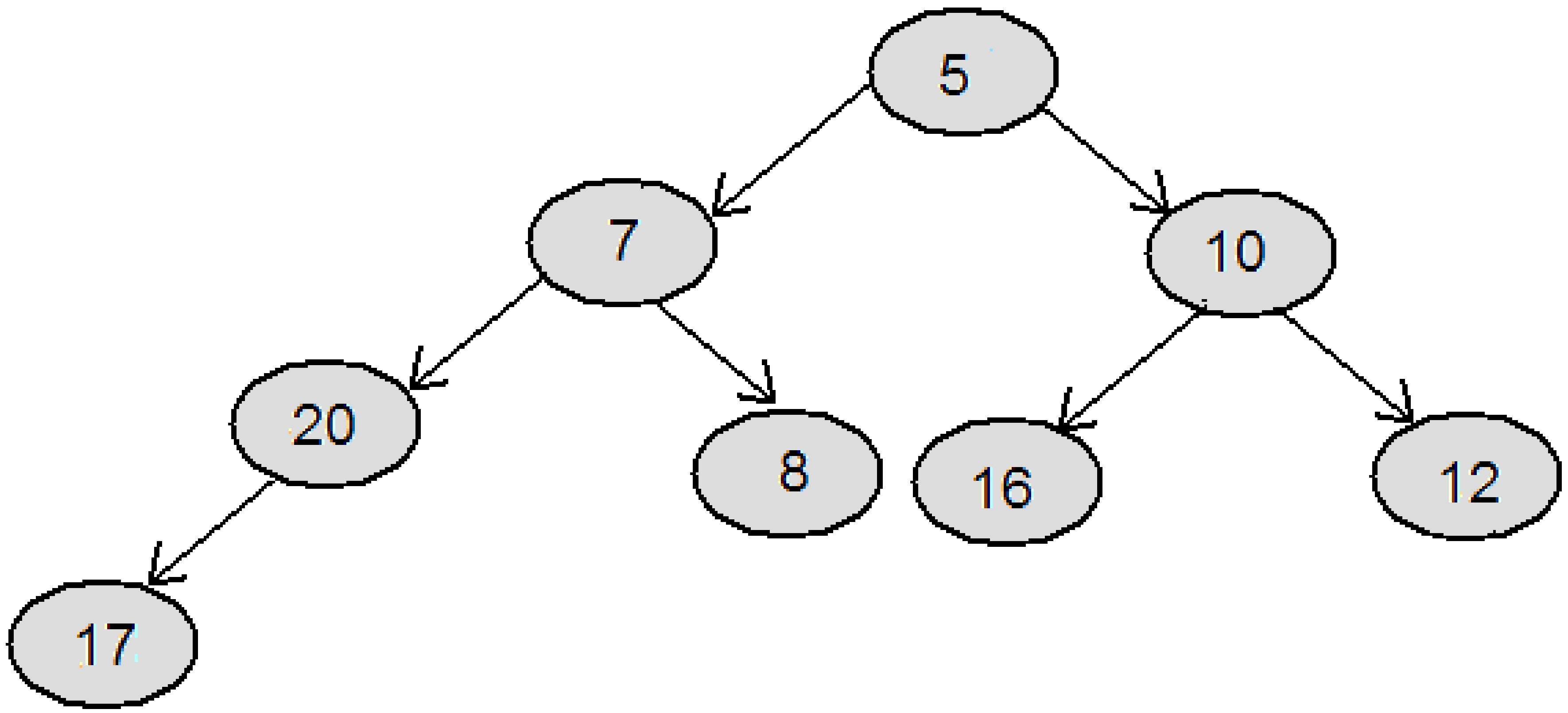
Heap Tree

- A complete binary tree where the value of each node is smaller than its children

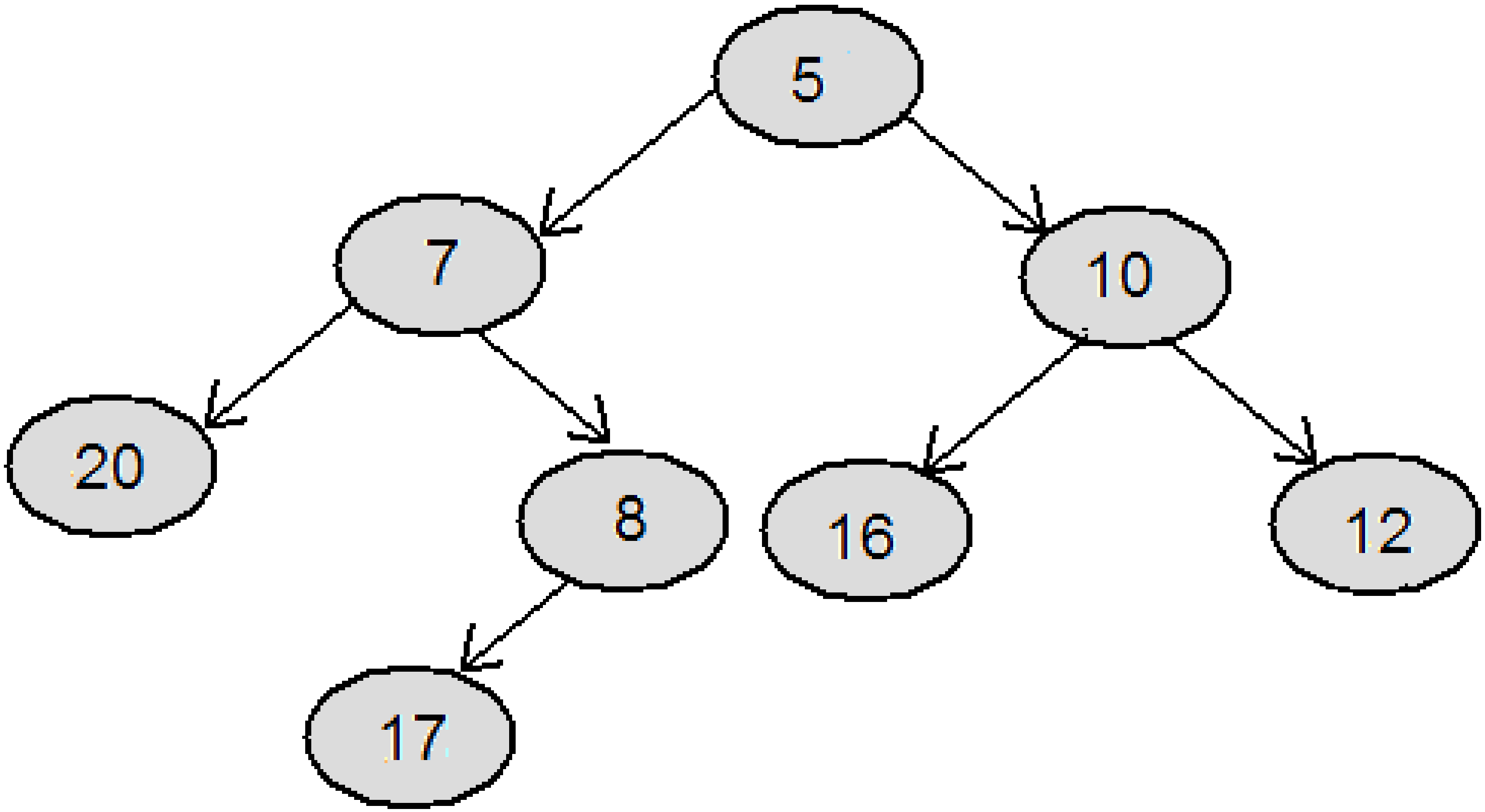
Example: Heap Tree



Example: Not a Heap Tree



Example: Not a Heap Tree



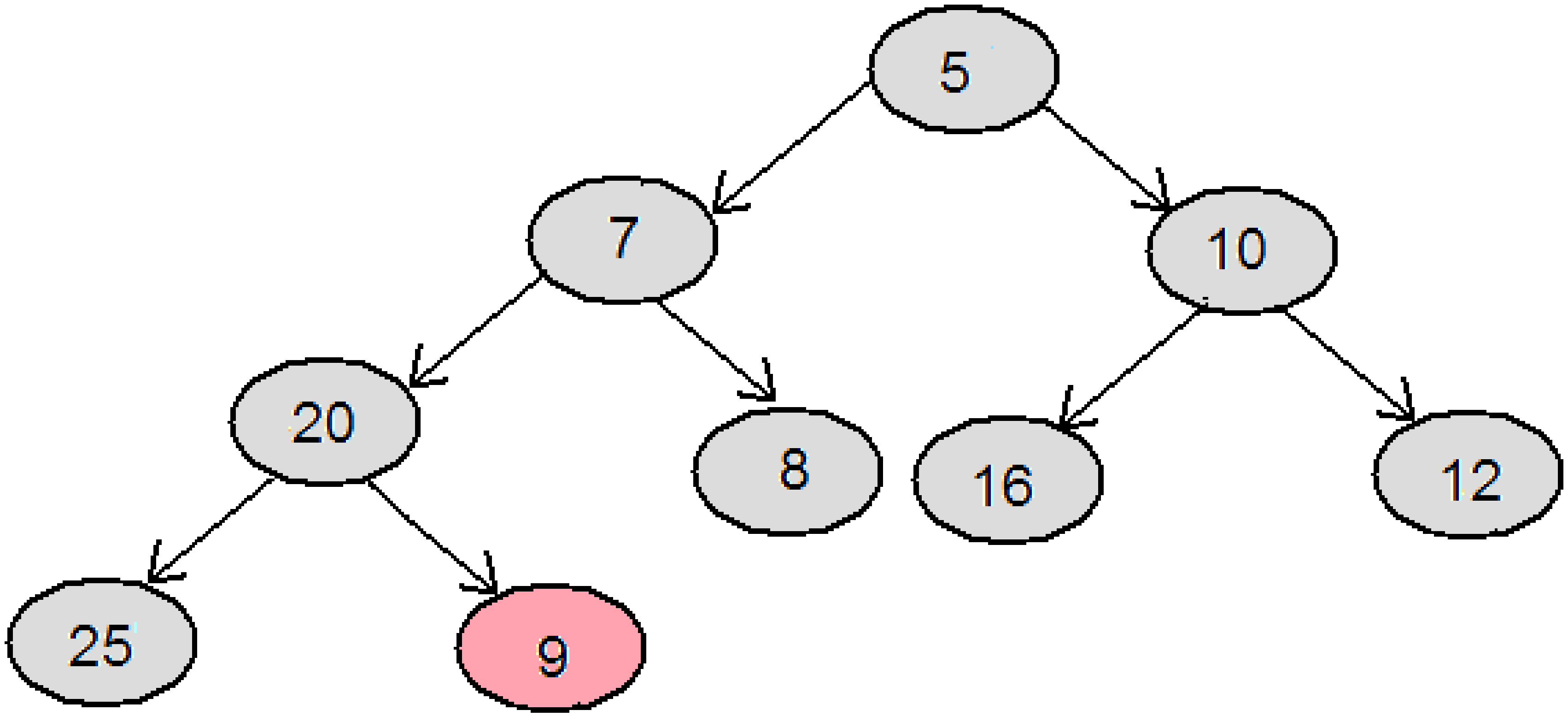
Operations in a Heap Tree

- Two main operations in a heap tree are
  - Insertion
  - Deletion (Always done from the root)

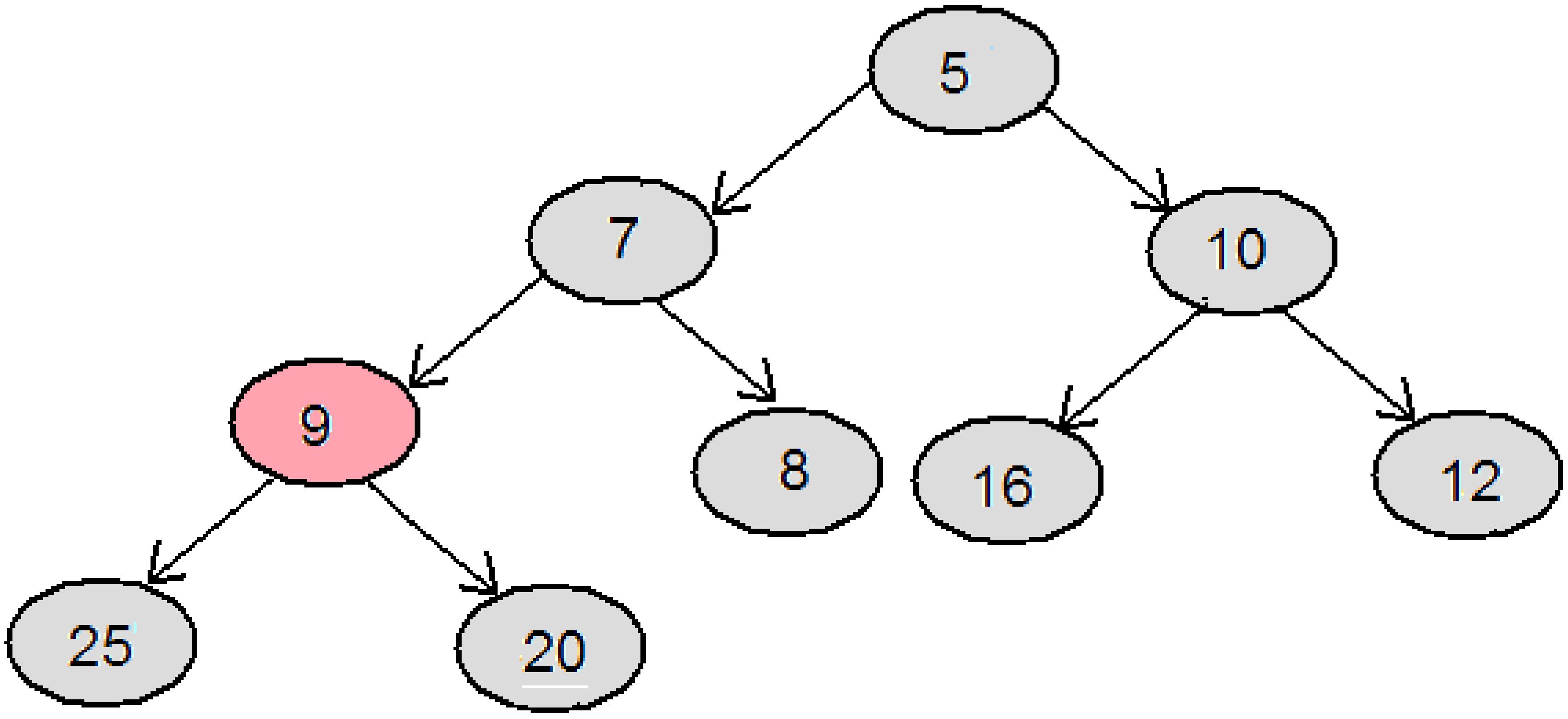
Insertion into a Heap Tree

- Algorithm
  - Add the new node to the last level of the tree preserving complete binary tree
  - Repeat
    - Compare the new node with its parent
    - If less than parent then swap new node and parent
  - Until new node is greater than its parent OR new node is the root

Heap Tree Insertion Example



Heap Tree Insertion Example

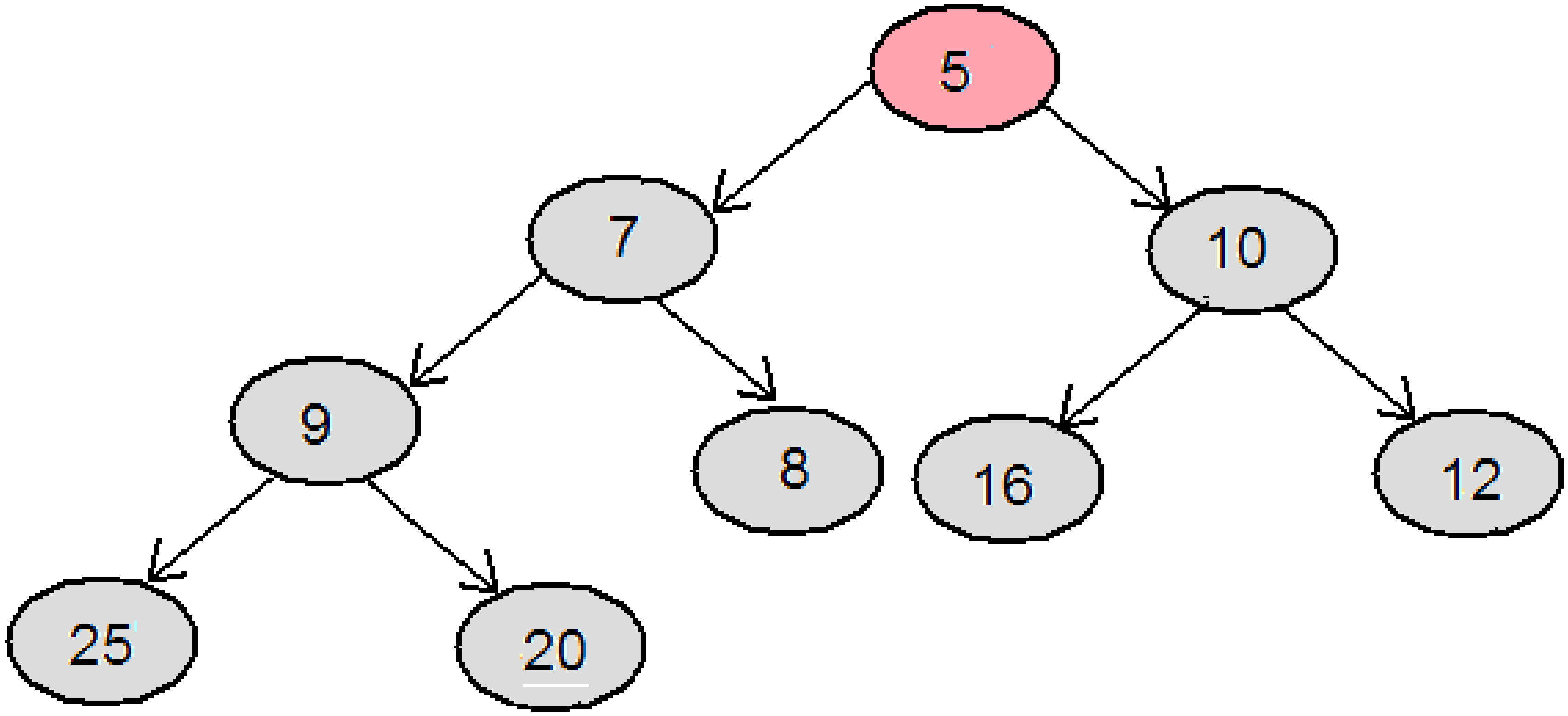


Deleting Root of a Heap Tree

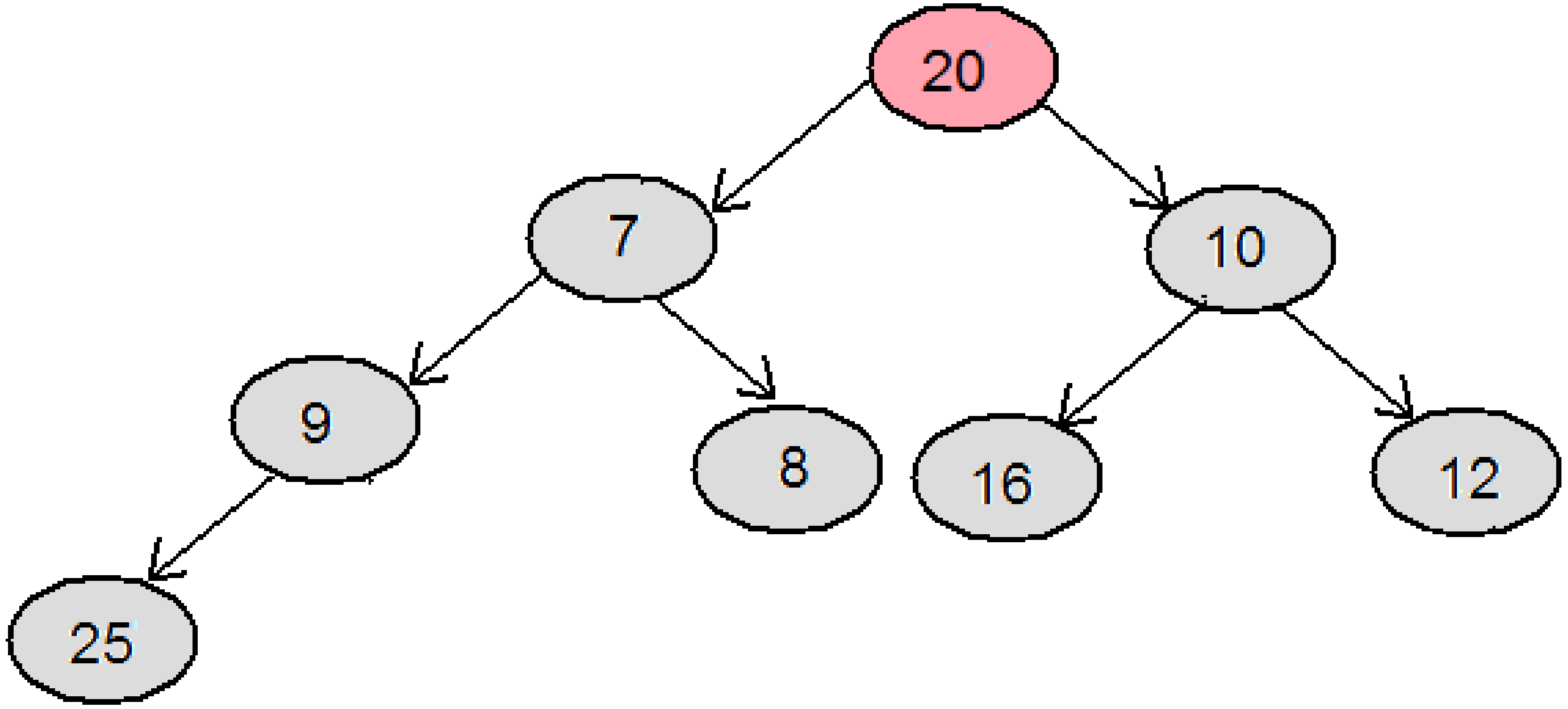
- Algorithm
  - Remove root
  - Move the last node to the place of the root
  - Repeat
    - If the moved node is greater than its child then
      - Swap the moved node with its smaller child
  - Until the moved node is less than its children OR

    the moved node is a leaf node

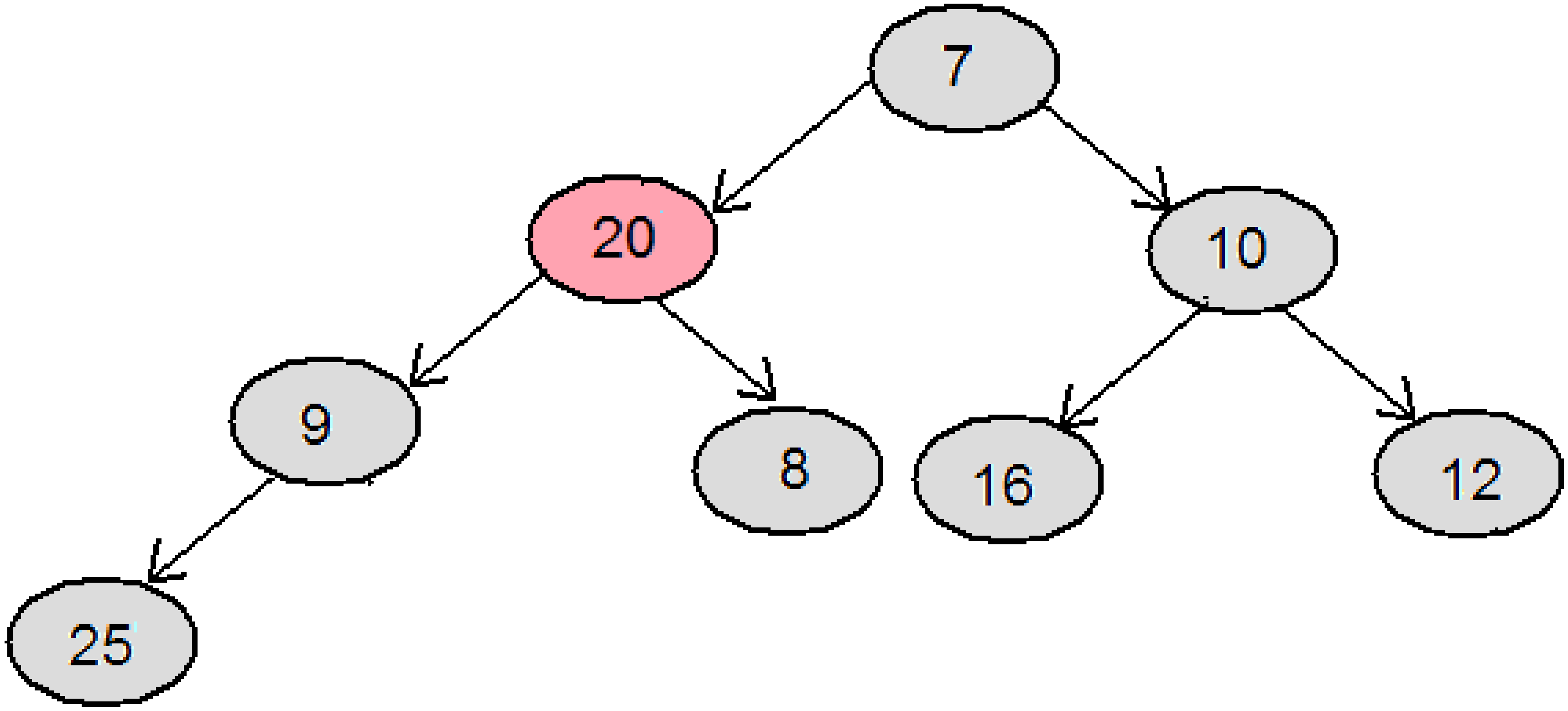
Delete Root of a Heap Tree



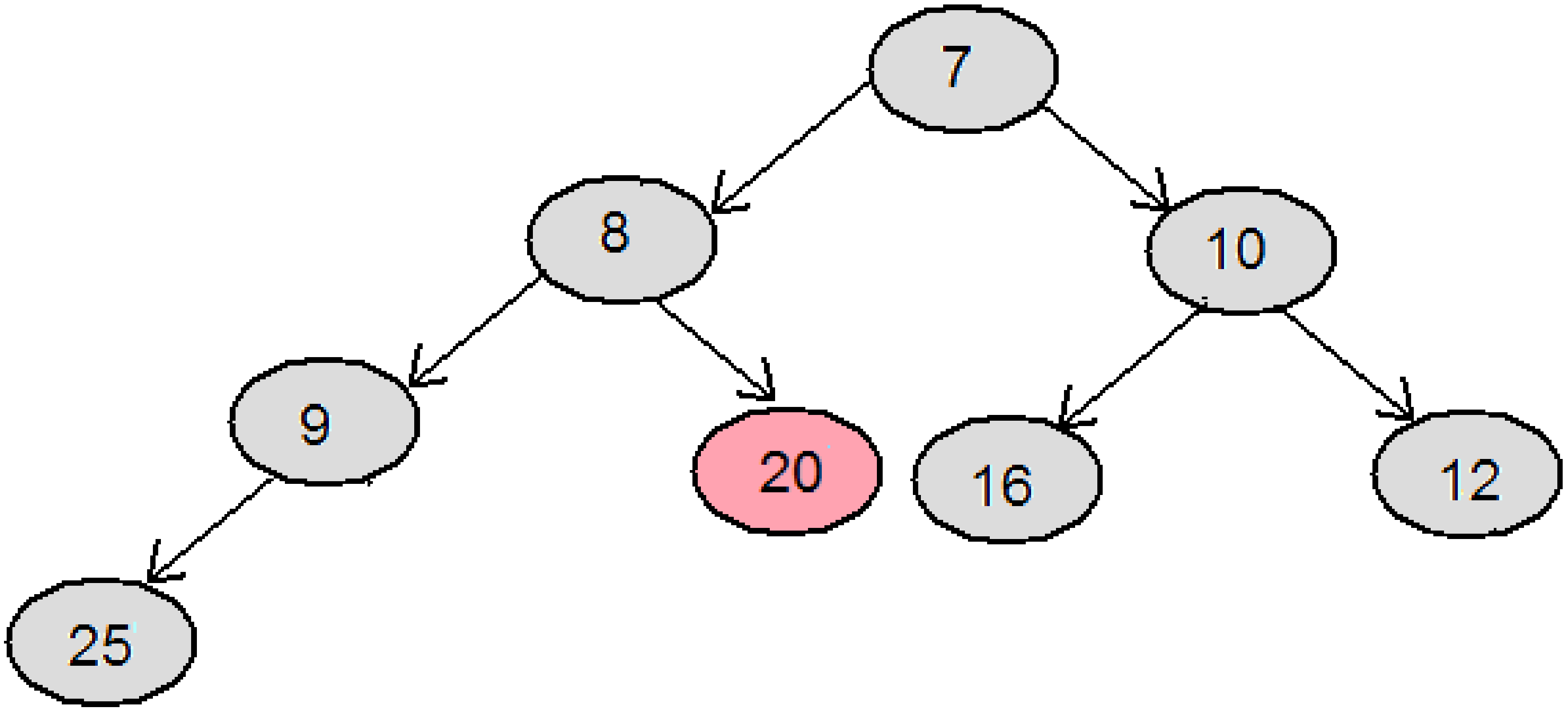
Delete Root of a Heap Tree



Delete Root of a Heap Tree



Delete Root of a Heap Tree



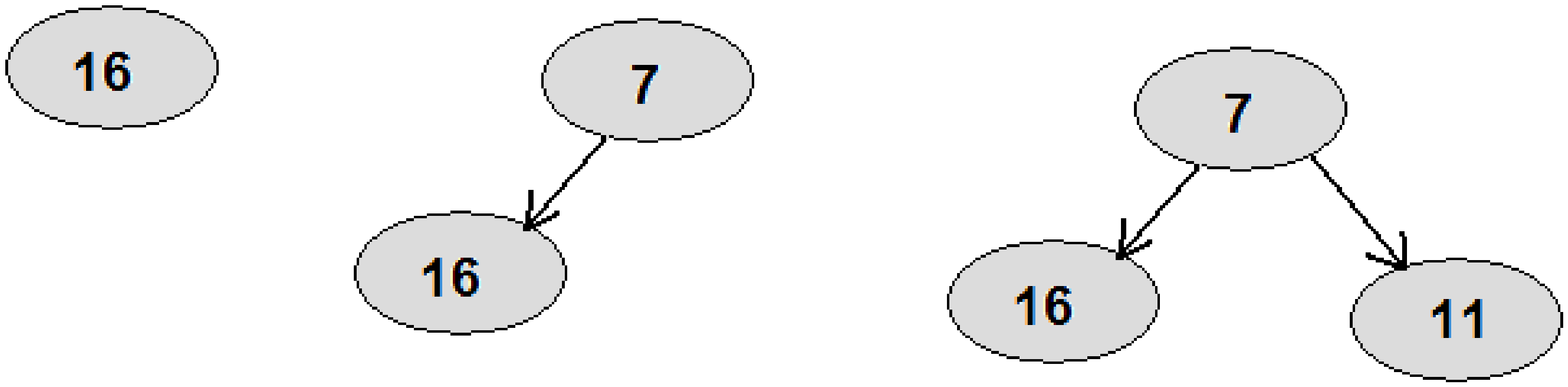
Heap Sort

- Heap sort inserts the data in a heap tree, then deletes from the root and prints until the tree is empty

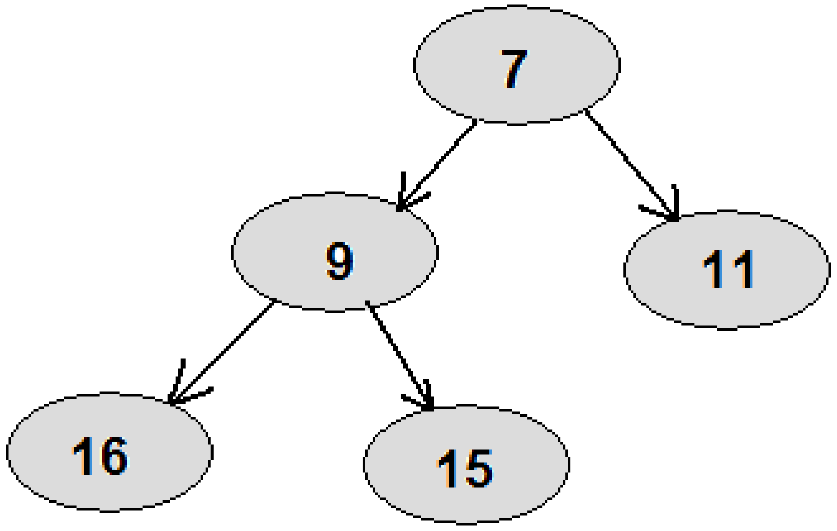
- Algorithm
  - For all data items
    - Insert data items into the heap tree
  - While the heap tree is not empty
    - Remove root and print it

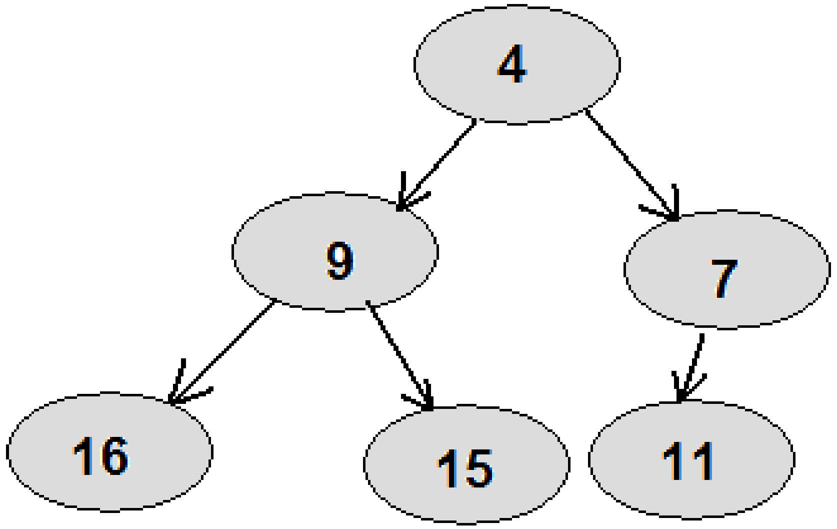
Example: Heap Sort

- Data : 16, 7, 11, 15, 9, 4, 2



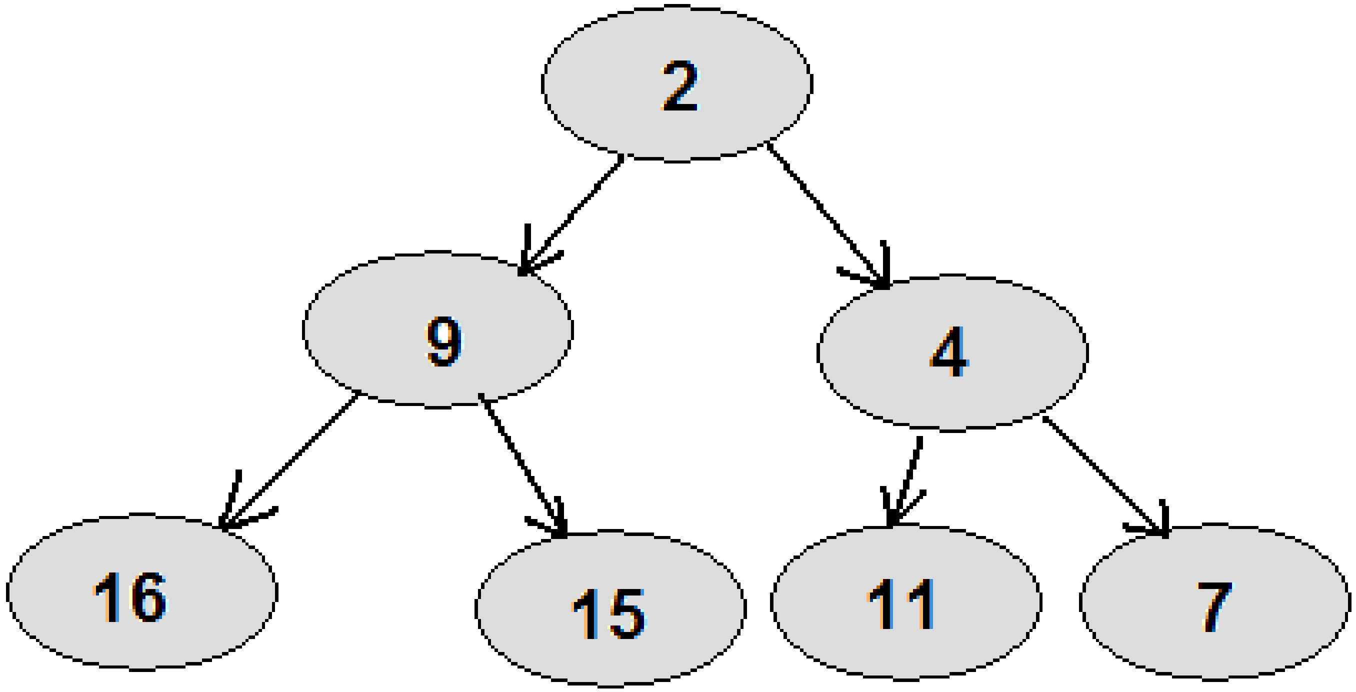
Example: Heap Sort





Example: Heap Sort

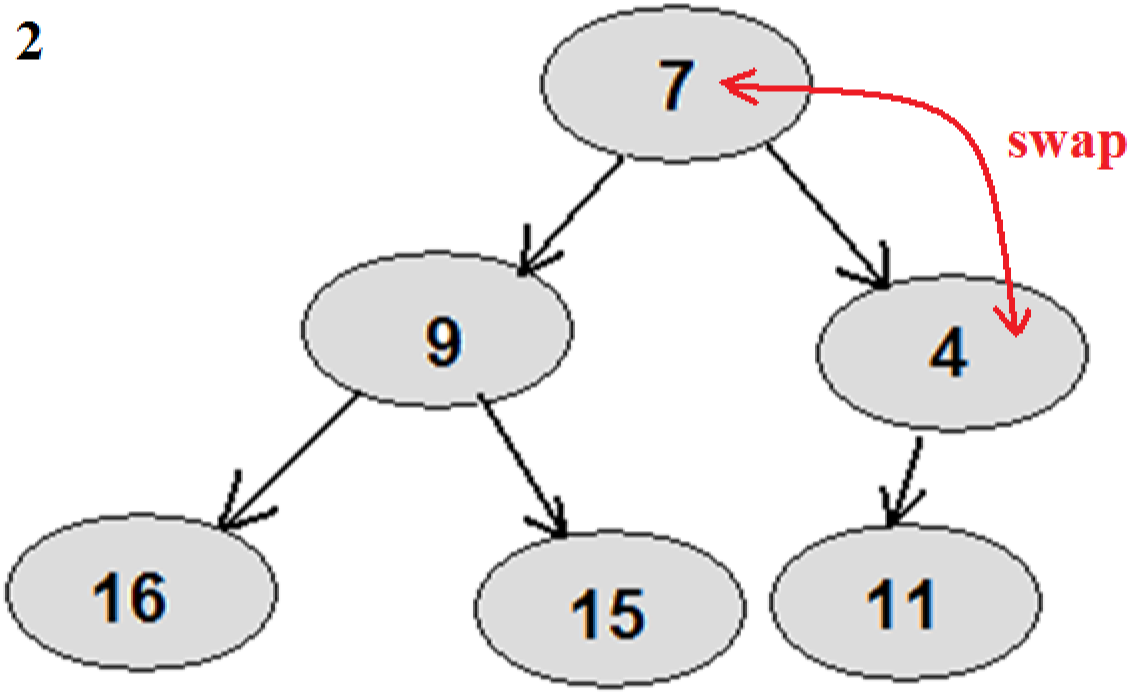
The root always has the smallest item



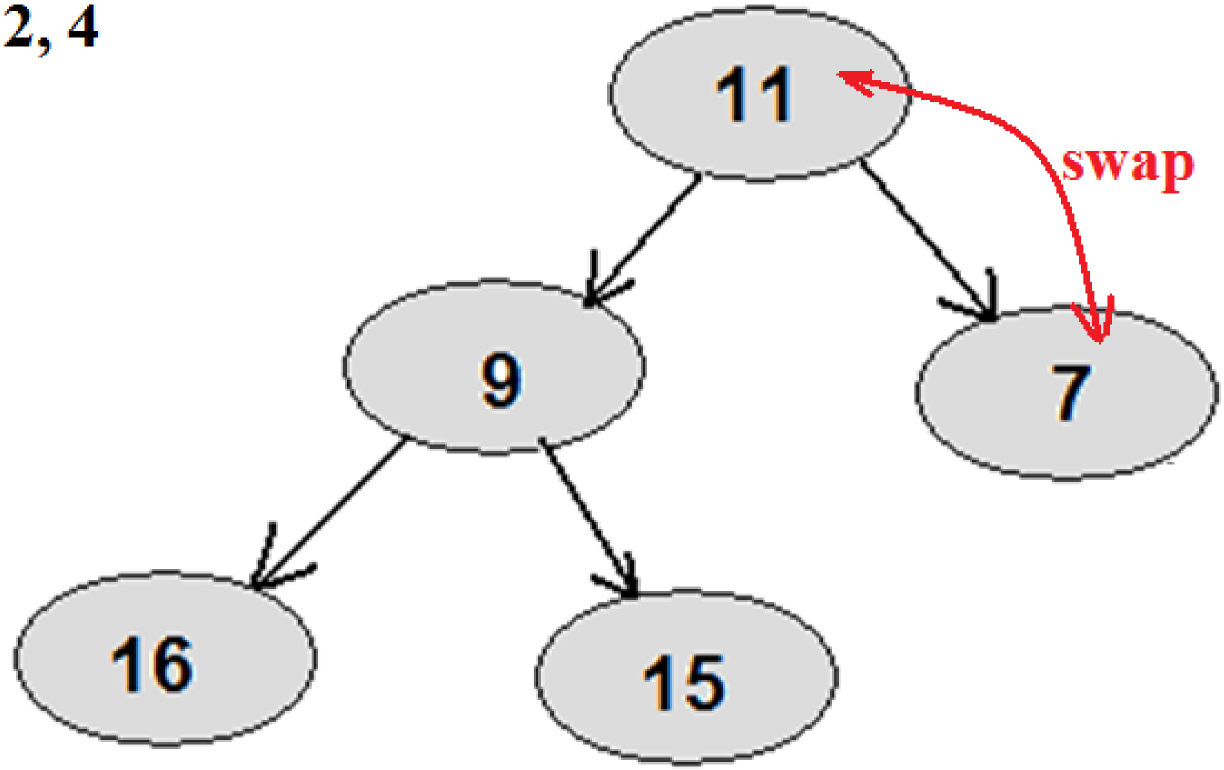
Deleting Loop

- Delete root and print it until the heap tree its empty

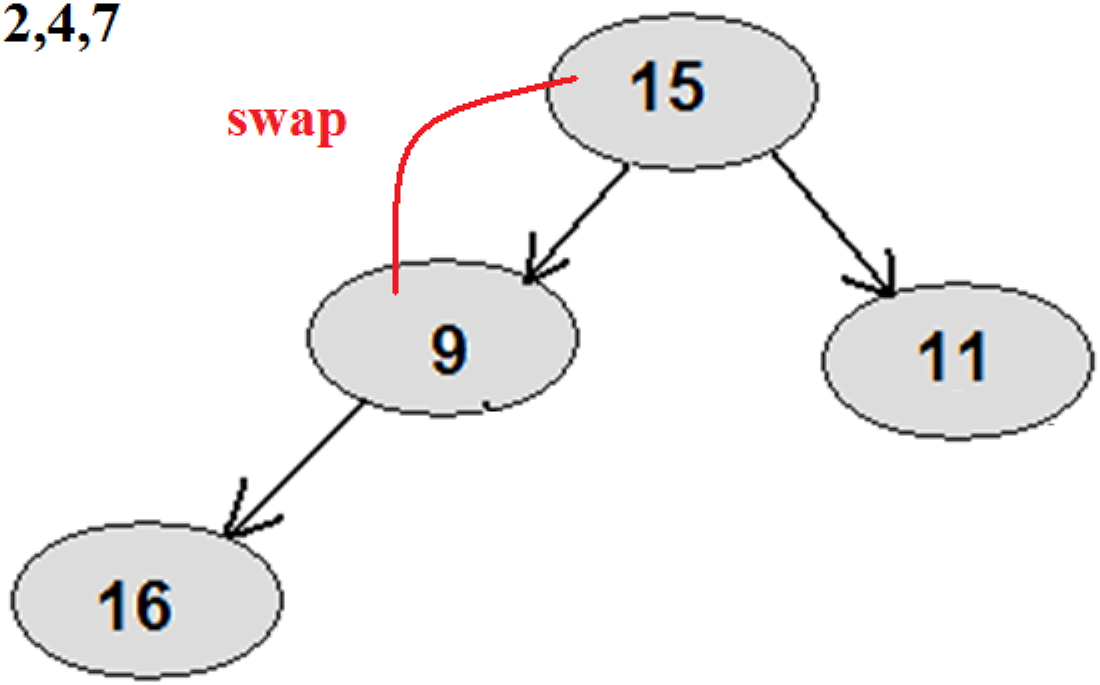
2



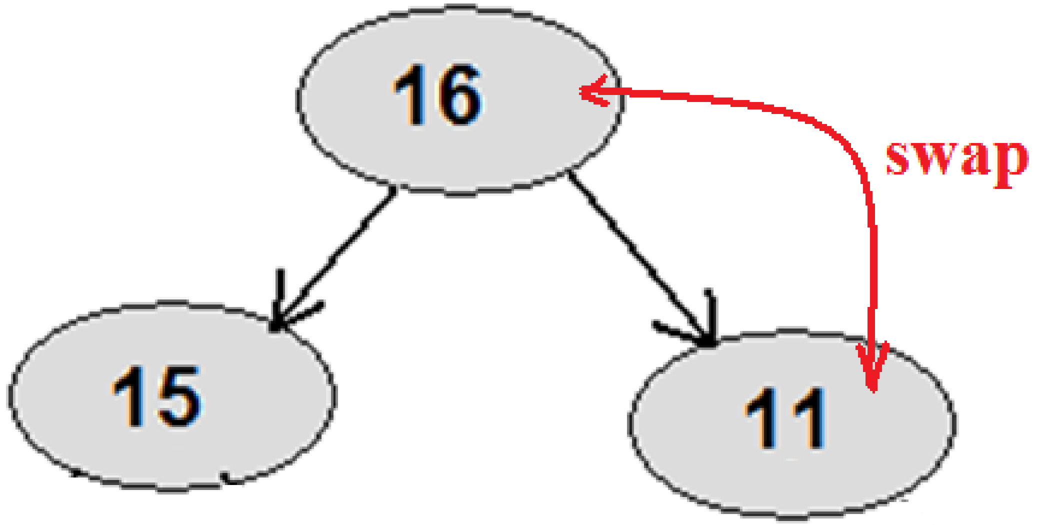
2, 4



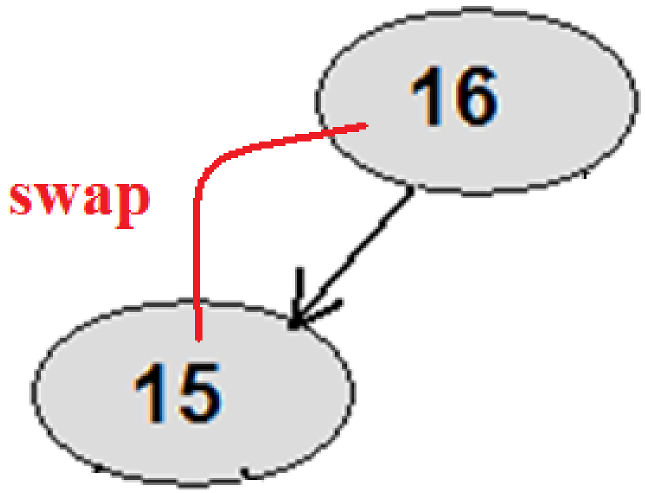
2,4,7



2, 4, 7, 9



2,4,7,9,11



Result

- 2, 4, 7, 9, 11, 15, 16

Important Property of Heap Sort

- Heap sort can overlap I/O operation with processing the heap tree.

- Read a record
- Insert the record into the heap tree while reading the second record

- Remove a record from root
- Write the record into the file while updating the heap tree

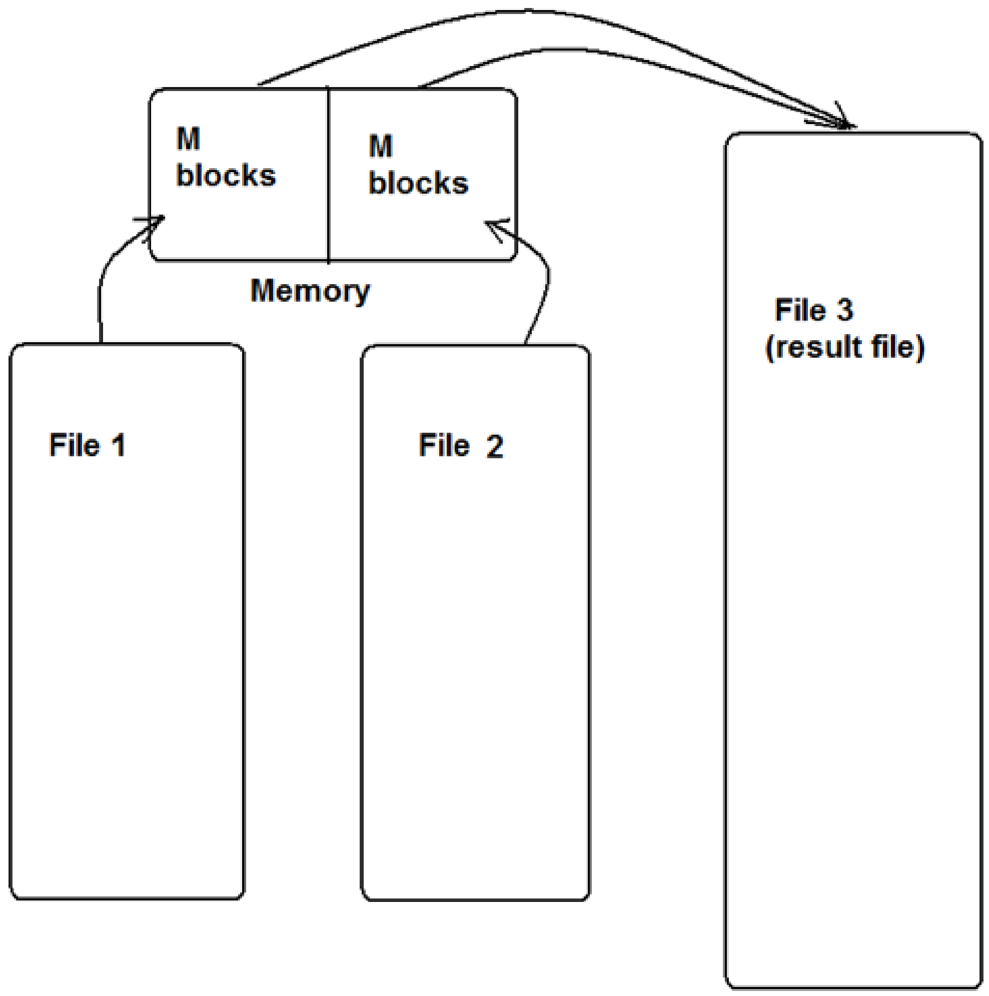
External Sorting

- In external sorting the data file is divided into Segments or Runs.
- Each Run should be small enough to be put in the memory
- Each Run is sorted and stored in a file.
- The sorted files are then merged into a single file
- Runs are deleted at the end.

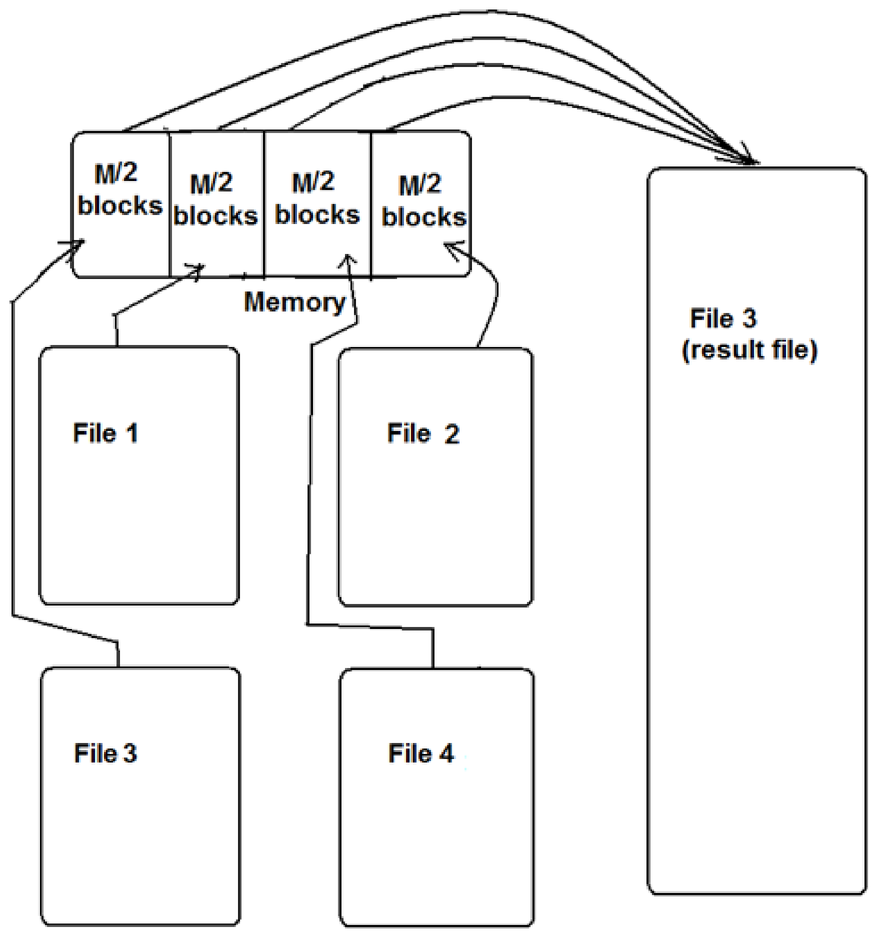
External Sorting

- Algorithm
  - Sort file in segments (Runs)
  - Merge sorted segments
- Larger runs will reduce merge time (less seek time)

Example: 2 way Merging



Example: 4 way Merge



Optimizing External Sorting

- The larger runs mean less seek time and faster merge
- The best algorithm for external sorting is the algorithm which can:
  - Generate larger runs (will be discussed next week)
  - Overlap I/O with sorting time

Questions?

Exam Questions

- (**15 points**) Explain the followings:
  - Bucket
  - Interleaving
  - Effective Block Transfer Time (ebt)

Exam Questions 1

- (**30 points**) A pile file contains 100,000 records of 256 bytes each. The block size is 2048 bytes. Besides, assume 25% deleted records are there in the pile file. The system administrator is asked to undelete 10 records. How can he undelete these records and how much time is needed for that.

- Use the following parameters provided from the disk specification
- s time : 18 msec
- r time : 10 msec
- btt : 0.12 msec
- ebt : 0.15 msec

Solution

- Solution 1: Find each record, change the DELETE mark
  - Blocking factor = 2048/256 = 8
  - Total number of records = 100,000 (active) + 25000 (deleted) = 125,000
  - Total number of blocks = 125,000 / 8 = 16,000
  - Find record = s+r+(b/2)*ebt
  - Change mark = 2r
  - Undelete 1 record = s+r+(b/2)*ebt + 2r
  - Undelete 10 records = (s+r+(b/2)*ebt + 2r) * 10

Solution

- Solution 2: Read all blocks, change the DELETE mark for the given 10 records
  - Blocking factor = 2048/256 = 8
  - Total number of records = 100,000 (active) + 25000 (deleted) = 125,000
  - Total number of blocks = 125,000 / 8 = 16,000
  - Read all blocks= s+r+b*ebt
  - Change marks = (2r) * 10
  - Undelete 10 records = s+r+b*ebt + (2r) * 10

Question 3

- **(25 points)** Assume the following key values will be sorted using quick sort. Show the first level of sorting the data. Show all changes to the list.
- 4, 11, 33, 22, 8, 20, 12, 5, 7, 9, 10, 3, 6, 1, 14, 13, 16, 15, 19, 18, 28, 17, 21, 2
- Pivot = 4
- Left = 11, right 2
- Swap 11,2
- Swap 33, 1
- Swap 22, 3
- Swap 3, 4 (pivot)

Question 4

- **(30 points)** Suppose that we want to update 100 records in a sorted sequential file having 200,000 records, with Bkfr=5. The updates include the key field values therefore, the file will not be sorted after these updates. Design a solution for these updates and compute the total time necessary.

- Use the following parameters:
- s time : 18 msec
- r time : 10 msec
- btt : 0.12 msec
- ebt : 0.15 msec

Solution

- Find each record
- Mark it as deleted
- Update the value record
- Add it to overflow area
- Repeat for all 100 records

- Find the records = (s+r+btt)*log(n) {n=200,000}
- Mark as deleted = 2r
- Add to overflow = s+r+btt+2r
- Total = 100*((s+r+btt)*log(n)+ 2r+ s+r+btt+2r)

Quiz

- Sort the following data using heap sort. Show all steps.

  23, 11, 3, 19, 2, 15, 14, 4, 18, 1, 8, 5