HEAPS.

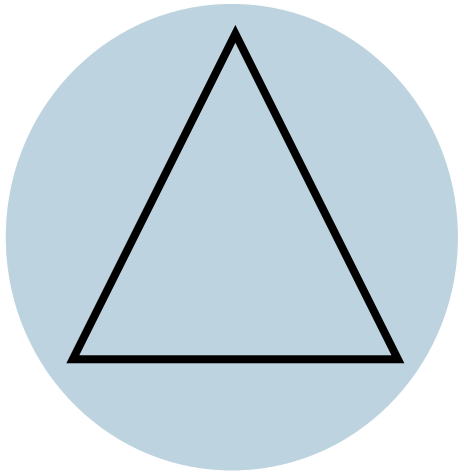
ALSO, A TREE.

Terminology.

Full Binary Tree.

Complete Binary Tree.

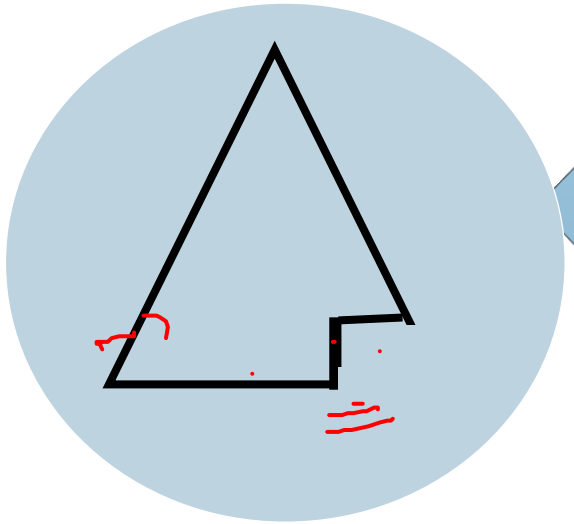
Full Binary Tree.



Every non-leaf node has two children

All the leaves are on the same level

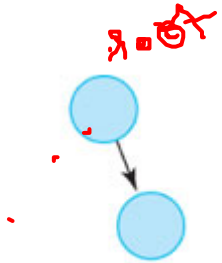
Complete Binary Tree.



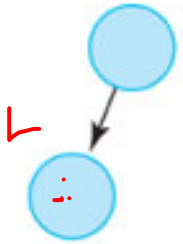
A binary tree that is either full or full through the next-to-last level

The last level is full from left to right (i.e., leaves are as far to the left as possible)

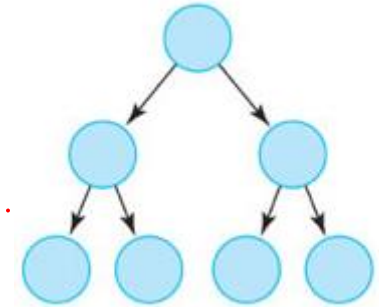
Full and complete



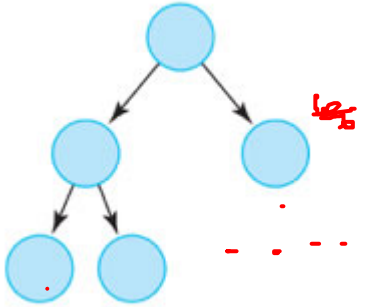
Neither full nor complete



Complete

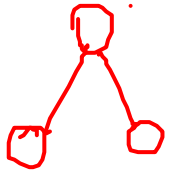


Full and complete



Complete

Heaps.

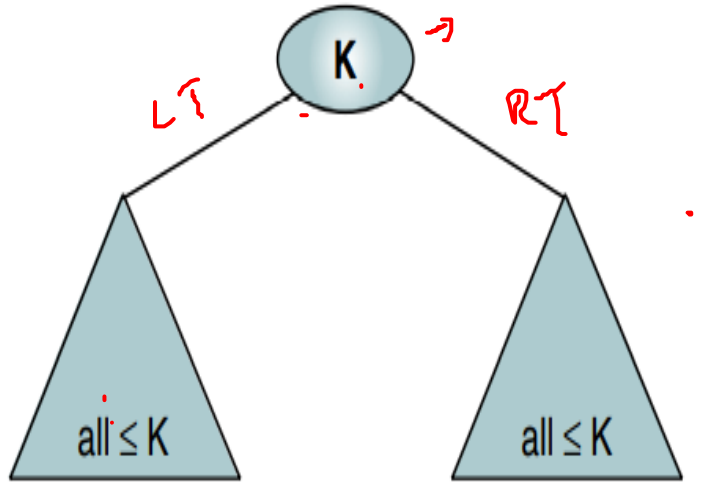


- A ***third type of tree*** is a heap.
- A heap is a binary tree whose left and right subtrees have values less than their parents.
- The root of a heap is guaranteed to hold the largest node in the tree; its subtrees contain data that have lesser values.
- Unlike the binary search tree, however, the **lesser-valued nodes of a heap can be placed on either the right or the left subtree**. Therefore, both the left and the right branches of the tree have the same properties.

Basic Concepts.

A heap is a binary tree structure *(heap ordered)* with the following properties:

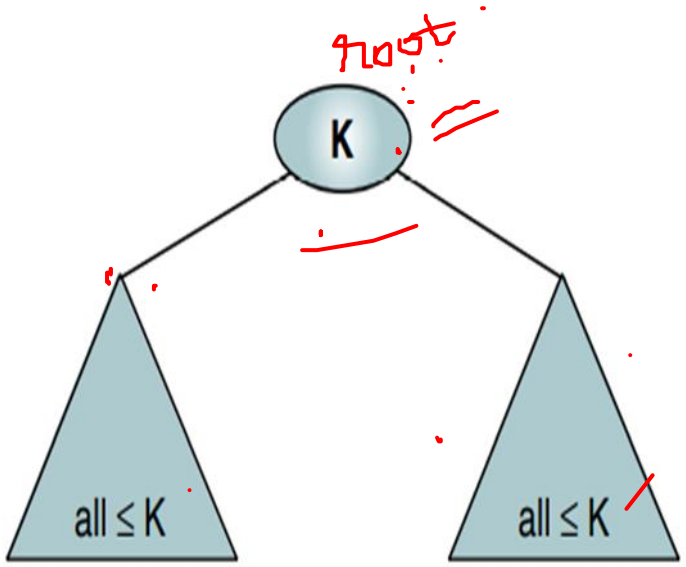
1. The tree is ***complete or nearly complete.***
2. The key value of each node is ***greater than or equal to the key*** value in each of its descendents.



Basic Concepts - Definition.

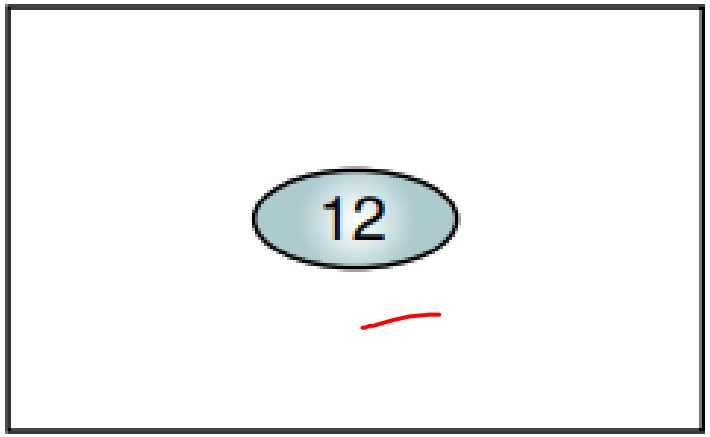
A heap is a ***complete or nearly complete binary tree*** in which the key value in a node is greater than or equal to the key values in all of its subtrees, and the subtrees are in turn heaps.

Heaps.

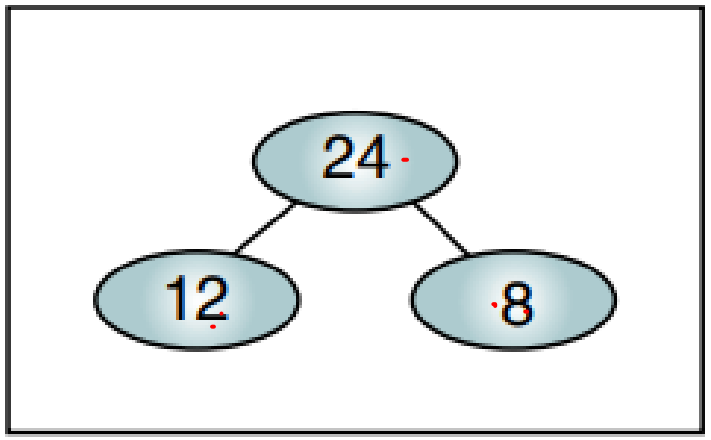


- Sometimes this structure is called a **max-heap.**
- The second property of a heap “key value is greater than the keys of the subtrees” can be reversed to create a min-heap.
- **Min heap**: *Key value in a node is less than the key values in all of its subtrees*.
- Generally speaking, whenever the **term heap** is used by itself, it **refers to a max-heap**.

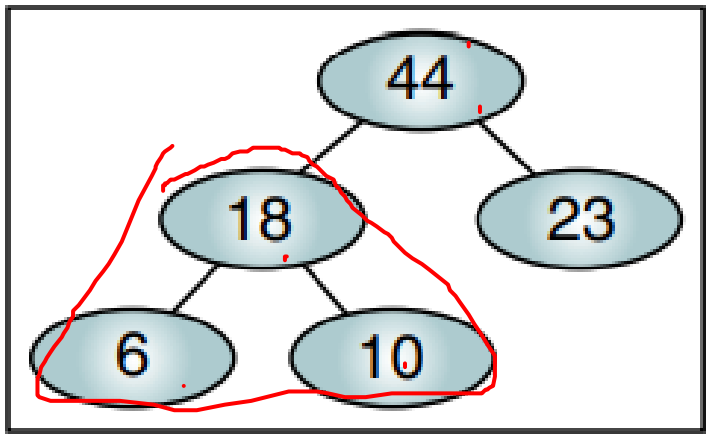
Examples of ***Valid*** Heap trees.



(a) Root-only heap

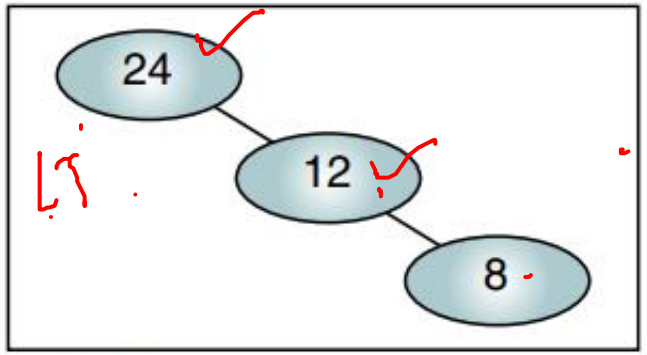


(b) Two-level heap

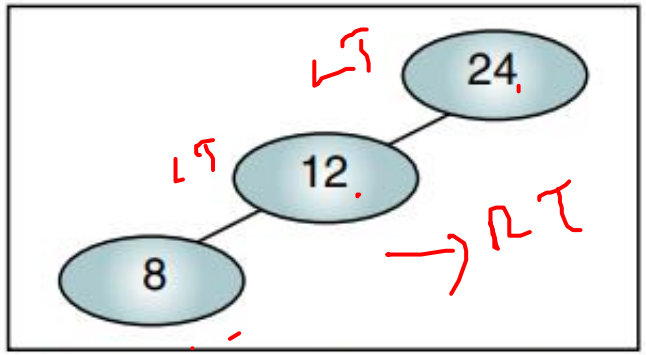


(c) Three-level heap

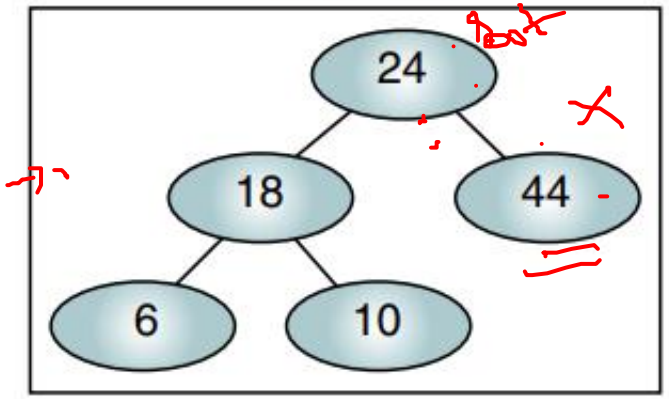
Examples of ***Invalid*** Heap trees.



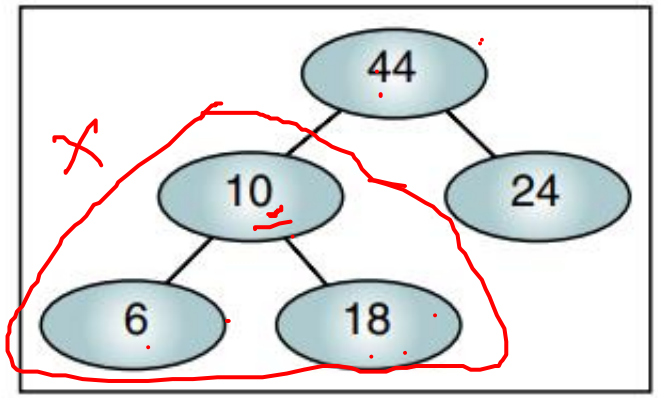
(a) Not nearly complete (rule 1)



(b) Not nearly complete (rule 1)

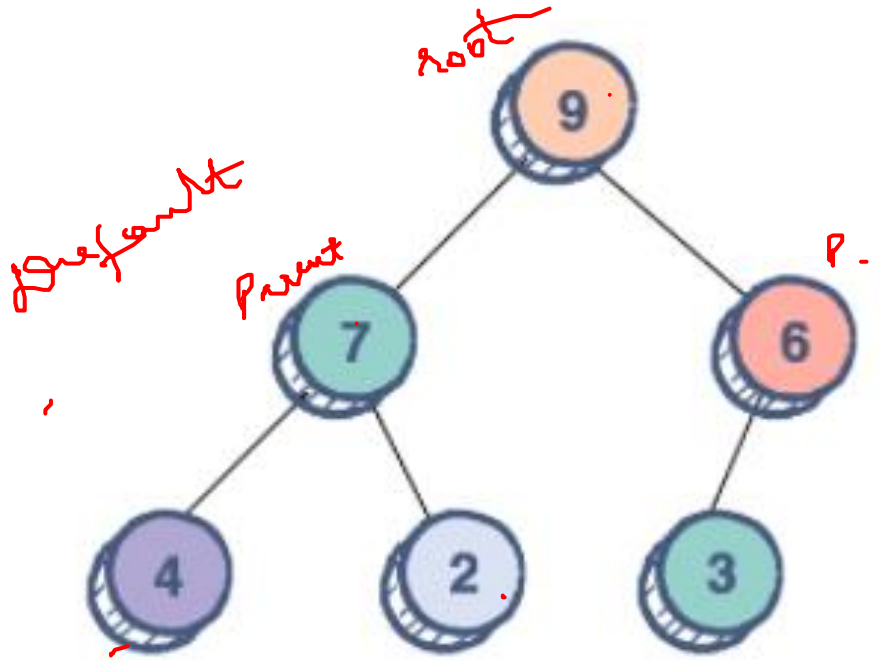


(c) Root not largest (rule 2)



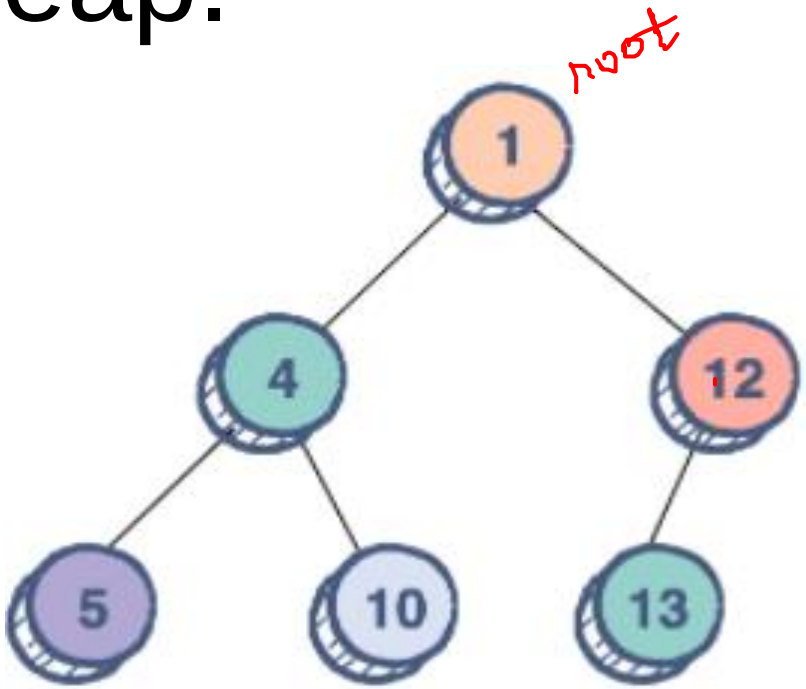
(d) Subtree 10 not a heap (rule 2)

Max Heap and Min Heap.



If Node A has a child node B, then,

$$key(A) >= key(B)$$



If Node A has a child node B, then,

$$key(A) <= key(B)$$

Source: https://www.educative.io/edpresso/what-is-a-heap

Some more terms.

Heapify: To rearrange a heap to maintain the heap property, that is, the key of the root node is more extreme (greater or less) than or equal to the keys of its children. If the root node's key is not more extreme, swap it with the most extreme child key, then recursively heapify that child's subtree. The child subtrees must be heaps to start.

or

Heapify is the process of converting a binary tree into a Heap data structure

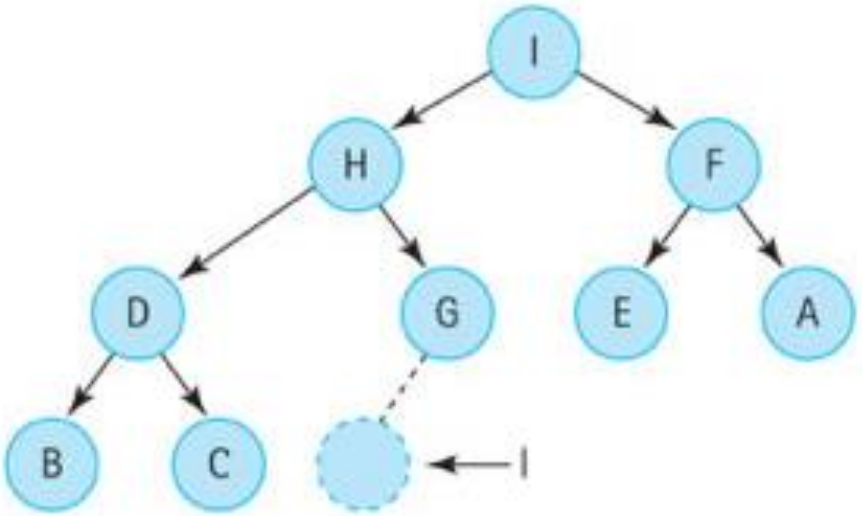
NIST definition.

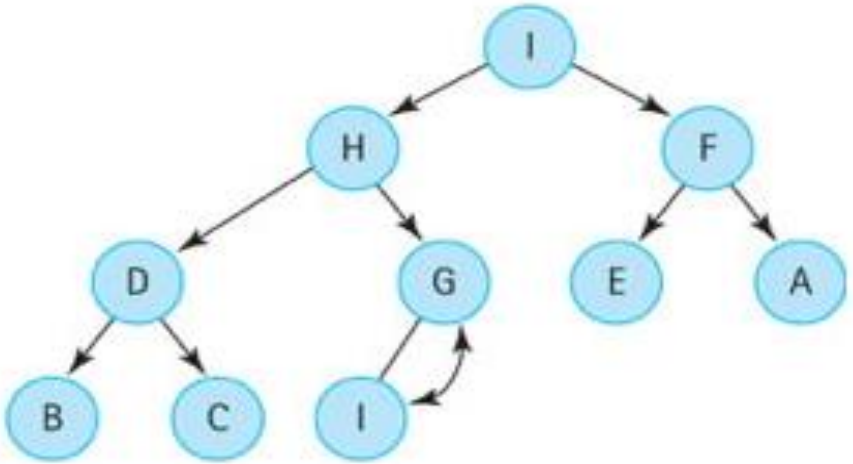
Some more terms.

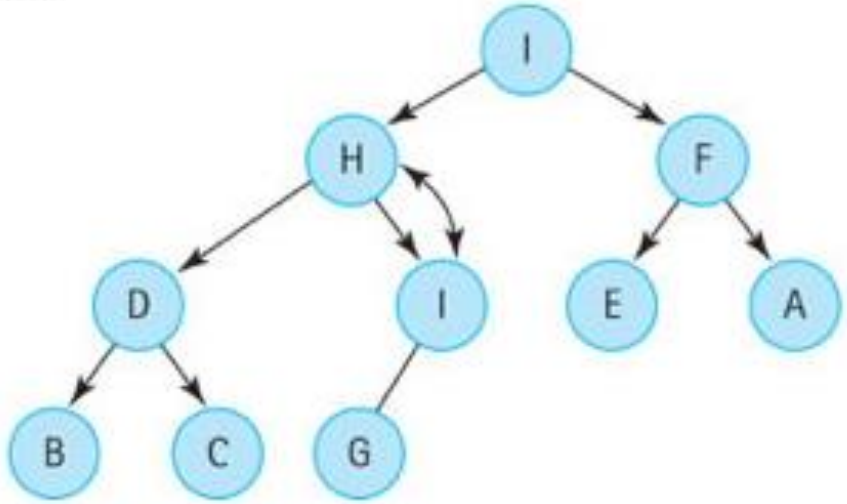
ReheapUp: Add the new item to the first empty leaf at the bottom level and re-establish the heap characteristic by swapping with the parent if the child should be the new parent. Repeat upward until the root is reached or no swap is necessary

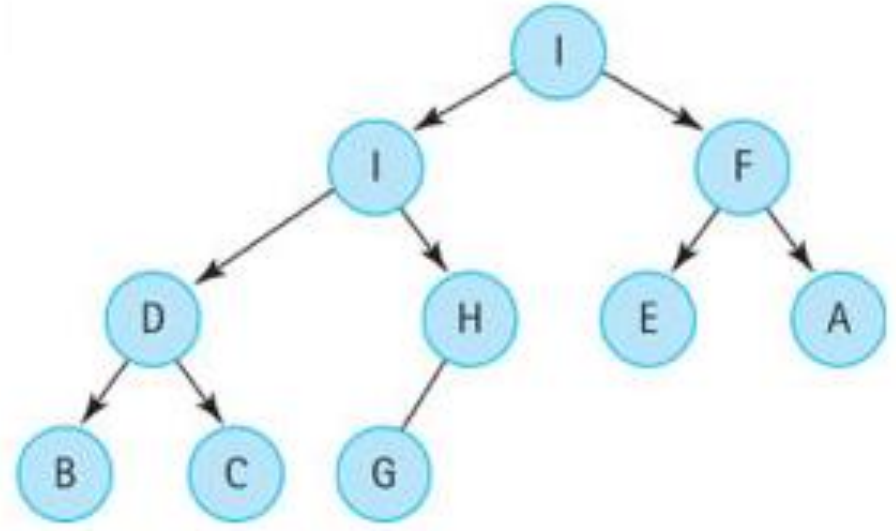
ReheapDown: Root is removed and then the heap is re-established by pulling upward the child that has the largest value. Repeat until no shift upward is necessary or the leaf is reached. Use the rightmost, bottom level value as the fill-in.

Reheap Up.



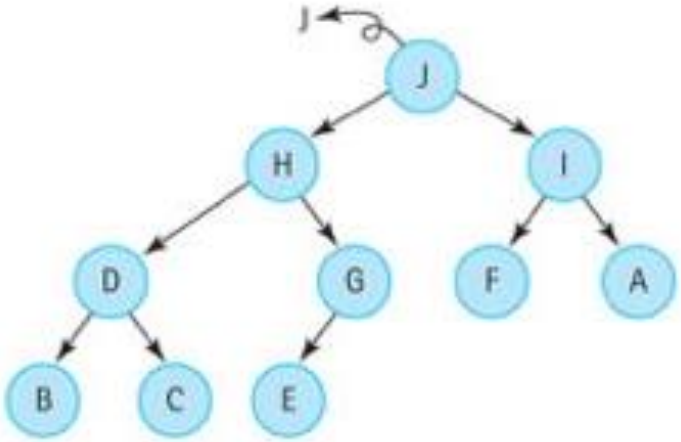


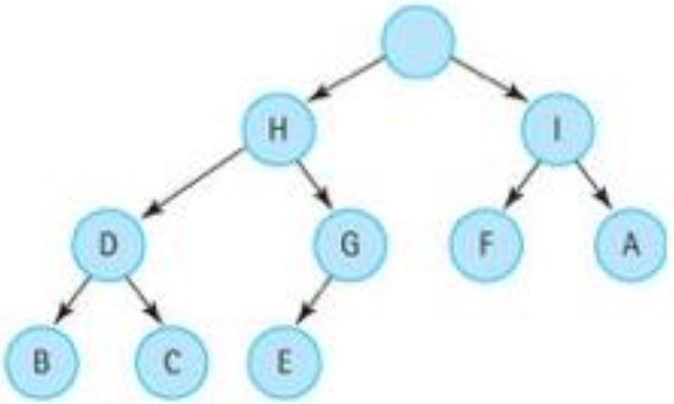


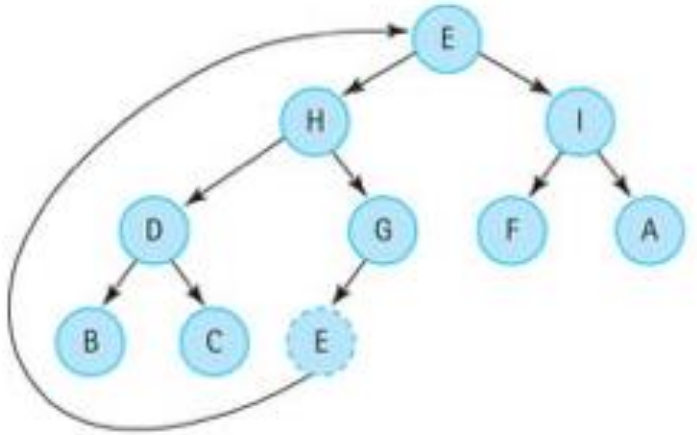


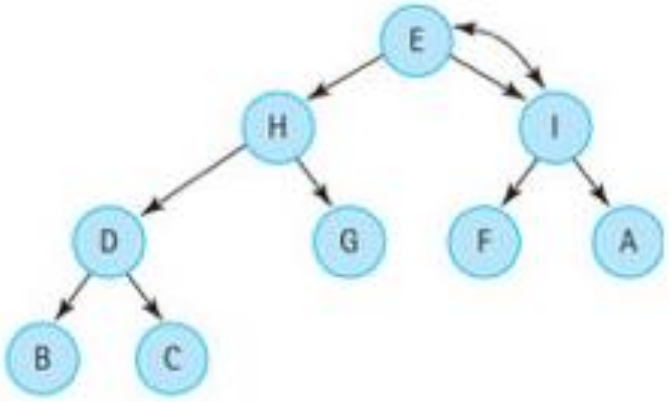
http://faculty.juniata.edu/kruse/cs240/pqheaps.htm

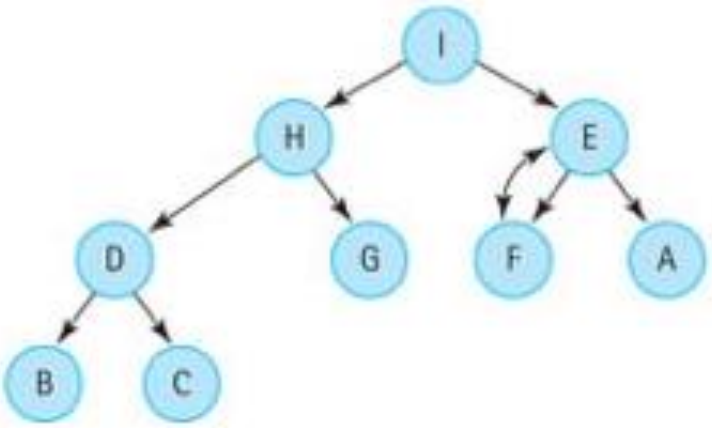
Reheap Down.

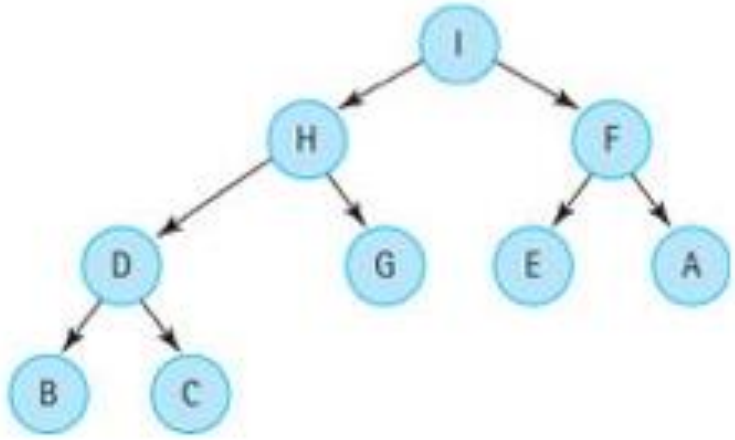












http://faculty.juniata.edu/kruse/cs240/pqheaps.htm

Heap Implementation.

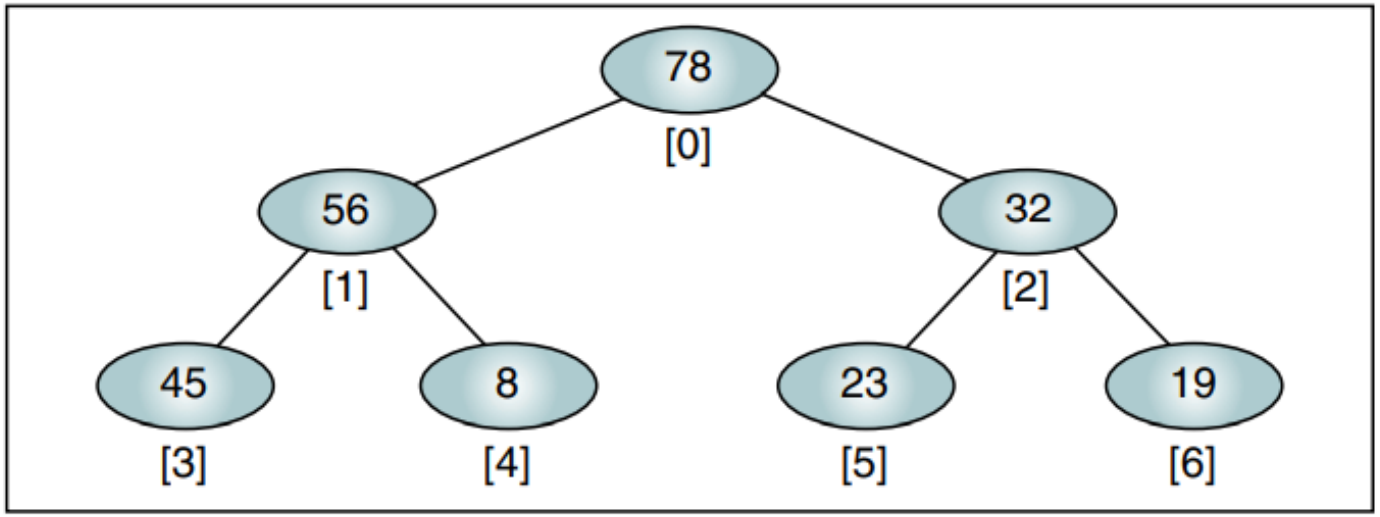
- Although a heap can be built in a dynamic tree structure, it is most often ***implemented in an array***.
- How ?

A heap can be implemented in an array because it must be a complete or nearly complete binary tree, which allows a fixed relationship between each node and its children.

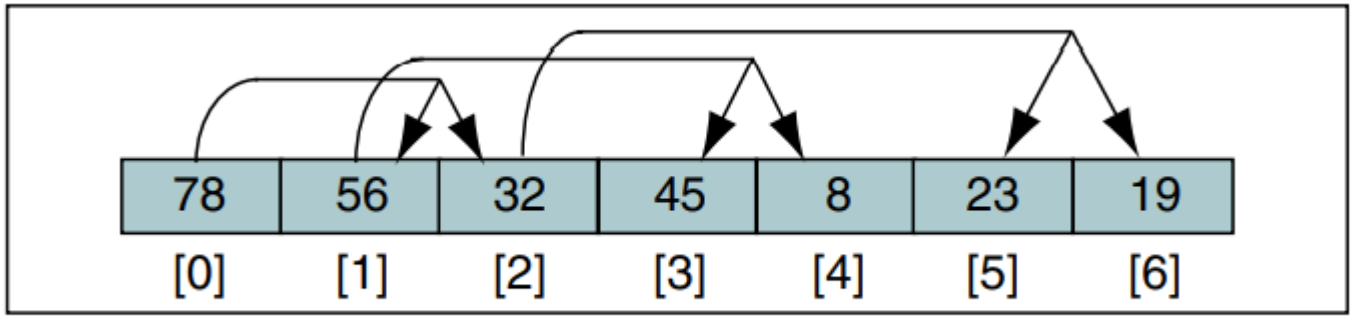
Heap Implementation Relationships.

1. For a node located at index i, its children are found at:
   a. Left child: $2i + 1$
   b. Right child: $2i + 2$
2. The parent of a node located at index i is located at $\lfloor (i - 1) / 2 \rfloor$.
3. Given the index for a left child, j, its right sibling, if any, is found at $j + 1$. Conversely, given the index for a right child, k, its left sibling, which must exist, is found at $k - 1$.
4. Given the size, n, of a complete heap, the location of the first leaf is $\lfloor (n / 2) \rfloor$.
5. Given the location of the first leaf element, the location of the last nonleaf element is one less.

Heaps in Arrays.



(a) Heap in its logical form



(b) Heap in an array

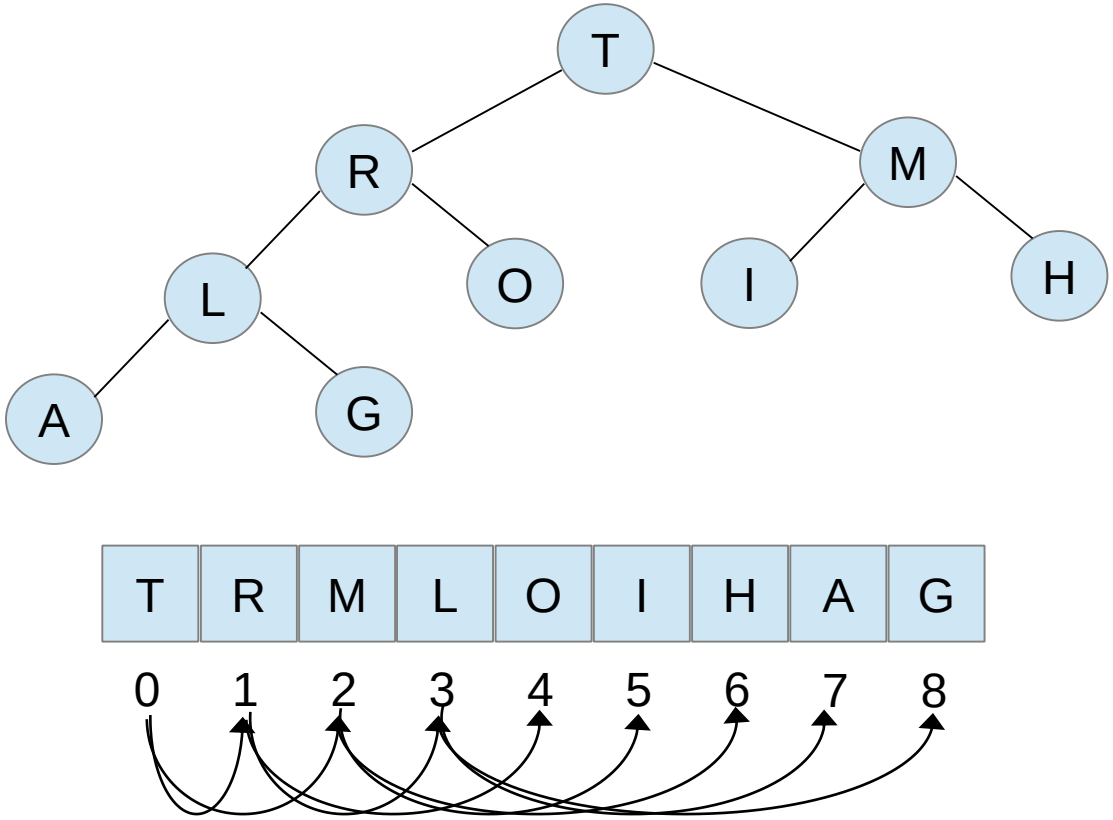
Heaps in Arrays.

1. The index of 32 is 2, so the index of its left child, 23, is 2 x 2 + 1, or 5. The index of its right child, 19, is 2 x 2 + 2, or 6 (Relationship 1
2. The index of 8 is 4, so the index of its parent, 56, is $\lfloor (4 - 1) / 2 \rfloor$, or 1 (Relationship 2).
3. In the first example, we found the address of the left and the right children. To find the right child, we could also have used the location of the left child (5) and added 1 (Relationship 3).
4. The total number of elements is 7, so the index of the first leaf element, 45, is $\lfloor (7 / 2) \rfloor$, or 3 (Relationship 4).
5. The location of the last nonleaf element, 32, is 3 – 1, or 2 (Relationship 5).

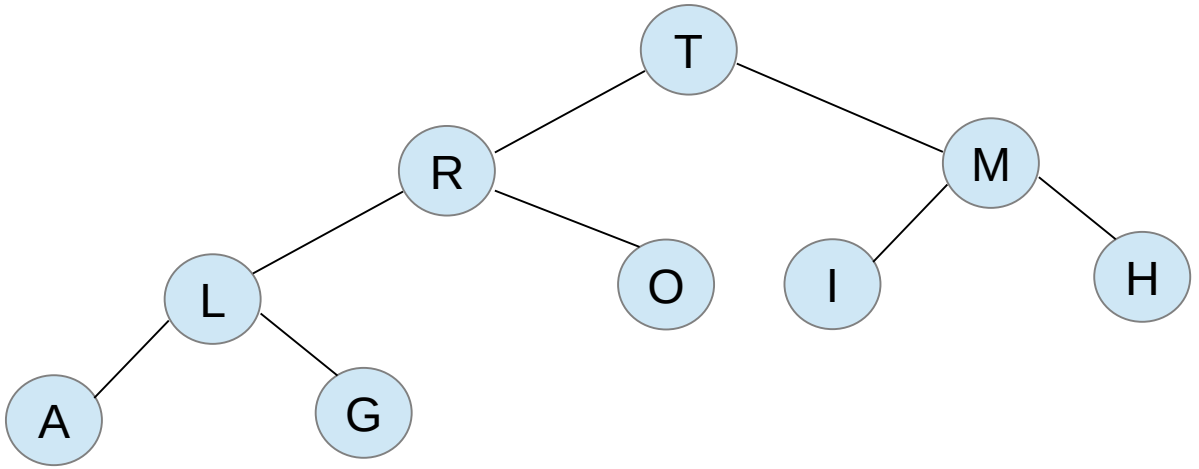
Operations on Heaps.

- Insertion
- Deletion

Heap Implementation (Example 3).



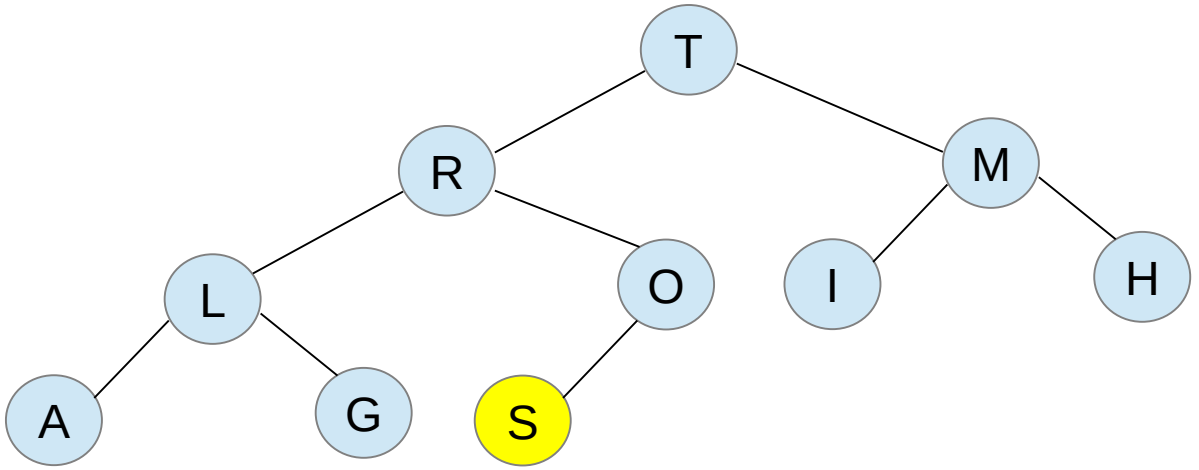
Heapification.



T	R	M	L	O	I	H	A	G
0	1	2	3	4	5	6	7	8

Heap-broken

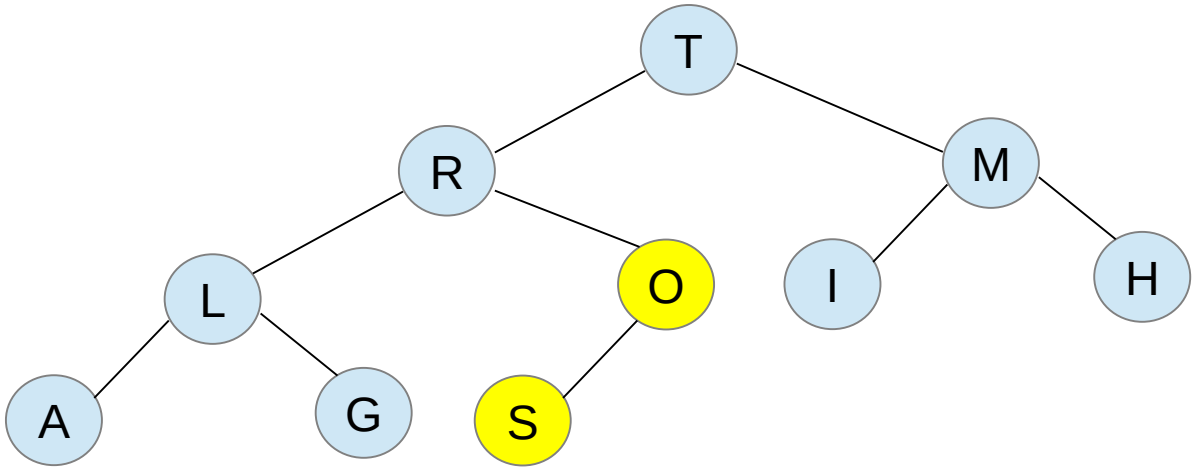
Heapification.



T	R	M	L	O	I	H	A	G	S
0	1	2	3	4	5	6	7	8	9

Heap-broken

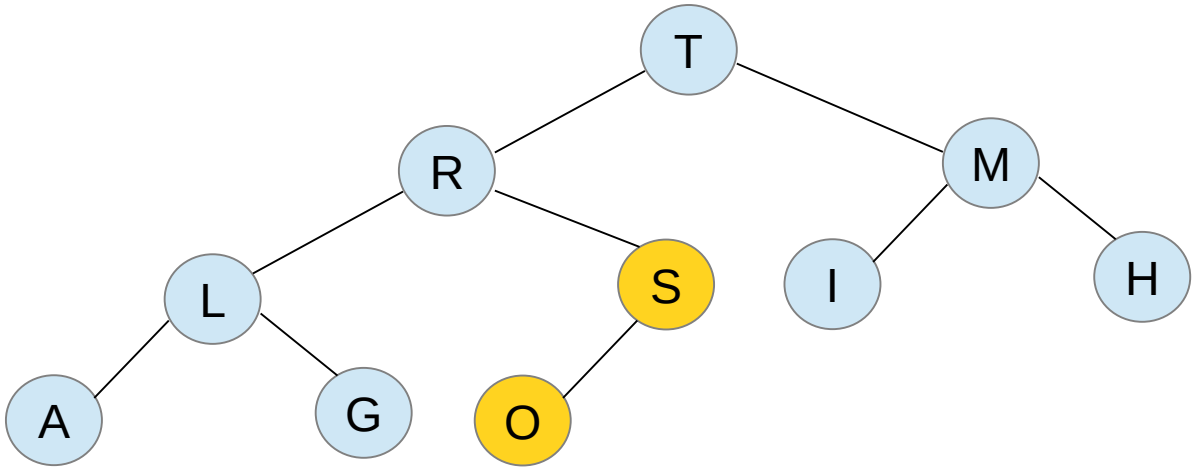
Heapification.



T	R	M	L	O	I	H	A	G	S
0	1	2	3	4	5	6	7	8	9

Heap-broken

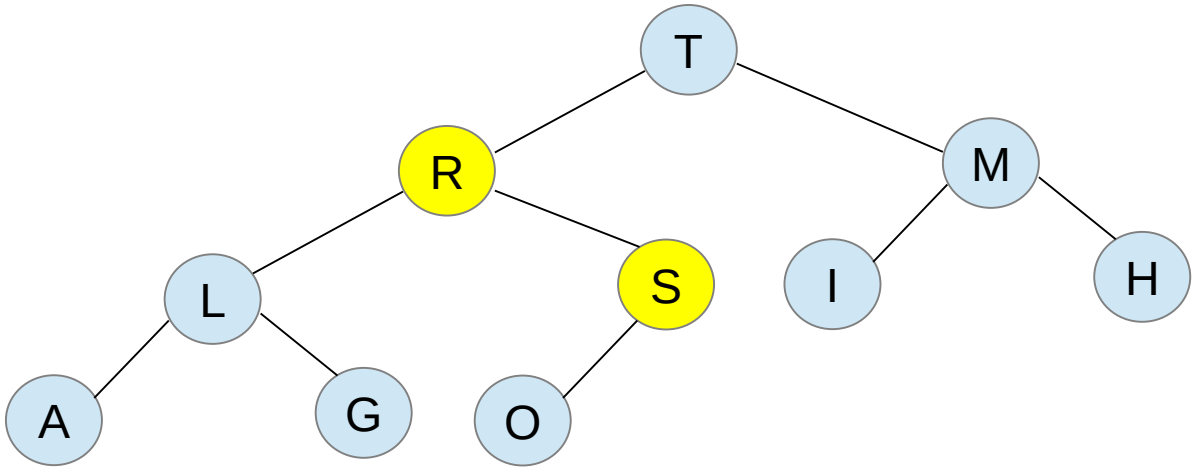
Heapification.



T	R	M	L	S	I	H	A	G	O
0	1	2	3	4	5	6	7	8	9

Heap-broken

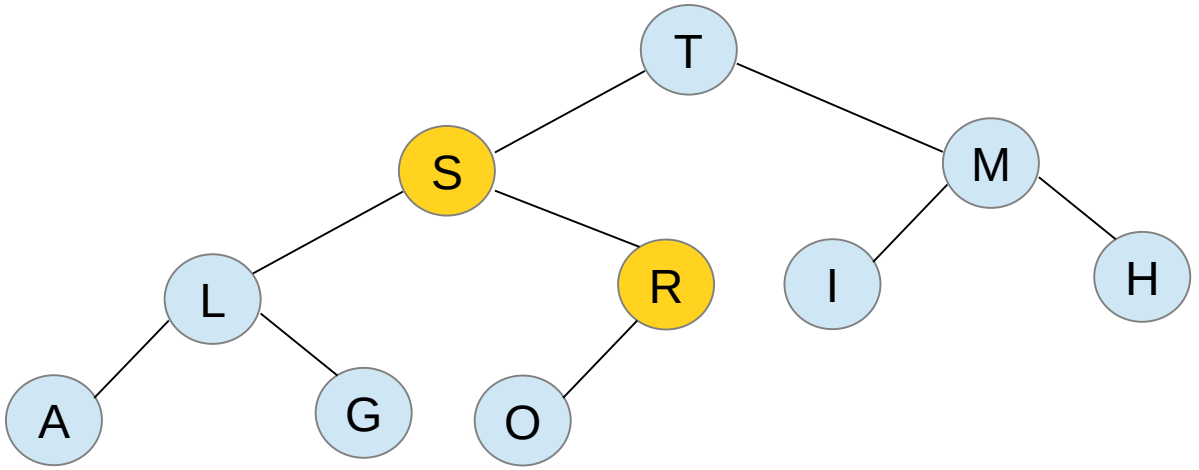
Heapification.



T	R	M	L	S	I	H	A	G	O
0	1	2	3	4	5	6	7	8	9

Heap-broken

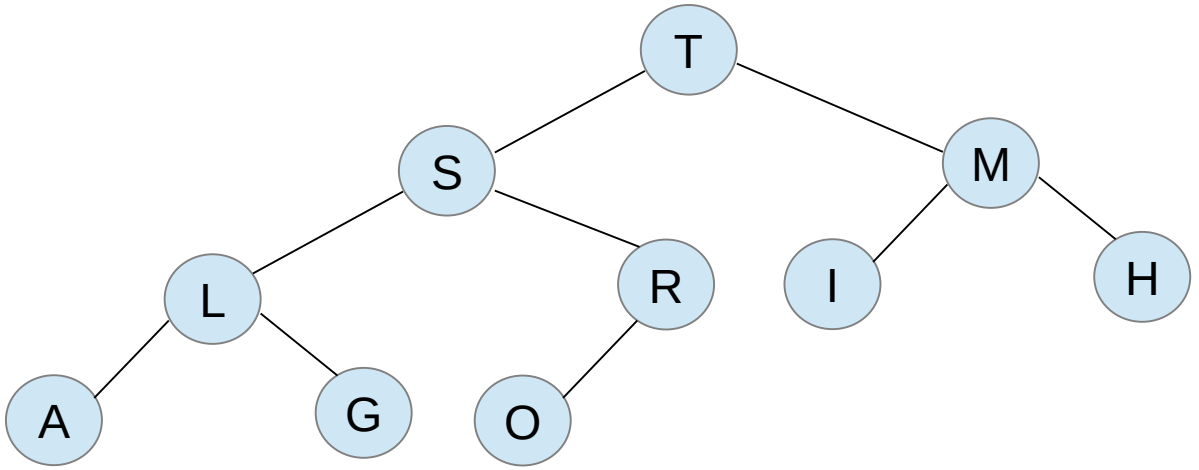
Heapification.



T	S	M	L	R	I	H	A	G	O
0	1	2	3	4	5	6	7	8	9

Heap-broken

Heapification.



T	S	M	L	R	I	H	A	G	O
0	1	2	3	4	5	6	7	8	9

Heap-fixed.

THANK YOU