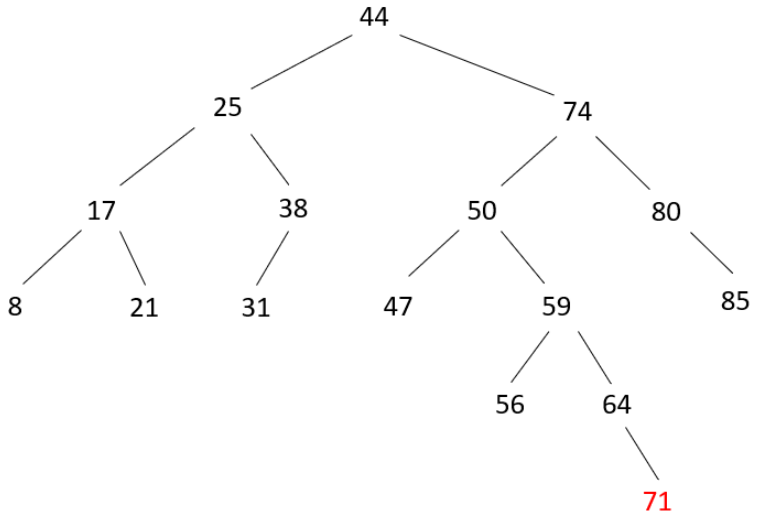
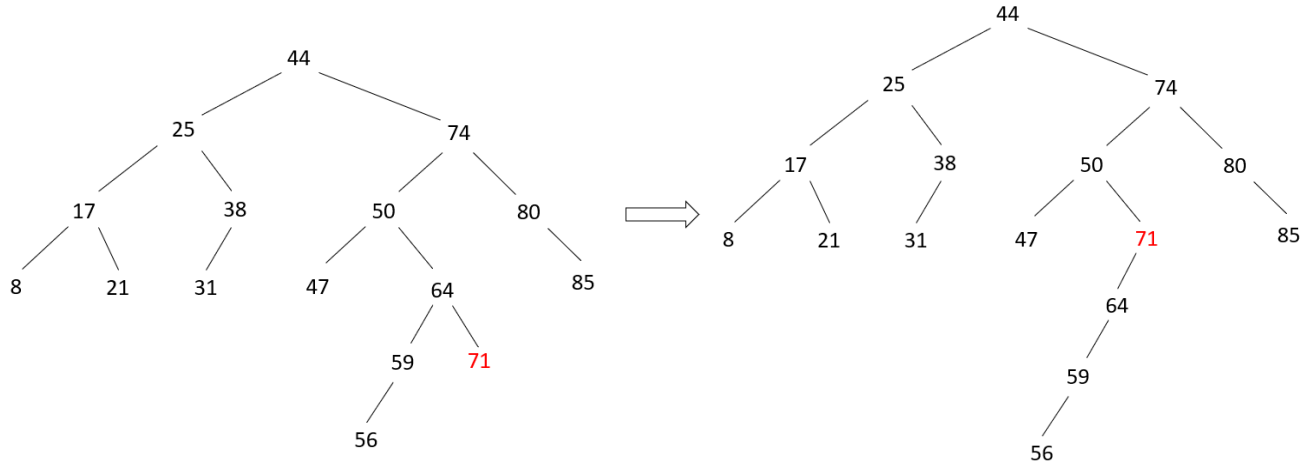
Com S 228
Spring 2019
Final Exam Sample Solution

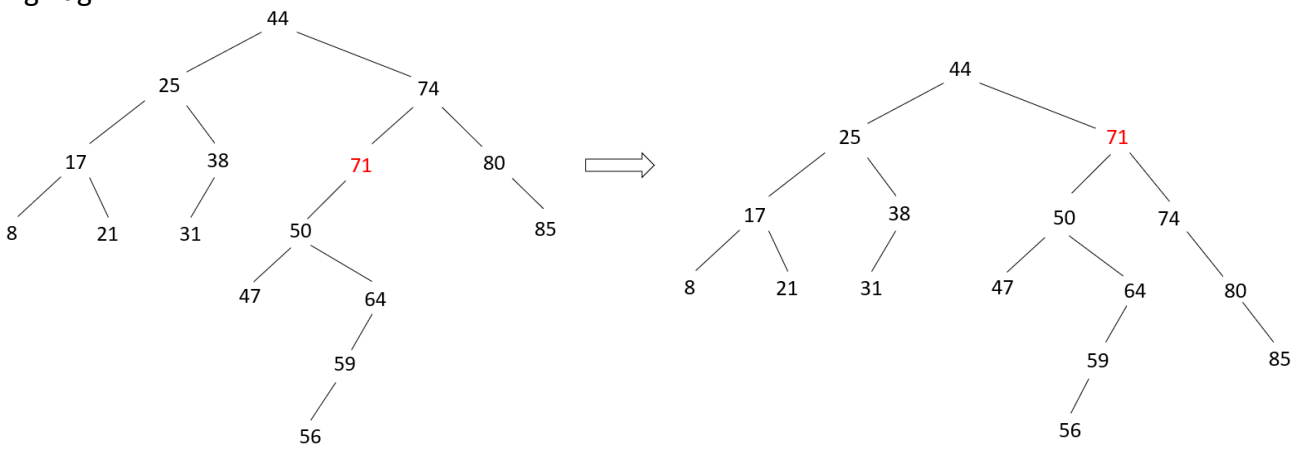
1 a) 7
b) 8
c) 14
d) 4
e) 2
f) 3
g) 44
h) 64
i) 44, 25, 17, 8, 21, 38, 31, 74, 50, 47, 59, 56, 64, 80, 85.
j) BST insert:



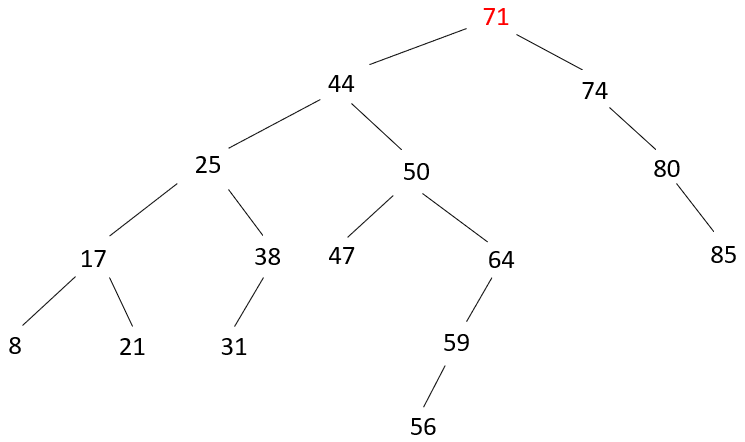

Zig-zig:



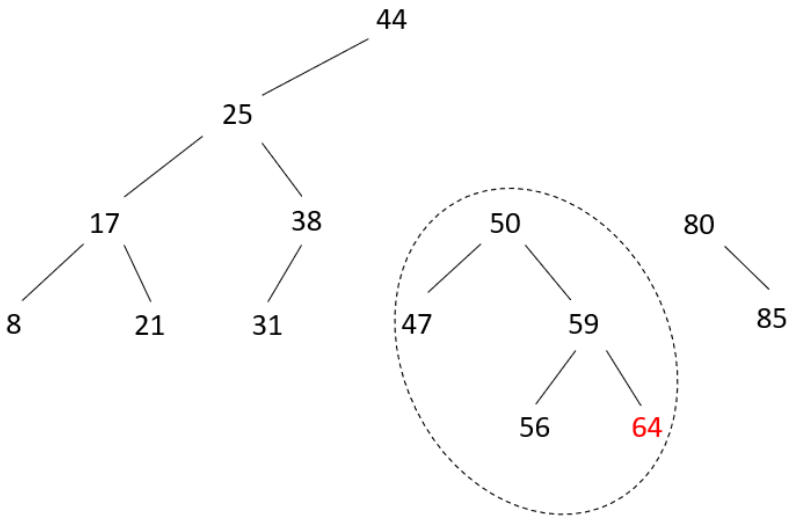
Zig-zag:



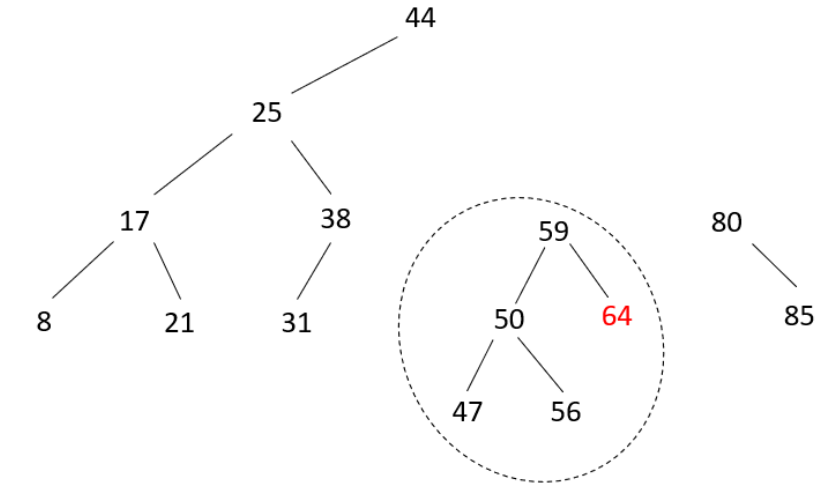

Zig:

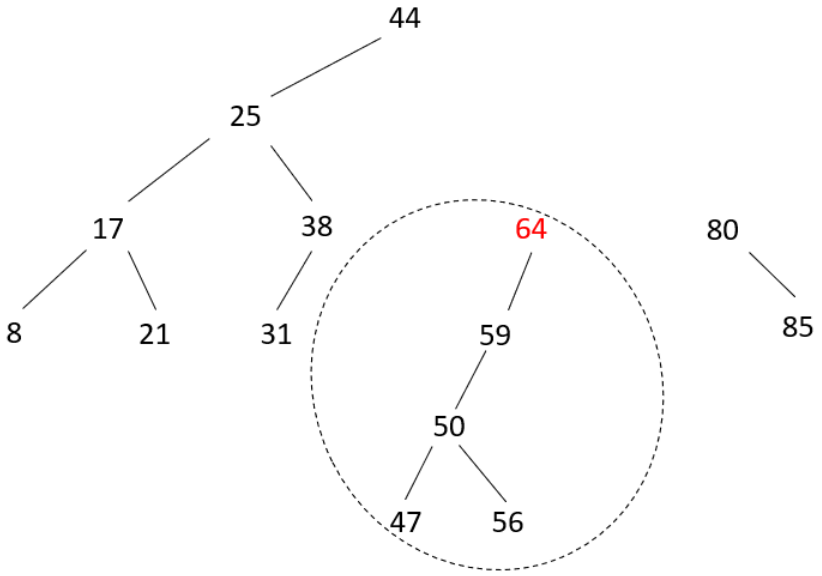



k) Remove 74:

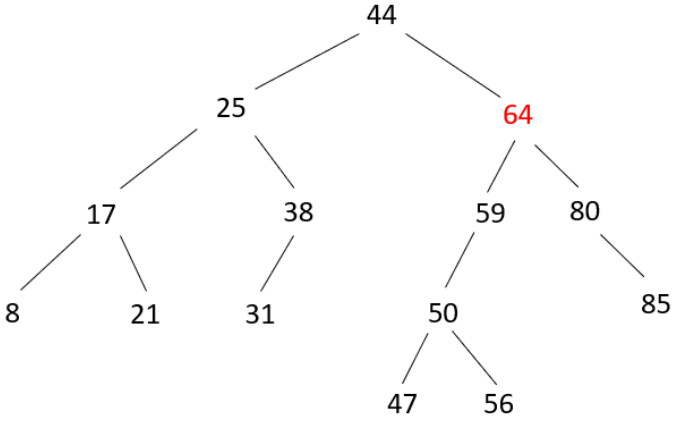


Join the subtrees of 74, starting with accessing the rightmost node in the left subtree:

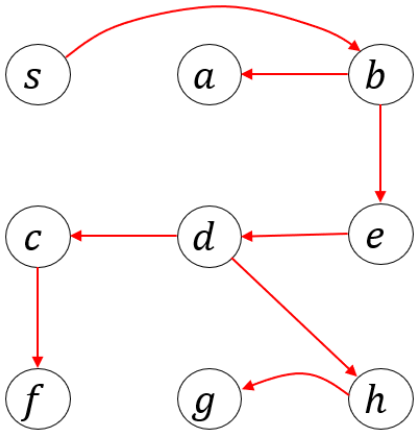


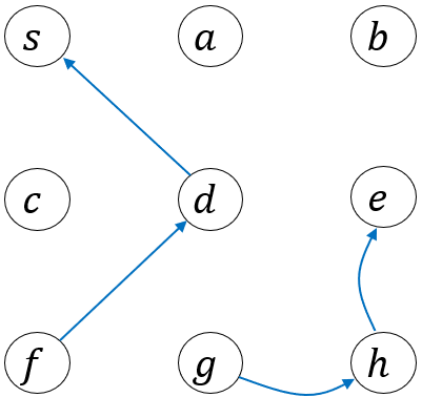

Replace 74 with the join.  No need to splay at the parent 44 of 64 since it is the root.



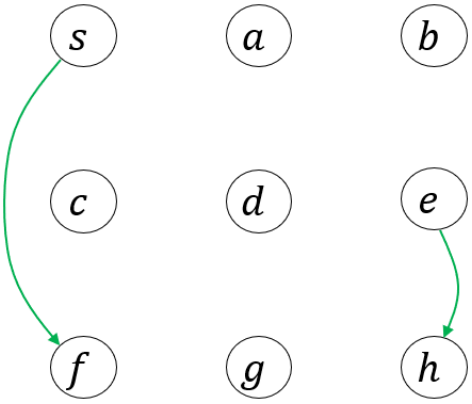
2a) 3
b) 4
c) No
d) Yes
e) DFS



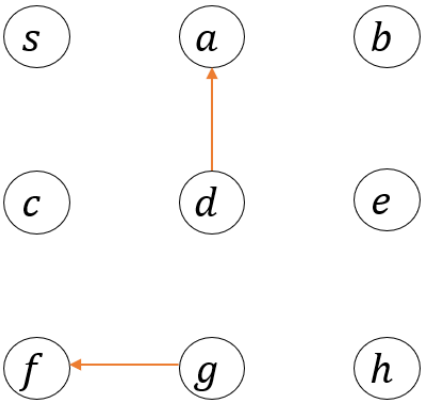

Tree edges




Back edges

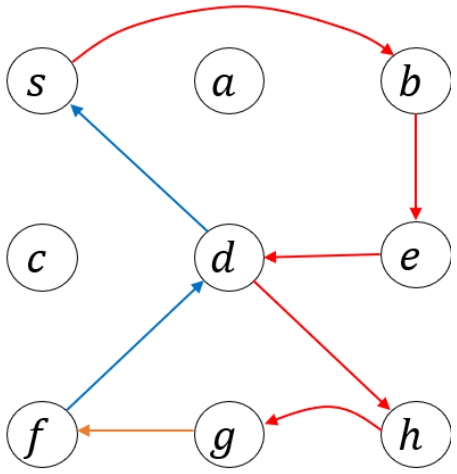



Forward edges

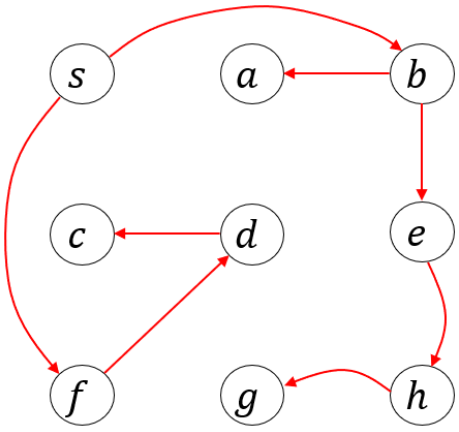



Cross edges

f) Either of the two simple cycles of length 4 suffices: $< s, b, e, d, s >$ and $< d, h, g, f, d >$.




g) BFS tree

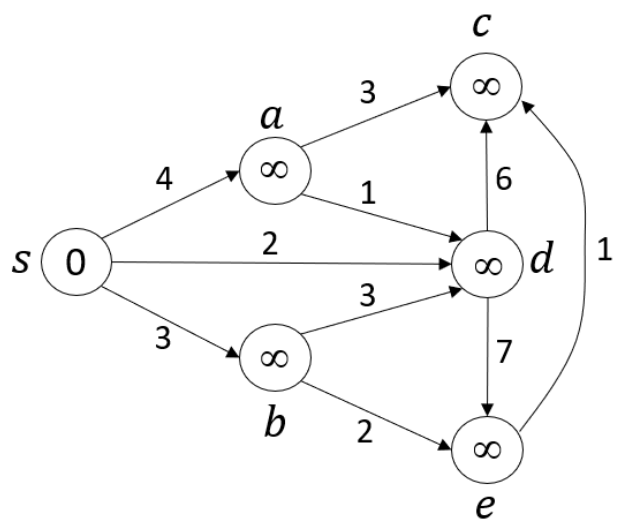


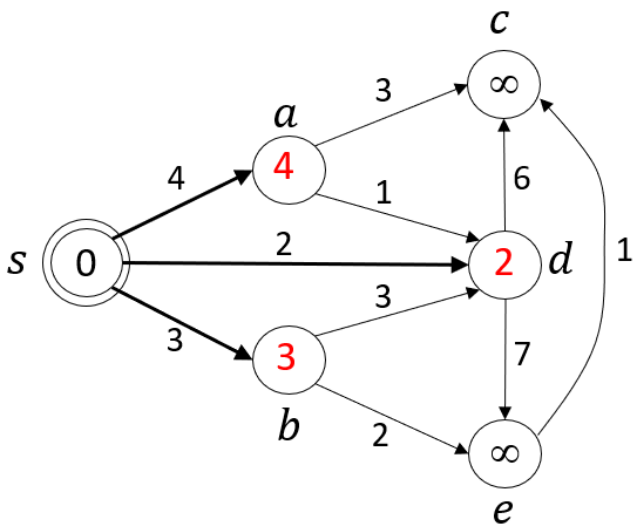

3a) There are only two correct topological sorts, either of which is sufficient as an answer.

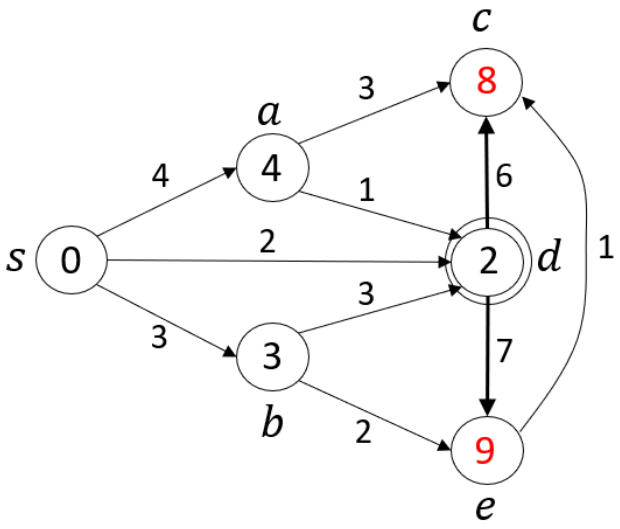
| | | | | | |
|---|---|---|---|---|---|
| $s$ | $a$ | $b$ | $d$ | $e$ | $c$ |

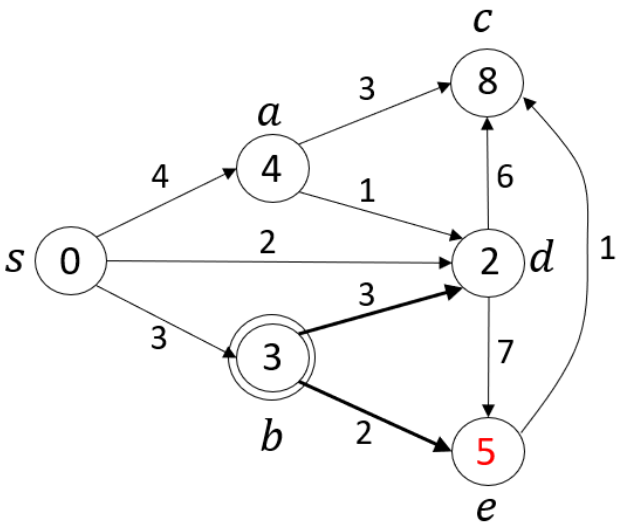
| | | | | | |
|---|---|---|---|---|---|
| $s$ | $b$ | $a$ | $d$ | $e$ | $c$ |

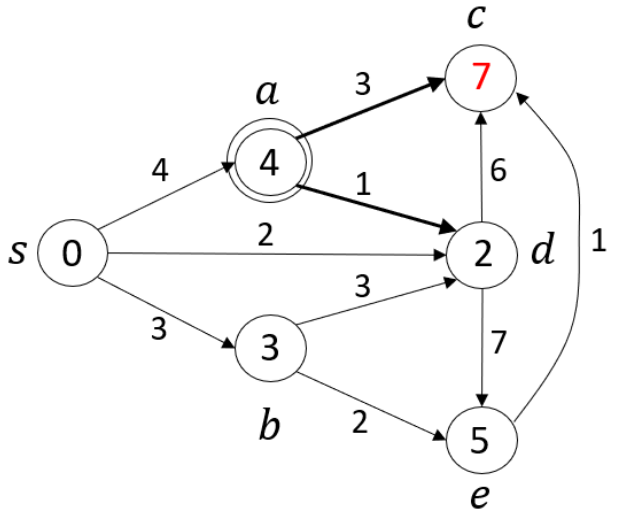
b)

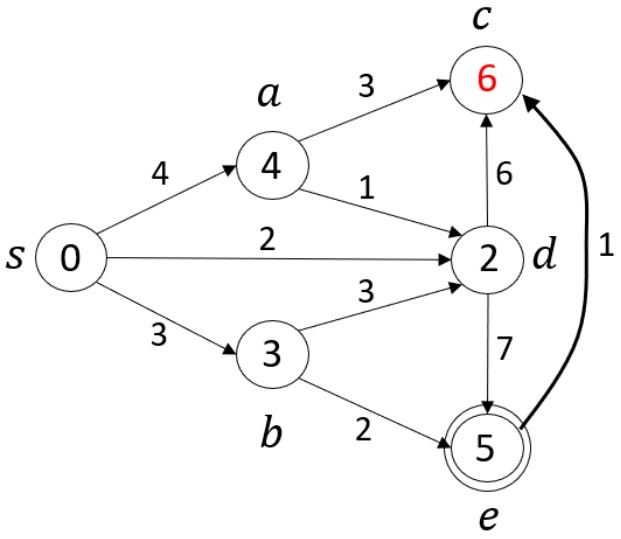

c
a
∞
3
4
1
6
s
0
2
∞
d
1
3
3
∞
7
b
2
∞
e

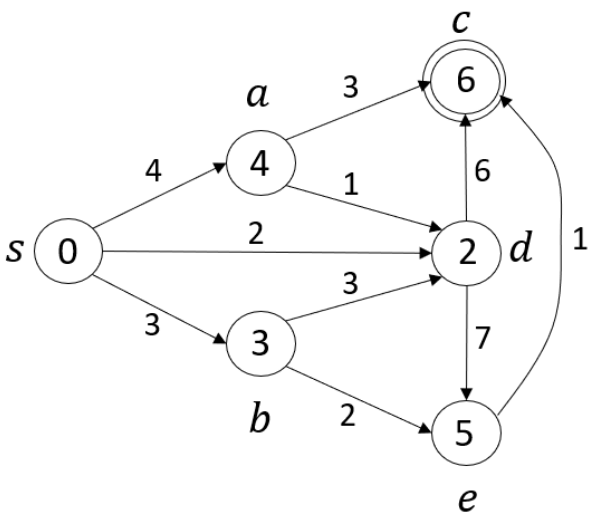

c
a
∞
3
4
4
1
6
s
0
2
2
d
1
3
3
3
7
b
2
∞
e


c
a
8
3
4
4
1
6
s
0
2
2
d
1
3
3
3
7
b
2
9
e


c
a
8
3
4
4
1
6
s
0
2
2
d
1
3
3
3
7
b
2
5
e


c
a
7
3
4
4
1
6
s
0
2
2
d
1
3
3
3
7
b
2
5
e


c
a
6
3
4
4
1
6
s
0
2
2
d
1
3
3
3
7
b
2
5
e


c
6
a
3
4
4
1
6
s
0
2
2
d
1
3
3
7
3
b
2
5
e

4. Row 3 with its entries underlined is the first line that is a heap.

| Row | | | | Array | | | |
|---|---|---|---|---|---|---|---|
| 0 | 5 | 2 | 4 | 0 | 6 | 3 | 1 |
| 1 | 5 | 6 | 4 | 0 | 2 | 3 | 1 |
| 2 | 6 | 5 | 4 | 0 | 2 | 3 | 1 |
| 3 | 1 | 5 | 4 | 0 | 2 | 3 | 6 |
| 4 | 5 | 1 | 4 | 0 | 2 | 3 | 6 |
| 5 | 5 | 2 | 4 | 0 | 1 | 3 | 6 |
| 6 | 3 | 2 | 4 | 0 | 1 | 5 | 6 |
| 7 | 4 | 2 | 3 | 0 | 1 | 5 | 6 |
| 8 | 1 | 2 | 3 | 0 | 4 | 5 | 6 |
| 9 | 3 | 2 | 1 | 0 | 4 | 5 | 6 |
| 10 | 0 | 2 | 1 | 3 | 4 | 5 | 6 |
| 11 | 2 | 0 | 1 | 3 | 4 | 5 | 6 |
| 12 | 1 | 0 | 2 | 3 | 4 | 5 | 6 |
| 13 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 14 | | | | | | | |
| 15 | | | | | | | |

5a) $O(n)$
b) $O(n)$
c) $O(n)$
d) $O(n \log n)$
e) $O(n)$
f) $O(n \log n)$
g) $O(n^2)$
h) $O(\log n)$
i) $O(V + E)$
j) $O(V + E)$

6.

```
public E removeRoot() throws IllegalStateException
{
	E data;

	if (size == 0)
		throw new IllegalStateException("No root removal on an empty tree");

	data = root.data;


	// BST has only one node.
	if (size == 1)
	{
		// insert code below (1 pt)
		root = null;
	}

	// BST has two or more nodes but no left subtree.
	else if (root.left == null)
	{
		// insert code below (3 pts)
		root = root.right;   // root must have a right child since size > 1.
		root.parent = null;
	}

	// BST has two or more nodes and a left subtree.
	// find the predecessor of the root and promote it to the new root.
	else
	{
		Node cur = root.left;
		Node prnt = root;

		// find the predecessor of the root.
		while (cur.right != null)
		{
			// insert code below (2 pts)
			prnt = cur;
			cur = cur.right;
		}

		// left child of root has a right subtree.
		if (prnt != root)
		{
			// update links related to the predecessor's subtree(s).
			// insert code below (4 pts)
			prnt.right = cur.left;
			if (cur.left != null)
				cur.left.parent = prnt;

			// set up the new root's left subtree.
			// insert code below (3 pts)
```

```
                cur.left = root.left;
                cur.left.parent = cur; // or root.left.parent = cur;
            }

            // set up the new root's right subtree.
            // insert code below (4 pts)
            cur.right = root.right;
            if (cur.right != null)  // or root.right != null
                cur.right.parent = cur;  // or root.right.parent = cur;


            // set up the new root.
            // insert code below (2 pts)
            cur.parent = null;
            root = cur;
        }

        // other updates if needed
        // insert code below (1 pt)
        size--;

        return data;
    }
```