Joanna Klukowska
joannakl@cs.nyu.edu

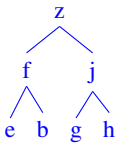
DNHI Homework 7 Solutions
Trees

Problem 1

A priority queue containing characters is implemented using max-heap and stored as an array. The capacity of the array used is 10 elements and the first 7 locations are occupied (indexes 0 through 6). Show the array after each of the following operations is performed (each operation should be modifying the array resulting from the previous step).

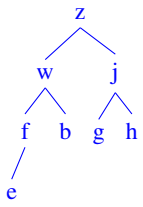
index	0	1	2	3	4	5	6	7	8	9
value	'z'	'f'	'j'	'e'	'b'	'g'	'h'			

The tree picture of this priority queue:



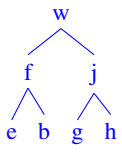
Assume that the name of the priority queue is `pq`.

1. `pq.enqueue('w');`



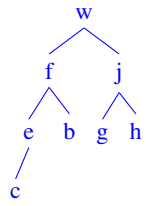
[z, w, j, f, b, g, h, e, ,]

2. `pq.dequeue();`



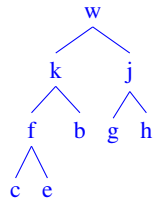
[w, f, j, e, b, g, h, , ,]

3. `pq.enqueue('c');`



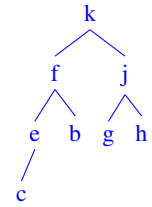
[w, f, j, e, b, g, h, c, ,]

4. `pq.enqueue('k');`



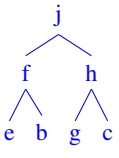
[w, k, j, f, b, g, h, c, e,]

5. `pq.dequeue();`



[k, f, j, e, b, g, h, c, ,]

6. `pq.dequeue();`



[j, f, h, e, b, g, c, , ,]

Problem 2

Given the array representing the max-heap in problem 1, show the tree representation of the heap. Show the tree representation of the final heap in problem 1 (after all of the operations are performed).

See the answers in problem 1.

Joanna Klukowska
joannakl@cs.nyu.edu

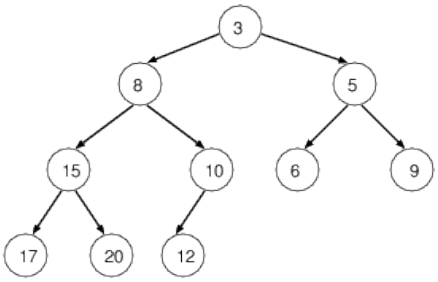
Problem 3

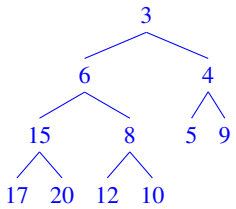
Specify the steps needed to sort a list of last names stored in a text file. You should use heap-sort. For each step, specify the complexity (using order notation) of the operations.

```
1
2   for each name in the input file                    // O(N )!!! amortized time
3     add the word to a min-heap                       //          for all additions
4
5   create an array large enough to tore all elements
6
7   for each element of the min-heap                   // O(N logN) for all removals
8     dequeue the smallest element                     // O(logN) for each removal
9     add it to the next location in the array         // O(1) to add it to array
10
```

Problem 4

Given the following min-heap, show the final state of the heap after executing the following operations: `enqueue(1)`, `enqueue(4)`, `dequeue()`.





[3, 6, 4, 15, 8, 5, 9, 17, 20, 12, 10, ...]

Problem 5

Specify the characteristic properties and requirements of the following trees:

- tree
  It's a data structures with connection from a parent node to a child node. There is a special node called root. There is a single path from every other node to the root.
- binary tree
  It's a tree. Each node has at most two children.
- binary search tree
  It's a binary tree. All nodes in a left subtree (a right subtree) are smaller than the node and all nodes in a right subtree (a left subtree) are larger than or equal to the node.
- AVL tree
  It's a BST. The balance factor of each node has to be between -1 and 1. (Balance factor is the difference in height between the two subtrees of a node.)
- heap
  It's a complete binary tree. For each node its children as smaller (min-heap) or larger (max-heap) than it.

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 6

Given the following node definition for a general tree

```
Node
  int data
  Node firstChild
  Node sibling
```

write a method that given a reference to the node prints to the screen the values stored in all of this node's children.

```
1
2  void printChildren ( Node n )
3    if n == null
4      return
5    if n.firstChild == null    //no children
6      return
7    Node current = n.firstChild
8    while current != null
9      print current.data
10     current = current.sibling
11
```

Problem 7

Enter the following nodes into an AVL tree: 45, 12, 3, 98, 20, 30, 75, 55, 100, 0, 60, 85. For each node that triggers a rotation, state what node it is and show the tree after the necessary rotation.

Nodes that become out of balance are marked with red circles.

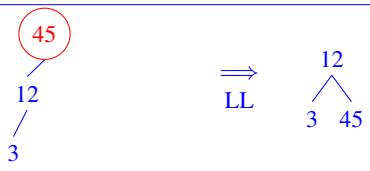
add 45: 45

---

add 12:

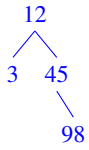
---

add 3:



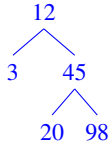
---

add 98:



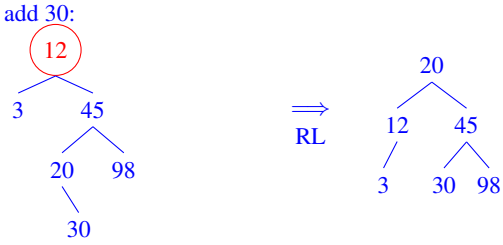
---

add 20:



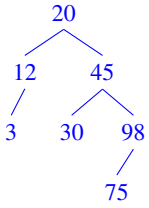
---

add 30:



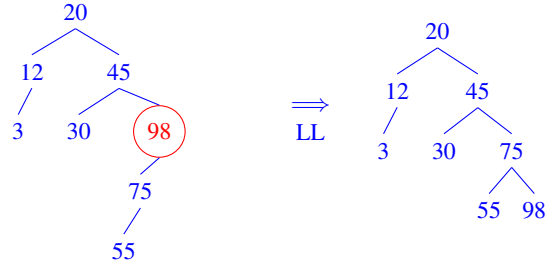
---

add 75:



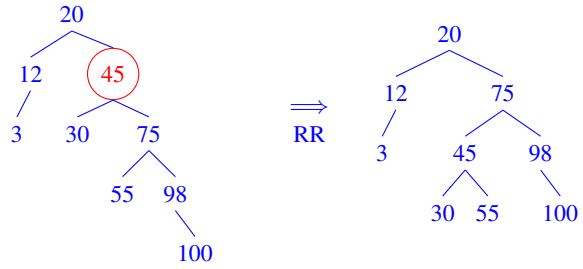
---

add 55:

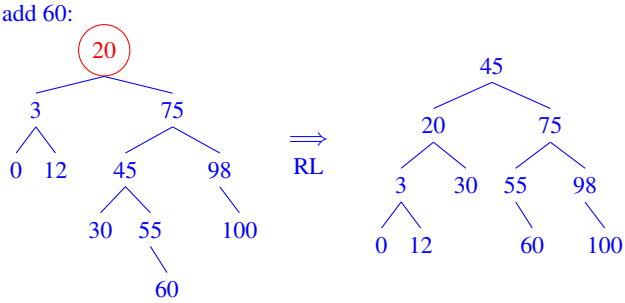


Joanna Klukowska
joannakl@cs.nyu.edu

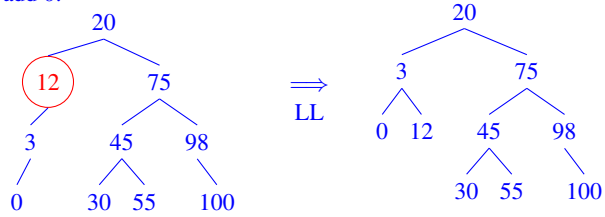
add 100:



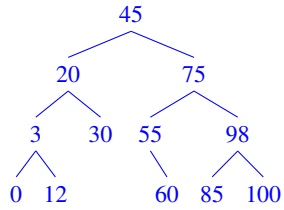
add 60:



add 0:

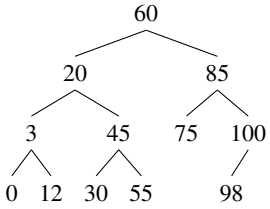


add 85:

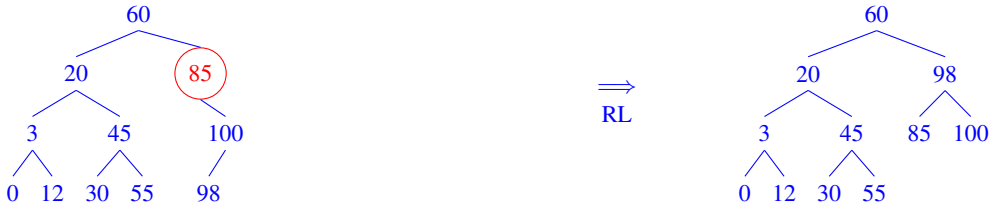


Problem 8

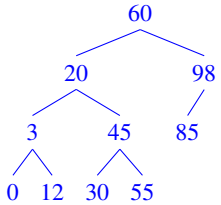
Given the AVL tree below show what the tree looks like after each of the following nodes is removed: 75, 100, 98.



remove 75:



remove 100:



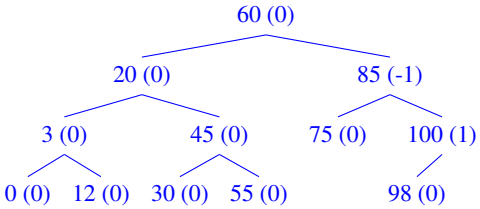
remove 98:

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 9

For each node in the tree in problem 8, state what the balance factor of that node is.

balance factor indicated in parenthesis next to the node:



Problem 10

Given the following key values: 439, 340 129, 342, 278, 947, 371 and a hash table stored in an array of size 10, show where in the array each of the keys is going to be stored if the hash function is computed as `h(x) = x % 10` and the following collision resolution method is used:

- separate chaining

index	0	1	2	3	4	5	6	7	8	9
value	|	|	|						|	|
	430	371	342					947	278	129
										439

- linear probing

index	0	1	2	3	4	5	6	7	8	9
value	340	129	342	371				947	278	439

Problem 11

Explain what load factor has to do with a performance of a hash table.

Load factor is a fraction of table locations occupied by an actual element. It is an indicator of how full the table is. The lower the load factor, the fewer collisions are expected. As the load factor increases, it becomes harder to find appropriate location for an item (linear/quadratic probing) or the chains become longer (separate chaining).

Problem 12

Given a heap with 3 levels (level 0, level 1 and level 2) what is the largest number of nodes that the heap may contain? what is the smallest number of nodes that the heap may contain?

The largest number of nodes are in a full tree (all levels full): $2^3 - 1 = 7$ nodes.
The smallet number of nodes are in a complete tree in which the last level has only one node: $(2^2 - 1) + 1 = 4$ nodes.

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 13

Consider the implementation of a tree where each node can have arbitrary many children. In the data structure used for storing the tree the children of a given node are organized in a singly lined list using the **`sibling`** field and the parent has **`firstChild`** field. Nodes are storing integer values. The declaration of a **`Node`** class is as follows:

```
class Node {
  int data;
  Node firstChild;
  Node sibling;
}
```

Draw the tree that is represented by the following nodes.