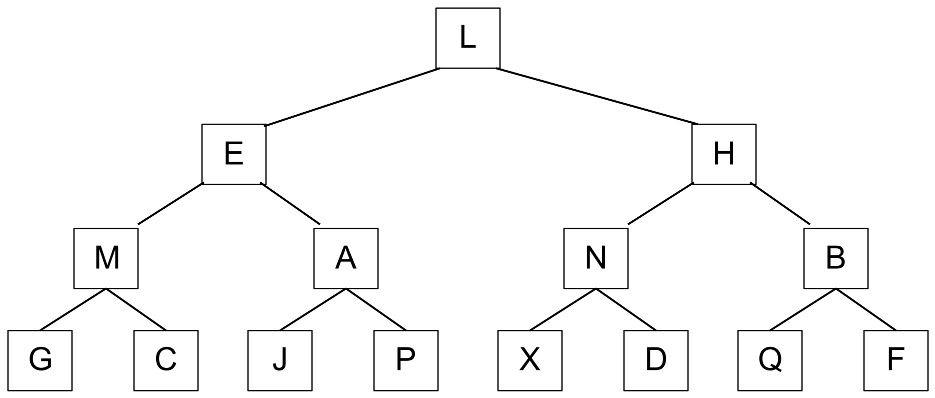
CS 101 Exam 3 **Summer 2016** **Solution**

1. For each algorithm below, write each indicated running time as the best Big O function of n. **[33 points]**

	Worst case	Average case	Best case
Insert a key into a binary search tree	**O(n)**	**O(lg n)**	**O(1)**
Find a given key in a binary search tree	**O(n)**	**O(lg n)**	**O(1)**
Remove a key from a binary search tree	**O(n)**	**O(lg n)**	**O(1)**
Insert a key into a min-heap	**O(lg n)**	**O(lg n)**	**O(1)**
Find the min key in a min-heap	**O(1)**	**O(1)**	**O(1)**
Find a given key in a min-heap	**O(n)**	**O(n)**	**O(1)**
Remove the min key from a min-heap	**O(lg n)**	**O(lg n)**	**O(1)**
Heap sort	**O(n lg n)**	**O(n lg n)**	**O(n)**
Counting sort, with items in range 0…m	**O(n+m)**	**O(n+m)**	**O(n+m)**
Bucket sort, with items in range 0…m	**O(n+m)**	**O(n+m)**	**O(n+m)**
Radix sort, with items in range 0…r^d–1	**O(d(n+r))**	**O(d(n+r))**	**O(d(n+r))**

2. Write each indicated traversal of the given tree. **[12 points]**



a. Preorder traversal: **L E M G C A J P H N X D B Q F**

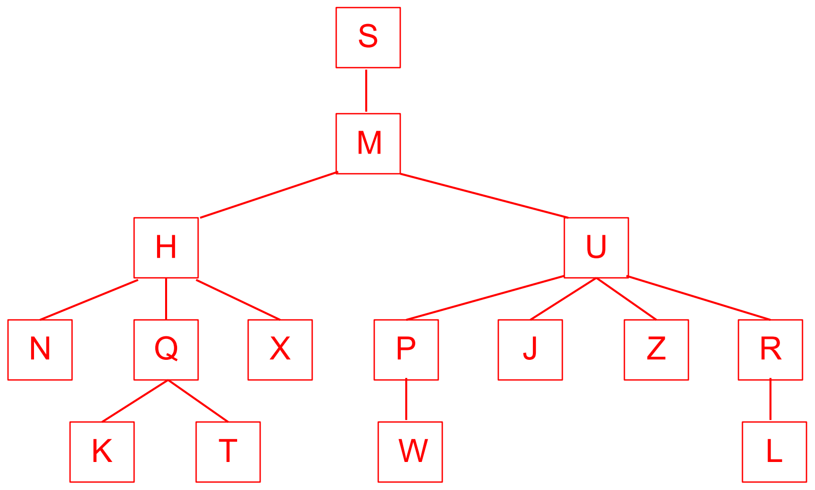
b. Postorder traversal: **G C M J P A E X D N Q F B H L**

c. Inorder traversal: **G M C E J A P L X N D H Q B F**

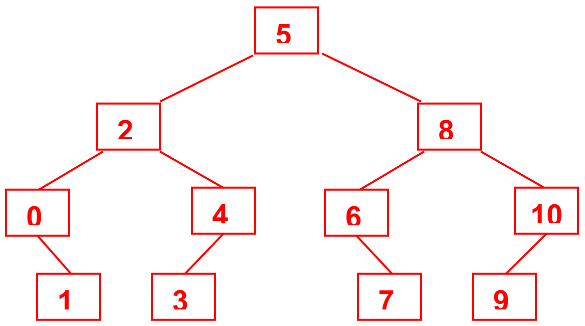
d. Level-order traversal: **L E H M A N B G C J P X D Q F**

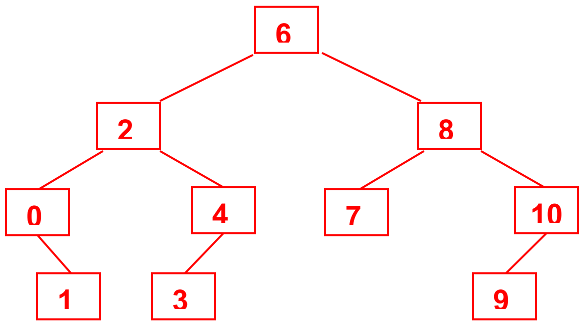
3. Draw the general (non-binary) tree that yields the pair of traversals below. **[4 points]**

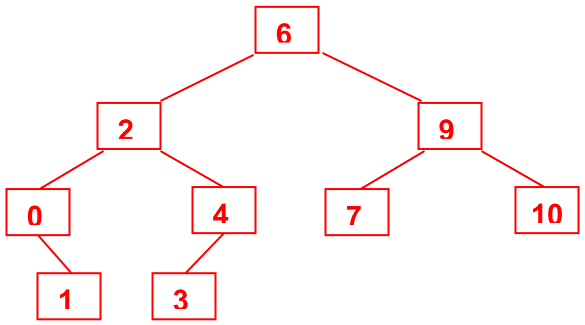
Postorder:	N K T Q X H W P J Z L R U M S
Level-order:	S M H U N Q X P J Z R K T W L

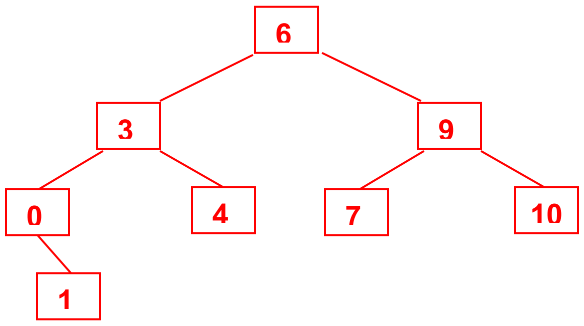


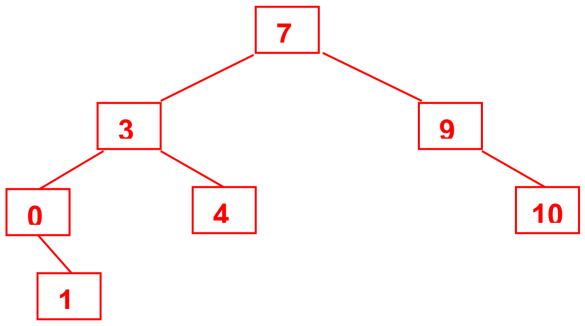
4. Given an initially empty binary search tree. First insert these keys in the order shown: 5, 8, 2, 6, 0, 10, 4, 3, 7, 1, 9. Draw the binary search tree after the final insert operation. Next remove these keys in the order shown: 5, 8, 2, 6, 9, 3. Where appropriate, use the successors (not the predecessors). Redraw the binary search tree at the end of each remove operation. Make sure that each tree you draw is a correctly formed binary search tree. **[15 points]**

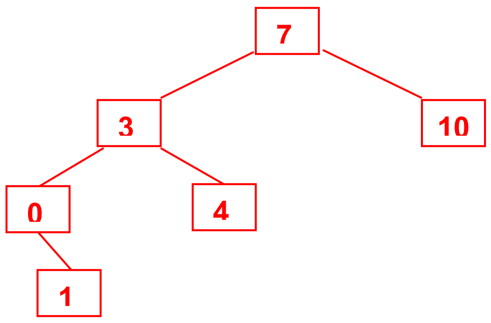


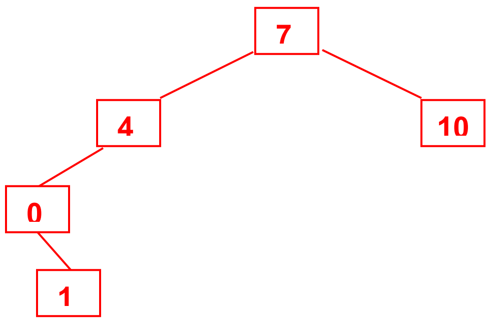




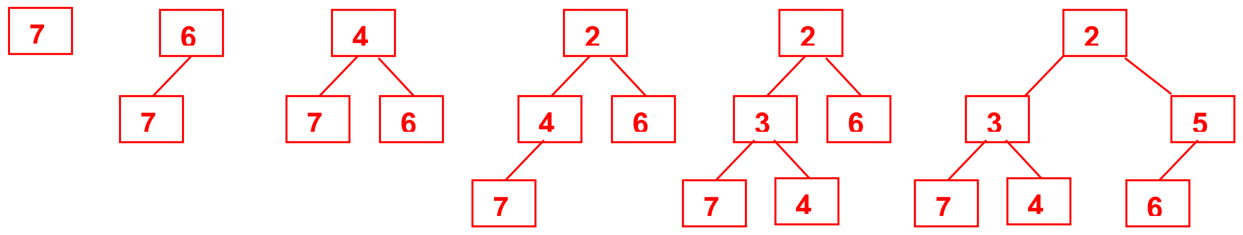


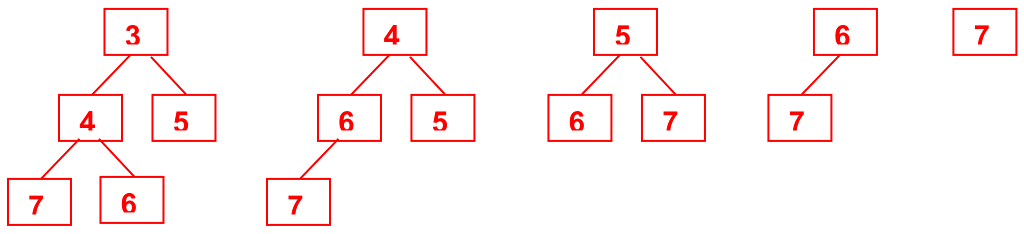




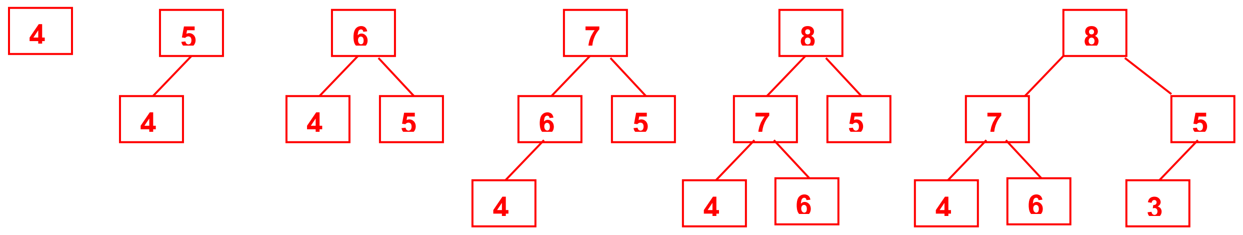


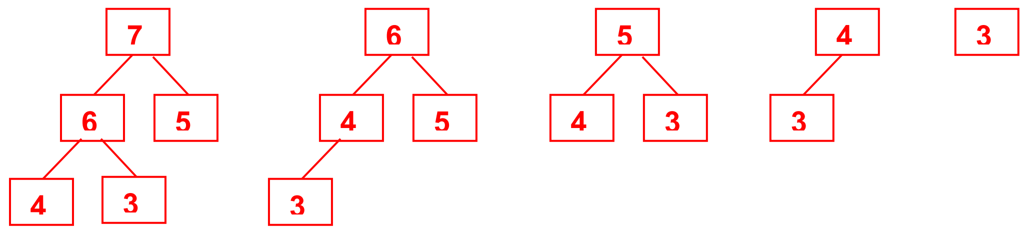
5. Given an initially empty min-ordered heap. First insert these keys in the order shown: 7, 6, 4, 2, 3, 5. Draw the min-heap after each insert operation. Next repeatedly perform a removeMin operation until the min-heap becomes empty. Redraw the min-heap at the end of each removeMin operation. Make sure that each tree you draw is a correctly formed min-heap. **[10 points]**





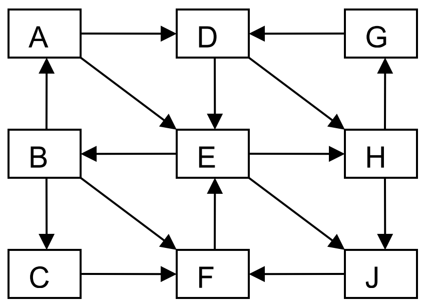
6. Given an initially empty max-ordered heap. First insert these keys in the order shown: 4, 5, 6, 7, 8, 3. Draw the max-heap after each insert operation. Next repeatedly perform a removeMax operation until the max-heap becomes empty. Redraw the max-heap at the end of each removeMax operation. Make sure that each tree you draw is a correctly formed max-heap. **[10 points]**





7. Answer the questions below regarding the given directed graph. **[20 points]**

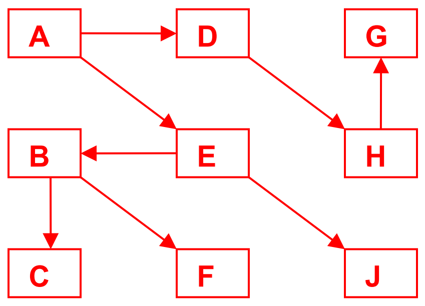
Draw the linked adjacency lists representation (only the outgoing neighbor lists), such that each neighbor list is arranged alphabetically. Also show the adjacency matrix representation.

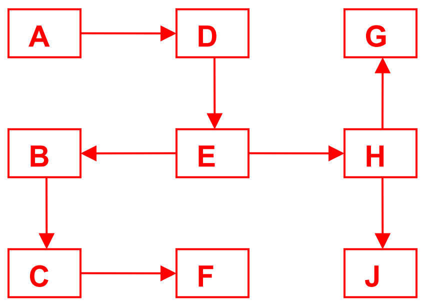


	out	
A		**→D→E**
B		**→A→C→F**
C		**→F**
D		**→E→H**
E		**→B→H→J**
F		**→E**
G		**→D**
H		**→G→J**
J		**→F**

	A	B	C	D	E	F	G	H	J
A	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**
B	**1**	**0**	**1**	**0**	**0**	**1**	**0**	**0**	**0**
C	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
D	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**
E	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**1**
F	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
G	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
H	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**
J	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**

Starting at node A, draw the *breadth*-first search tree and write the *breadth*-first search order. Next, again starting at node A, draw the *depth*-first search tree and write the *depth*-first search order.

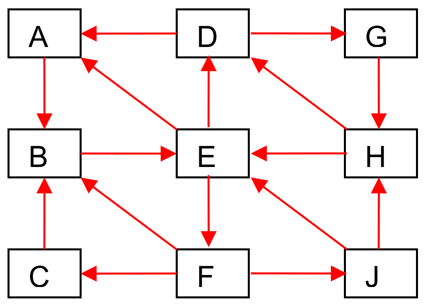




BFS: A D E H B J G C F

DFS: A D E B C F H G J

Draw the reverse of the above directed graph. Also draw its linked adjacency lists representation (only the outgoing neighbor lists), such that each neighbor list is arranged alphabetically. Also show its adjacency matrix representation.



	out	
A		**→B**
B		**→E**
C		**→B**
D		**→A→G**
E		**→A→D→F**
F		**→B→C→J**
G		**→H**
H		**→D→E**
J		**→E→H**

	A	B	C	D	E	F	G	H	J
A	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
B	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
C	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
D	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**
E	**1**	**0**	**0**	**1**	**0**	**1**	**0**	**0**	**0**
F	**0**	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**1**
G	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
H	**0**	**0**	**0**	**1**	**1**	**0**	**0**	**0**	**0**
J	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**