Written assignment 2

---

1. (17') Divide and Conquer

Suppose you are given an array A[1..n] of sorted integers that has been circularly shifted k positions to the right. For example, [31,43,3,17,26,28,29] is a sorted array that has been circularly shifted k = 2 positions, while [26,28,31,43,3,17] has been shifted k = 4 positions. We can obviously find the largest element in A in O(n) time. Describe an O(logn) algorithm by using a divide and conquer solution.

Solution:

1. Divide the array A into two part A_L and A_R.
2. Compare the first element of A_L and A_R, if $A_L[0] > A_R[0]$, then the largest element is in A_L; if $A_L[0] < A_R[0]$, then the largest element is in A_R.
3. Then recursively search for the largest element in A_L or A_R until meet the base case: the array has only one element.

findMax(Array A, left, right){ //left and right are index of left and right element in A
    IF left == right: //base case
        return A[left]
    ELSE:
        IF A[left] > A[$\lfloor$(left + right)/2$\rfloor$]: //$A_L[0] > A_R[0]$
            return findMax(A, left, $\lfloor$(left + right)/2$\rfloor$)
        ELSE: //$A_L[0] < A_R[0]$
            return findMax(A, $\lfloor$(left + right)/2$\rfloor$, right)
}

2. (18') Heap Sort

Sort the array {1, 3, 7, 4, 5, 6} 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 | 3 | 7 | 4 | 5 | 6 |
1: insert (1)
| 1 | 3 | 7 | 4 | 5 | 6 |
2: insert (3)
| 3 | 1 | 7 | 4 | 5 | 6 |
```

Solution:

```
3: insert (7)
```

```
| 7 | 1 | 3 | 4 | 5 | 6 |
4: insert (4)
| 7 | 4 | 3 | 1 | 5 | 6 |
5: insert (5)
| 7 | 5 | 3 | 1 | 4 | 6 |
6: insert (6)
| 7 | 5 | 6 | 1 | 4 | 3 |
7: deleteMax()
| 6 | 5 | 3 | 1 | 4 | 7 |
8: deleteMax()
| 5 | 4 | 3 | 1 | 6 | 7 |
9: deleteMax()
| 4 | 1 | 3 | 5 | 6 | 7 |
10: deleteMax()
| 3 | 1 | 4 | 5 | 6 | 7 |
11: deleteMax()
| 1 | 3 | 4 | 5 | 6 | 7 |
12: deleteMax()
| 1 | 3 | 4 | 5 | 6 | 7 |
```

3. (10') Binary Search Trees

Verify the binary search tree. Given a binary tree `root` node, determine whether it is a valid binary search tree by writing the pseudo code of the function
`Boolean isBST(root, lower, upper).`

A valid binary search tree is defined as follows:

a) The `left` subtree of a node only contains numbers less than the current node.
b) The `right` subtree of a node only contains numbers greater than the current node.
c) All left and right subtrees must also be binary search trees.

Note: The number of nodes in the tree is in the range [MIN_VALUE, MAX_VALUE], and MIN_VALUE <= `Node.key` <= MAX_VALUE. In the function `isBST(root, lower, upper)`, `lower` and `upper` are the two boundaries of the `keys` of a valid BST subtree.

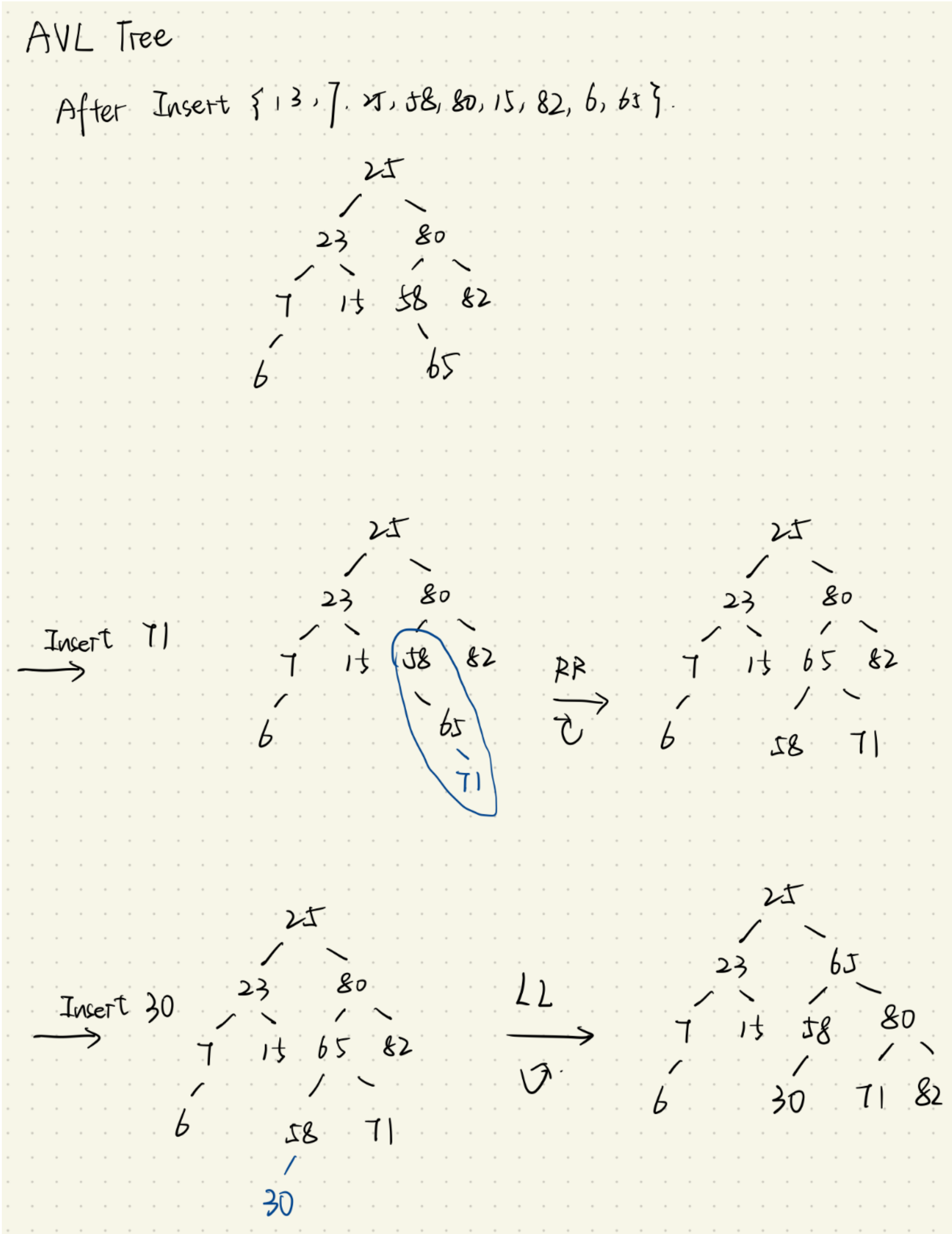
Solution:

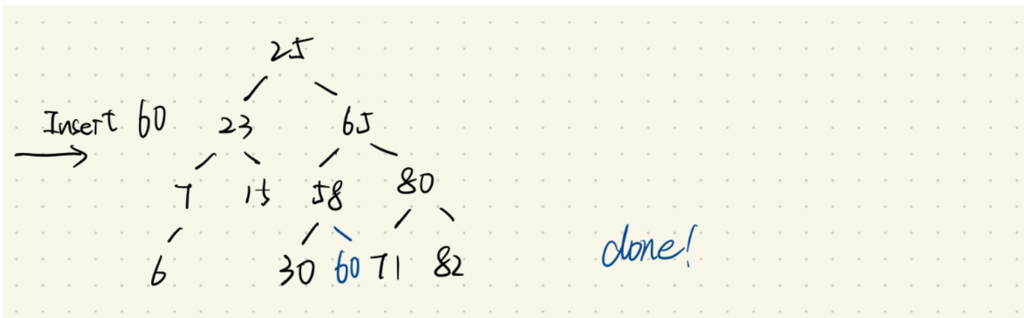
```
Boolean isBST(root, lower, upper){
   IF root==NULL:
      return true
   IF root.key<lower OR root.key>upper:
      return false
   bool left=isBST(root.getLeft(), lower, root.key)
   bool right=isBST(root.getRight(), root.key, upper)

   IF left==TRUE AND right==TRUE:
      return TRUE
   ELSE:
      return FALSE
}
```

4. (15') AVL Trees

Construct an AVL tree by inserting the input array {13, 7, 25, 58, 80, 15, 82, 6, 65, 71, 30, 60}. Draw the three trees after inserting the last three elements: 71, 30, and 60, respectively.

Solution:





5. (40’) B+ Trees

5.1. *(10’)* 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` *(16 bytes).*
- The size of a pointer is 8 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.

Solution:

Internal node: (M-1)*16+M*8=24M-16.

The size of one block is 2048 bytes, so 24M-16=2048, M=86.

L= the size of one block/the size of each employee record=2048/256=8.

5.2. *(30')* In this question, we use a B+ tree with M=L=4. The initial B+ tree is shown in Figure 1.

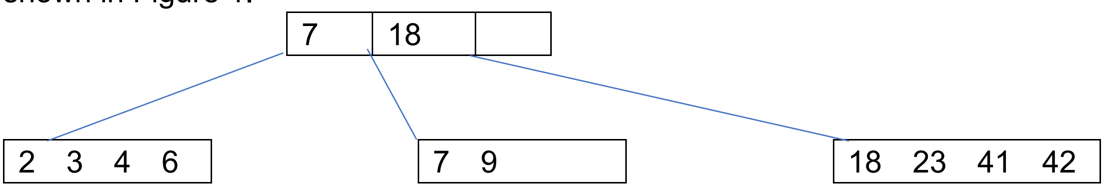


Figure 1: The initial B+ Tree

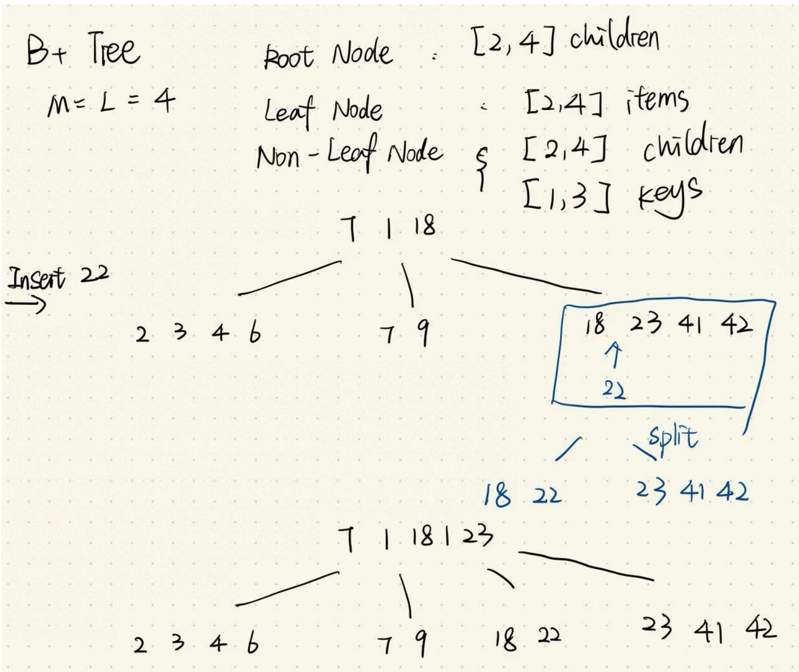
a) (15') Given an insertion sequence *{22, 12, 32}*, draw the three B+ trees after each insertion, respectively. Please start from the initial B+ tree shown in Figure 1.

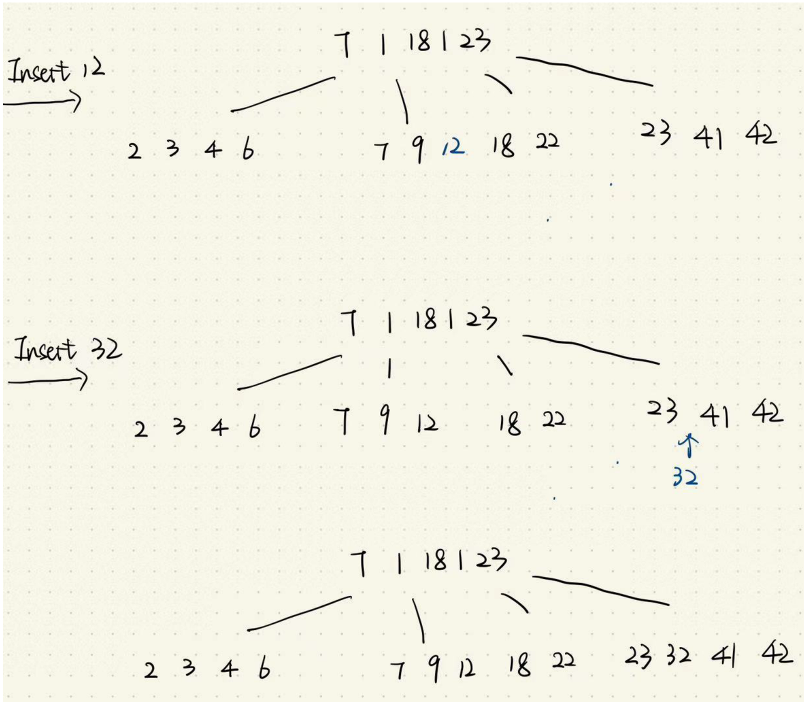
b) *(15')* Given a deletion sequence *{ 23, 9, 7}*, draw the three B+ trees after each deletion. Please start from the initial B+ tree shown in Figure 1.

Note: please strictly follow the lecture notes when you do the operations.

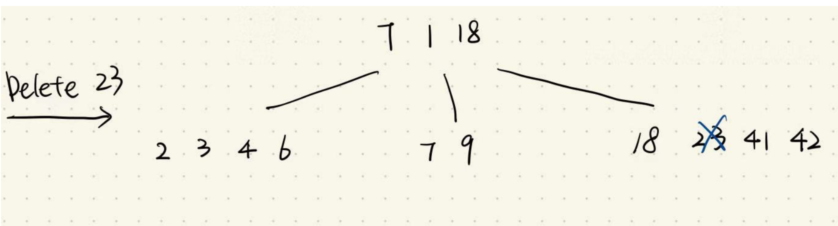
Solution:

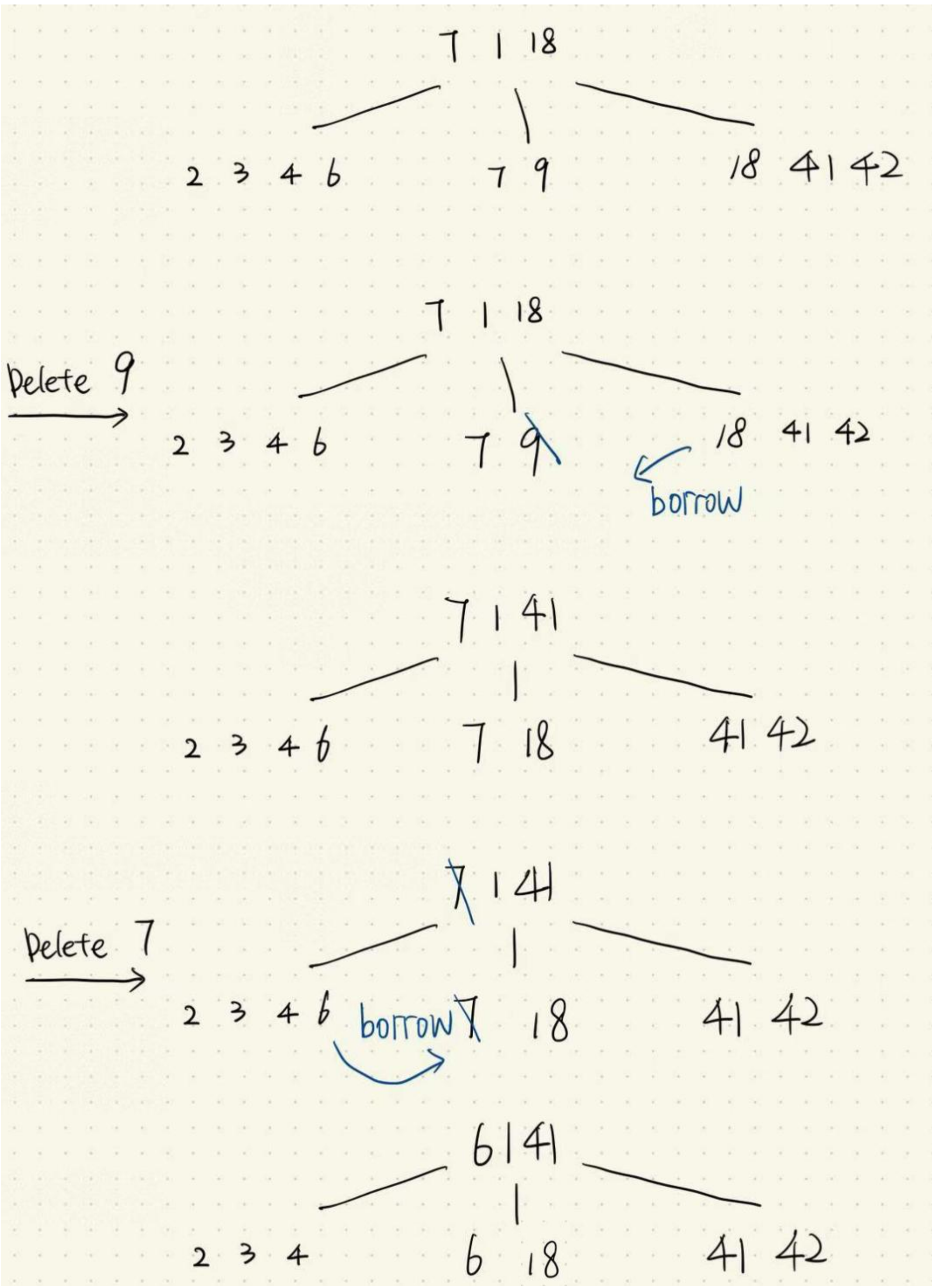
(a)


B+ Tree
M = L = 4
Root Node : [2, 4] children
Leaf Node : [2, 4] items
Non-Leaf Node { [2, 4] children
[1, 3] keys
Insert 22
7 | 18
2 3 4 6
7 9
18 23 41 42
22
split
18 22
23 41 42
7 | 18 | 23
2 3 4 6
7 9
18 22
23 41 42


Insert 12
7 | 18 | 23
2 3 4 6
7 9 12
18 22
23 41 42
Insert 32
7 | 18 | 23
2 3 4 6
7 9 12
18 22
23 41 42
32
7 | 18 | 23
2 3 4 6
7 9 12
18 22
23 32 41 42

(b)


Delete 23
7 | 18
2 3 4 6
7 9
18 ~~23~~ 41 42



6. (15’) Graph

Given a graph as shown in Figure 2. Imagine that node 6 is the source vertex:

a) *(2’)* Write out its adjacency matrix.

b) *(3’)* Write out its adjacency List.

c) *(10’)* Use the algorithm of “BFS + Path Finding” (as shown in Lecture 15, page 3) to work out the BFS tree step by step.

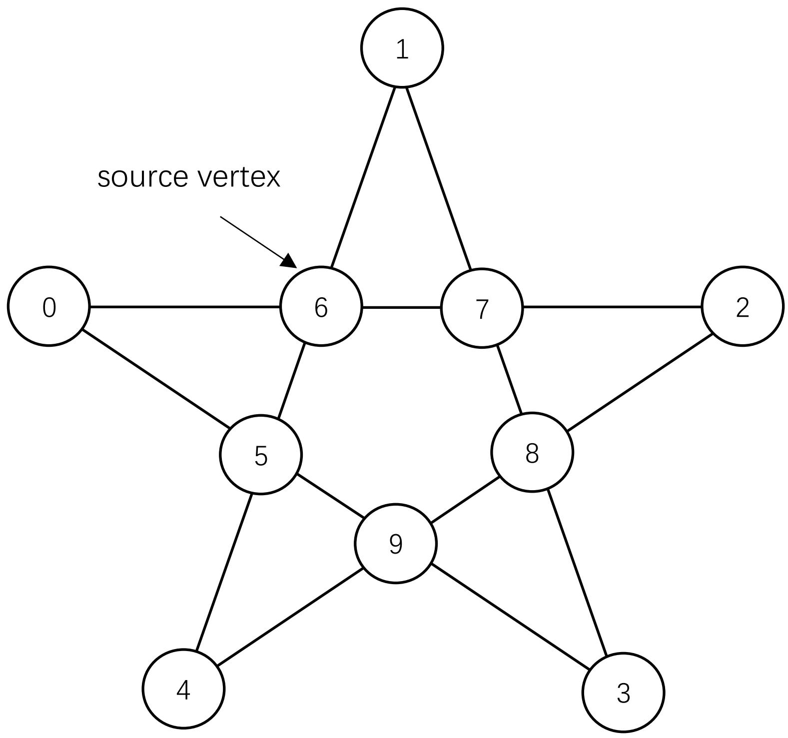


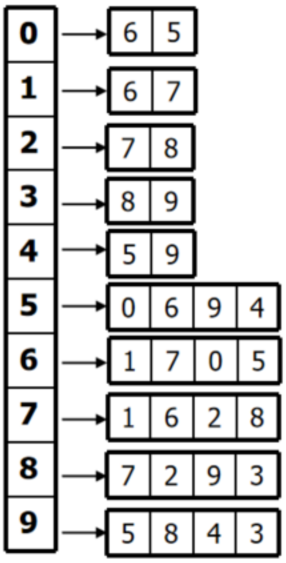
Figure 2 The initial graph

Solution:

(a) adjacency matrix:

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	0	0	1	1	0	0
2	0	0	0	0	0	0	0	1	1	0
3	0	0	0	0	0	0	0	0	1	1
4	0	0	0	0	0	1	0	0	0	1
5	1	0	0	0	1	0	1	0	0	1
6	1	1	0	0	0	1	0	1	0	0
7	0	1	1	0	0	0	1	0	1	0
8	0	0	1	1	0	0	0	1	0	1
9	0	0	0	1	1	1	0	0	1	0

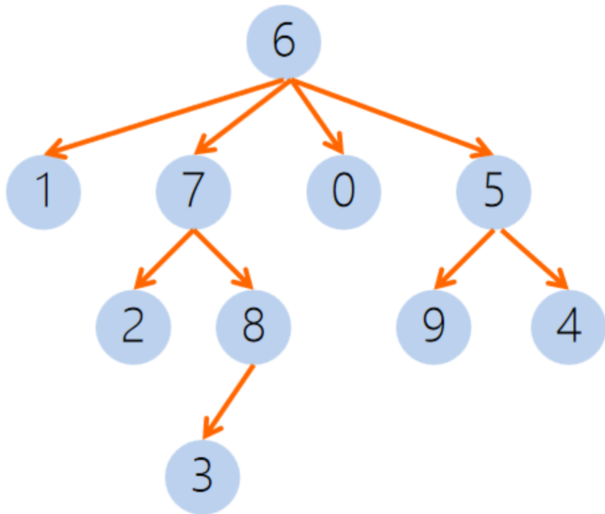
(b) adjacency list:



(c) BFS + Path Finding:

Visited Table (T/F)

		Pred
0	T	6
1	T	6
2	T	7
3	T	8
4	T	5
5	T	6
6	T	-
7	T	6
8	T	7
9	T	5



Step-by-step:

1. Initialize

Q={}

Visited Table (T/F)		Pred
0	F	-
1	F	-
2	F	-
3	F	-
4	F	-
5	F	-
6	F	-
7	F	-

8	F	-
9	F	-

2. Enqueue 6

   Q={6}

Visited Table (T/F)		Pred
0	F	-
1	F	-
2	F	-
3	F	-
4	F	-
5	F	-
6	T	-
7	F	-
8	F	-
9	F	-

3. Dequeue 6, Enqueue 1,7,0,5

   Q={1,7,0,5}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	F	-
3	F	-
4	F	-
5	T	6
6	T	-
7	T	6
8	F	-
9	F	-

4. Dequeue 1

   Q={7,0,5}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	F	-
3	F	-
4	F	-
5	T	6

6	T	-
7	T	6
8	F	-
9	F	-

5. Dequeue 7, Enqueue 2, 8

Q={0,5,2,8}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	T	7
3	F	-
4	F	-
5	T	6
6	T	-
7	T	6
8	T	7
9	F	-

6. Dequeue 0

Q={5,2,8}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	T	7
3	F	-
4	F	-
5	T	6
6	T	-
7	T	6
8	T	7
9	F	-

7. Dequeue 5, Enqueue 9,4

Q={2,8,9,4}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	T	7
3	F	-
4	T	5
5	T	6
6	T	-

7	T	6
8	T	7
9	T	5

8. Dequeue 2
   Q={8,9,4}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	T	7
3	F	-
4	T	5
5	T	6
6	T	-
7	T	6
8	T	7
9	T	5

9. Dequeue 8,Enqueue 3
   Q={9,4,3}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	T	7
3	T	8
4	T	5
5	T	6
6	T	-
7	T	6
8	T	7
9	T	5

10. Dequeue 9
   Q={4,3}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	T	7
3	T	8
4	T	5
5	T	6
6	T	-
7	T	6

8	T	7
9	T	5

11. Dequeue 4
    Q={3}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	T	7
3	T	8
4	T	5
5	T	6
6	T	-
7	T	6
8	T	7
9	T	5

12. Dequeue 3
    Q={}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	T	7
3	T	8
4	T	5
5	T	6
6	T	-
7	T	6
8	T	7
9	T	5

Alternative solution:

1. Initialize
   Q={}

Visited Table (T/F)		Pred
0	F	-
1	F	-
2	F	-
3	F	-
4	F	-
5	F	-
6	F	-

7	F	-
8	F	-
9	F	-

2. Enqueue 6
   Q={6}

Visited Table (T/F)		Pred
0	F	-
1	F	-
2	F	-
3	F	-
4	F	-
5	F	-
6	T	-
7	F	-
8	F	-
9	F	-

3. Dequeue 6, Enqueue 0, 1, 5, 7
   Q={0,1,5,7}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	F	-
3	F	-
4	F	-
5	T	6
6	T	-
7	T	6
8	F	-
9	F	-

4. Dequeue 0
   Q={1,5,7}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	F	-
3	F	-
4	F	-

5	T	6
6	T	-
7	T	6
8	F	-
9	F	-

5. Dequeue 1
   Q={5,7}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	F	-
3	F	-
4	F	-
5	T	6
6	T	-
7	T	6
8	F	-
9	F	-

6. Dequeue 5, Enqueue 4, 9
   Q={7,4,9}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	F	-
3	F	-
4	T	5
5	T	6
6	T	-
7	T	6
8	F	-
9	T	5

7. Dequeue 7, Enqueue 2,8
   Q={4,9,2,8}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	T	7
3	F	-
4	T	5
5	T	6

6	T	-
7	T	6
8	T	7
9	T	5

8. Dequeue 4
   Q={9,2,8}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	T	7
3	F	-
4	T	5
5	T	6
6	T	-
7	T	6
8	T	7
9	T	5

9. Dequeue 9, Enqueue 3
   Q={2,8,3}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	T	7
3	T	9
4	T	5
5	T	6
6	T	-
7	T	6
8	T	7
9	T	5

10. Dequeue 2
   Q={8,3}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	T	7
3	T	9
4	T	5
5	T	6
6	T	-

7	T	6
8	T	7
9	T	5

11. Dequeue 8
    Q={3}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	T	7
3	T	9
4	T	5
5	T	6
6	T	-
7	T	6
8	T	7
9	T	5

12. Dequeue 3
    Q={}

Visited Table (T/F)		Pred
0	T	6
1	T	6
2	T	7
3	T	9
4	T	5
5	T	6
6	T	-
7	T	6
8	T	7
9	T	5