Written Assignment 2

1. (15’) Quick Sort

Suppose that the quicksort algorithm is performed on the following array of integers.

6	2	7	3	4	5	1

a) (5 marks) What is the best choice of the pivot for the first round of quicksort?
Best pivot: 4

b) (5 marks) If the “median of three” strategy is used for pivot selection, what is the value of the pivot for the first round of quicksort?
Median of 3 pivot: 3

c) (5 marks) If the “median of three” strategy is used for pivot selection, what is the content of the array after the partition procedure in the first round of quicksort? Please strictly follow the partition algorithm introduced in the class.

1	2	3	5	4	7	6

2. Heap Sort

Sort the array {*1, 6, 3, 5, 4, 2*} with a **maximum** heap without using extra memory. Write down the content of the array every time an *insert()* or a *deleteMax()* operation completes. The initial state and the array content after the first two insertions are already written for you:

```
0: initial state

          | 1 | 6 | 3 | 5 | 4 | 2 |

     1: insert (1)

          | 1 | 6 | 3 | 5 | 4 | 2 |

     2: insert (6)

          | 6 | 1 | 3 | 5 | 4 | 2 |

     3: insert (3)

          | 6 | 1 | 3 | 5 | 4 | 2 |

     4: insert (5)

          | 6 | 5 | 3 | 1 | 4 | 2 |

     5: insert (4)

          | 6 | 5 | 3 | 1 | 4 | 2 |
     6: insert (2)

          | 6 | 5 | 3 | 1 | 4 | 2 |

     7: deleteMax ()

          | 5 | 4 | 3 | 1 | 2 | 6 |

     8: deleteMax ()

          | 4 | 2 | 3 | 1 | 5 | 6 |

     9: deleteMax ()

          | 3 | 2 | 1 | 4 | 5 | 6 |

     10: deleteMax ()

          | 2 | 1 | 3 | 4 | 5 | 6 |

     11: deleteMax ()

          | 1 | 2 | 3 | 4 | 5 | 6 |

     12: deleteMax ()

          | 1 | 2 | 3 | 4 | 5 | 6 |

    (2 marks per step)
```

3. Binary Search Trees

Suppose that every node of a binary search tree has 3 pieces of information:

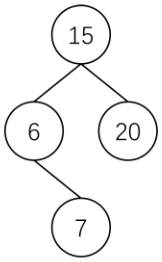
- key: an integer which is the key of the node
- left: a pointer which points to the left child of the node
- right: a pointer which points to the right child of the node

Write the pseudo code for a function, *printKeys*, which prints all the keys on the binary search tree in DESCENDING order.

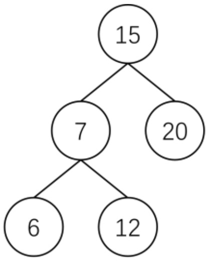
```
printKeys (root)
    IF root=NULL
            RETURN
    printKeys(root.right)
    print(root.key)
    printKeys(root.left)
```

4. AVL Trees

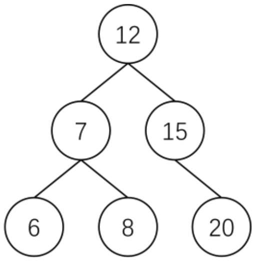
Construct an AVL tree by inserting the input array {6, 20, 15, 7, 12, 8}. Draw the three trees after inserting the last three elements: 7, 12, and 8, respectively.



After inserting 7



After inserting 12



After inserting 8

5. B+ Trees

5.1. Suppose you are managing employee records on a computer with the following setting:

- Computer hard disk access is block-based and the size of one block is *2048 bytes*.
- The size of each employee record is *256 bytes* (including the primary key).
- The primary key for an employee record is of type *long long (8 bytes).*
- The size of a pointer is 4 bytes.

You decide to store the data using a B+ tree. Propose the best setting for *M* and *L.* Show the steps that lead to your proposal.
Note: The definition of *M* and *L is* as described in the lecture slides.

For efficient access, the size of a B+ tree node, including the internal node and the leaf, should be that of a block. Therefore, the size of a B+ tree node is as close as possible to, but not greater than 2048 bytes.

- A full leaf node contains ⌊2048/256⌋ = 8 records, so L = 8.
- A full internal node contains M pointers to its children and M-1 keys, so its size is:

  M * 4 + (M-1) * 8 <= 2048

  Then we have M <= 171.3, so M = 171.

5.2. In this question, we use a B+ tree with M=L=3. The initial B+ tree is shown below

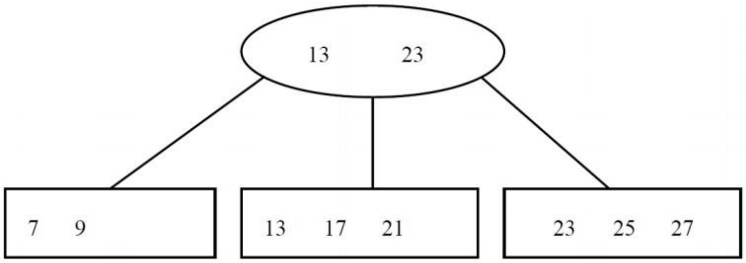
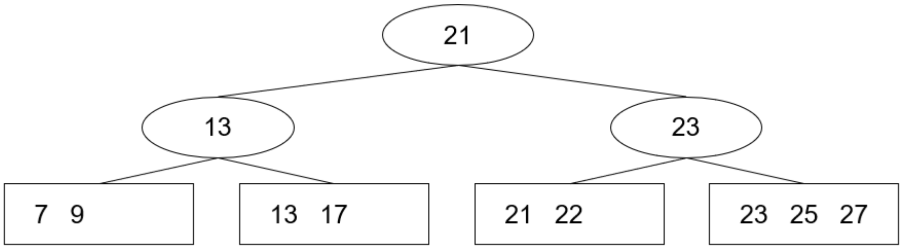
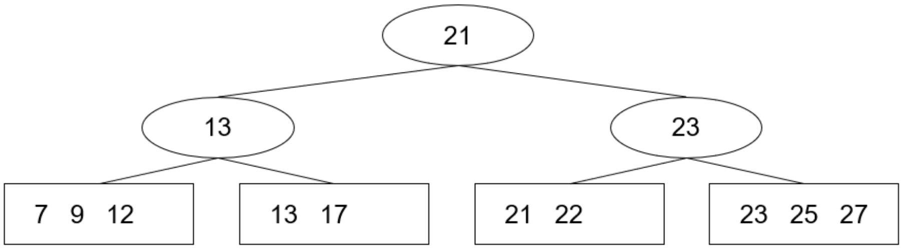


Figure 1: The initial B+ Tree

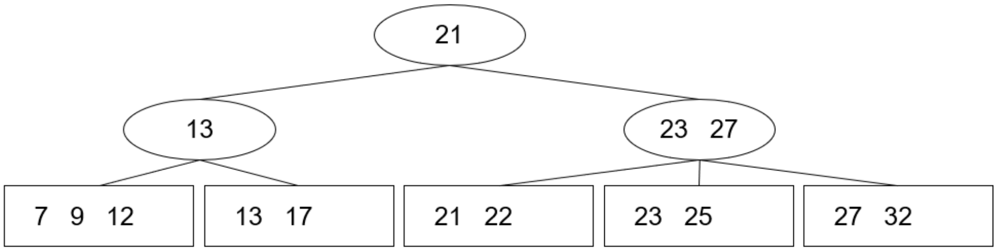
a) Given an insertion sequence *{22, 12, 32}*, draw the three B+ trees after each insertion, respectively. Please start from the initial B+ tree shown in Figure 1. *Note: please strictly follow the lecture notes when you do the operations.*



After inserting 22

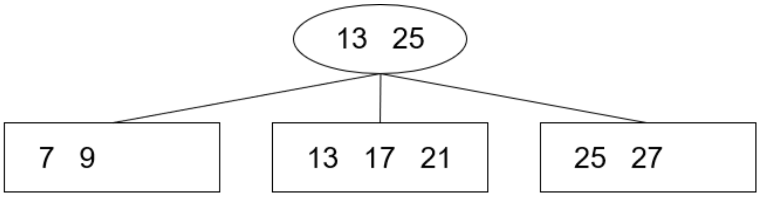


After inserting 12

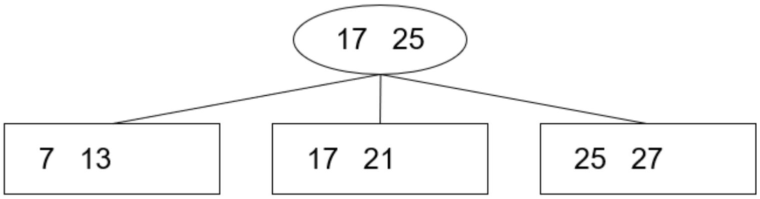


After inserting 32

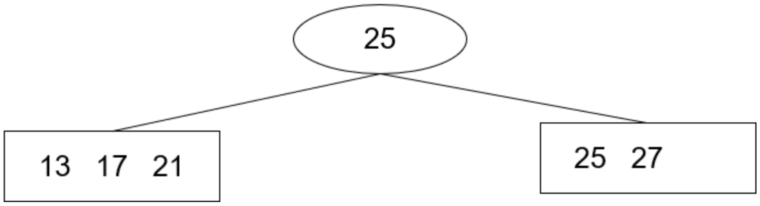
b) Given a deletion sequence *{ 23, 9, 7}*, draw the three B+ trees after each deletion. Please start from the initial B+ tree shown in Figure 1. *Note: please strictly follow the lecture notes when you do the operations.*



After deleting 23



After deleting 9



After deleting 7