1. Trace the execution of the following sorting algorithms

a. Selection Sort	**b. Heap Sort**
c. Buble Sort	**d. Merge Sort**
e. Insertion Sort	**f. Radix Sort**

Using the numbers
129, 778, 111, 352, 233, 710, 783, 812, 165, 10

2. Trace the Quicksort's partitioning algorithm as it partitions the following array. Use the first item as the pivot.

90	**100**	**200**	**60**	**40**	**50**	**80**	**70**	**120**	**88**

3. Which of the above sorting algorithms in question 1 are stable?

4. An algorithm works by dividing the search space in half recursively at each stage. The most accurate estimate of the running time is?

5. What is the *average* "Big-O" complexity of the following operations? Assume a *good* implementation. Use $O(1)$ to denote constant time.

a. **Preorder traversal of a binary tree**
b. **Insertion sort**
c. **Bucket sort**
d. **Quick sort**
e. **Find in a hash table**

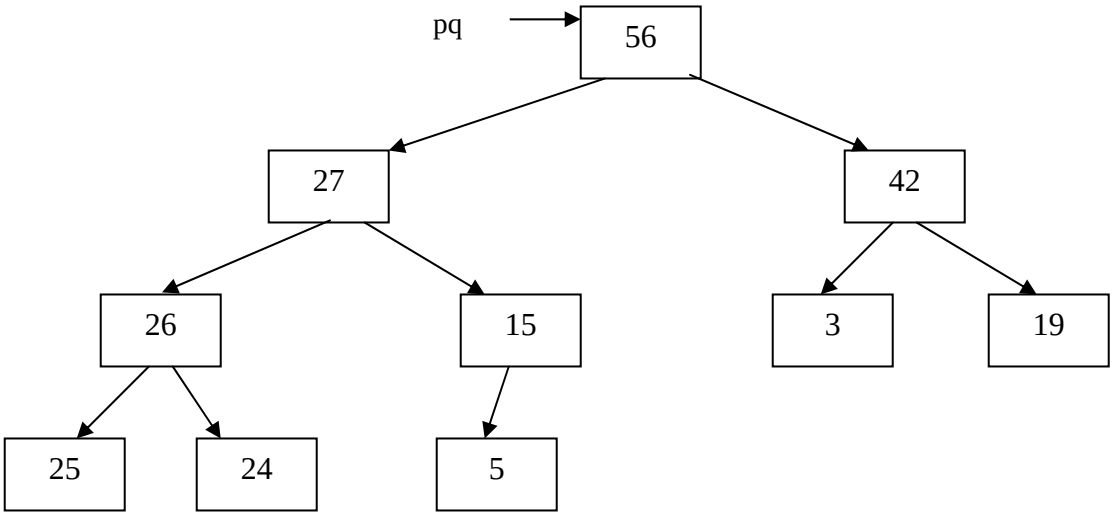
6. True or false

a. $n^{\alpha} = O(n^{\beta})$ **if** $0 < \beta \leq \alpha$
b. $\log_b x^y = x \log_b y$
c. $\log_b(xy) = \log_b(x) + \log_b(y)$
d. $5n^2 - 9n = \Omega(n^2)$
e. $6n^3 = O(n^2)$
f. $f(n) = \Theta(g(n))$ **if and only if** $g(n) = \Theta(f(n))$
g. $2^n = o(\pi^n)$

7. Suppose you have an algorithm in which you reduce the search space by two-thirds in each step (i.e. in each step, you eliminate all but 1/3 of the remaining possible answers). What is the most likely "Big-O" efficiency of this algorithm?

a. 3^n
b. $n^{1/3}$
c. $\log n$
d. $\log_3 n$
e. n^3

8. A priority queue is implemented as a heap:



Show how the heap would look like after

a. pq.Enqeue(28);
b. pq.Enqueue(2);
c. pq.Enqueue(40);
d. pq.Dequeue(root)

9. A priority queue is implemented as a heap stored in an array. The precondition states that this priority queue cannot contain duplicate elements. Currently, the priority queue holds 10 elements, as shown below. What values might be stored in array positions 7-9 so that the properties of a heap are satisfied.

[0]	Z
[1]	F
[2]	J
[3]	E
[4]	B
[5]	G
[6]	H
[7]	?
[8]	?
[9]	?

10. Convert the following binary tree to heap using reheapdown algorithm, or explain why it is not possible.

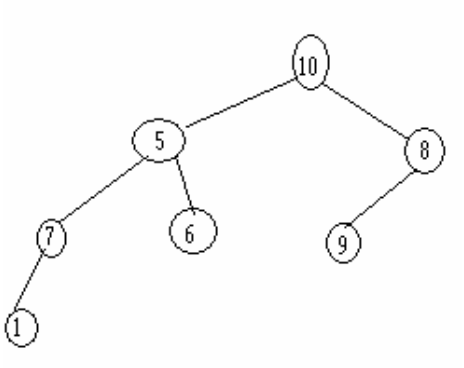
10. If an O (N^2) algorithm (such as Bubble Sort) takes 3.1 milliseconds to run on an array of 200 elements, how long would you expect it to take to run on a similar array of:

a. 400 elements?

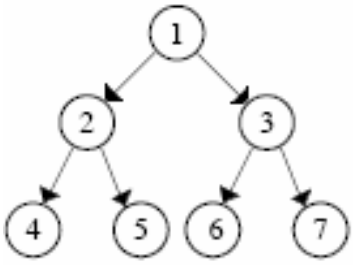
b. 40, 000 elements?

11. Convert the following binary tree to heap using reheapdown algorithm, or explain why it is not possible.

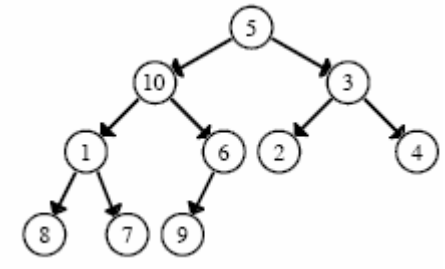
a



b



c



Questions for Hashing:

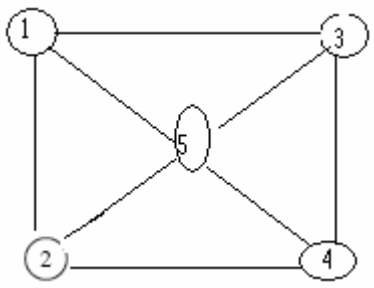
For questions 1-4 use the following values:

66 47 87 90 126 140 145 153 177 285 393 395 467 566 620 735

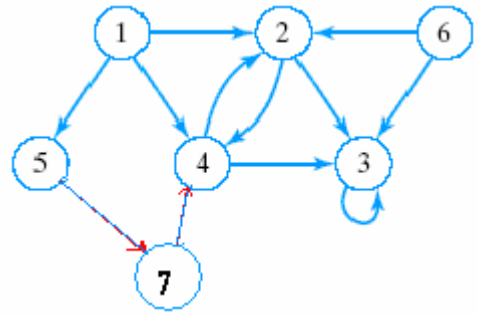
1. Store the values into a hash table with 20 positions, using the division method of hashing and the linear probing method of resolving collisions.
2. Store the values into a hash table with 20 positions, using rehashing as the method of collision resolution. Use *key % tableSize* as the hash function, and (*key + 3) % tableSize* as the rehash function.
3. Store the values into a hash table with ten buckets, each containing three slots. If a bucket is full, use the next (sequential) bucket that contains free slot.
4. Store the values into a hash table that uses the hash function *key % 10* to determine into which of ten chains to put the value.
5. Why primality is important in determining a hash function?
6. What is linear probing?
7. What is quadratic probing?
8. What is chain hashing?

Questions related to Graphs:

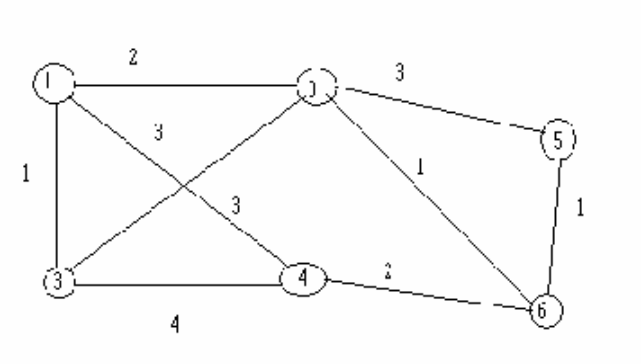
1. **Give the adjacency matrix and adjacency link list representation of the following graph.**



2. **Travers the following graph with DFS and BFS, staring from node 1.**



3. **Find the minimum spanning tree of the graph given below using**
    a. **Prim's algorithm**
    b. **Kruskal's algorithm**



4. **Find the shortest path of the graph given above using Dijkastra's algorithm.**