CSCI 2110
Data Structures and Algorithms
Practice Question Bank for the Final Exam
SOLUTIONS

1. Derive the big O complexities for each of the following running times (Show steps).

   a. $0.25\ n^3 + 1000\ n^2 + n + 100000$

   $\rightarrow 0.25n^3 + 1000n^2 + n + 1$
   $\rightarrow 0.25n^3 \rightarrow n^3 \rightarrow O(n^3)$

   b. $2n(15 + \log n) + 4\log n(3n + 5n^2)$

   $= 30n + 2n\log n + 12\log n + 20n^2\log n$
   $\rightarrow 20\ n^2\log n \rightarrow O(n^2\log n)$

   c. $(1 + 2 + 3 + \ldots + n) * (1+2+3+ \ldots + n)$

   $= \frac{n(n+1)}{2} * \frac{n(n+1)}{2} = \frac{n^2(n+1)^2}{4} = \frac{n^2(n^2+2n+1)}{4}$
   $= \frac{n^4}{4} + \frac{2n^3}{4} + \frac{n^2}{4} \rightarrow O(n^4)$

   d. $(1+2+3+\ldots n) / (1+3+5+\ldots+(2n-1))$

   $= \frac{n(n+1)/2}{n^2} = \frac{n^2+n}{2n^2} = \frac{1}{2} + \frac{1}{2n} \rightarrow O(1)$

   e. $\log n^n + n \log n^{10} + n \log n^2$

   $= n\log n + 10n\log n + 2n\log n$
   $= 13n\log n \rightarrow O(n\log n)$

2. Arrange the following growth functions in *increasing order* of growth rates. (Reduce the growth functions wherever necessary).

   a. The growth functions that you derived in Question 1 above.

   $O(1) < O(n\log n) < O(n^2\log n) < O(n^3) < O(n^4)$

   b. $10000n$, $100n^2$, $\sqrt{n}$, $10n^3$, 2^n, $\sqrt{n} * \log n$, $n\log n$

   $O(n)$ $\quad O(\sqrt{n}) < O(\sqrt{n}\log n) < O(n) < O(n\log n) < O(n^2) < O(n^3) < O(2^n)$

   c. $(7/2)n$, $n^2\log n$, $(3/2)^n$, $n^{3/2}$, $n\log n^2$, $(\log_n 3)/(\log_n 2)$, 1000000

   $O(n)$ $\quad 1.5^n$ $\quad 2n\log n$ $\quad \log_2 3 \rightarrow O(1)$

   $O(1) < O(n) < O(n\log n) < O(n^{3/2}) < O(n^2\log n) < O((3/2)^n)$

3. An algorithm with complexity $O(n^{1.5})$ takes 5 ms to process 1000 data items.

   a. Estimate how long it will take to process 1000,000 data items.

$(1000)^{1.5} \rightarrow 5$ ms

$(1000000)^{1.5} \rightarrow x$?

$x = \frac{5 \times (1000000)^{1.5}}{(1000)^{1.5}}$

$= 158113.88$ ms

   b. Estimate how much data can be processed in 100 ms.

$5 \rightarrow (1000)^{1.5}$

$100 \rightarrow x^{1.5}$

$x^{1.5} = \frac{(1000)^{1.5} \times 100}{5}$

$x = 1000 \times \sqrt[1.5]{20} \approx 7368$ items

4. Software packages A and B spend exactly $T_A(n) = c_A n^2$ and $T_B(n) = c_B n^3 + 500$ milliseconds to process n data items, respectively, where c_A and c_B are some constants. During a test, A takes 1000 milliseconds and B takes 600 milliseconds to process n = 100 data items. Which package is better for processing 10 data items? Which package is better for processing 1000 data items? (Show steps).

$T_A = C_A n^2$

$1000 = C_A \times 100 \times 100$

$\therefore C_A = \frac{100 \times 100}{1000} = 0.1$

$T_B = C_B n^3 + 500$

$600 = C_B \times 100 \times 100 \times 100 + 500$

$\therefore C_B = \frac{100}{100 \times 100 \times 100} = 0.0001$

$n = 10$

$T_A = C_A n^2 = 0.1 \times 10 \times 10 = 10$ ms

$T_B = C_B n^3 + 500 = 0.0001 \times 10^3 + 500 = 500.1$ ms

A better

$n = 1000$

$T_A = C_A n^2 = 0.1 \times 1000^2 = 100,000$ ms

$T_B = C_B n^3 + 500 = 0.0001 \times 1000^3 + 500 = 100,500$ ms

A better

5. Give one example each of an algorithm that has the following time complexity:

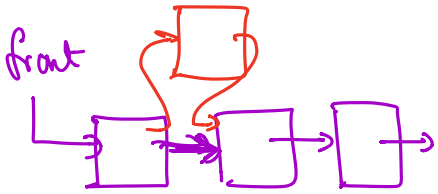
   a. O(n) → Sequential Search of an unordered list

   b. O(n log n) → Heap Sort

6. Implement the following methods in the LinkedList class.

You may assume that the Node<T> class exists (given on the reference sheet) and the LinkedList class has the following structure:

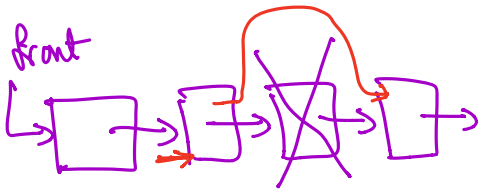
```
public class LinkedList<T>
{
   private Node<T> front;
   private int count;

   public LinkedList()
   {
        front = null;
        count=0;
   }
```



a. Add a given item after the first node. If the list is empty, just return.
b. Remove the second last item (last but one) in the list. If the list has only one node, just return.

a)

```
public void addSecond (T item)
{  if (front == null) return;
   else
    {  Node<T> newN = new Node<T>(item,
                                  front.getNext());
       front.setNext(newN);
       count++;
    }
}
```



b)

```
public void removeLastButOne()
{  if ((count==0) || (count==1)) return;
   if (count==2)  front = front.getNext();
   else
    {  Node<T> curr = front;
       while (curr.getNext().getNext().getNext()!=null)
                curr = curr.getNext();
       curr.setNext(curr.getNext().getNext());
    }
   count--;
}
```

7. Show the step-by-step process of how the binary search algorithm would work on the following array of integers.

-5	-2	10	12	17	25	35	45	55	63	105
0	1	2	3	4	5	6	7	8	9	10

for the following search:

a) Search for 13

lo = 0 hi = 10 mid = $\frac{0+10}{2}$ = 5 a[5] = 25 13 < 25. Go left

lo = 0 hi = 4 mid = $\frac{0+4}{2}$ = 2 a[2] = 10 13 > 10. Go right

lo = 3 hi = 4 mid = $\frac{3+4}{2}$ = 3 a[3] = 12 13 > 12. Go right

lo = 4 hi = 4 mid = 4 a[4] = 17 13 < 17. Go left

lo = 4 hi = 3 ⟶ lo > hi Stop! 13 not found

8. What is the maximum number of searches required for a binary search algorithm of n items in the worst case when
   a. n is a power of 2 ⟶ $\log_2 n + 1$
   b. n is not a power of 2 ⟶ $\lceil \log_2 n \rceil + 1$

9. You are given the methods in the Unordered List class on the reference sheet. ***Using these methods***, implement the following method.

```
//create a new list that has the union of list1
//and list2. Order doesn't matter but items must not be
//repeated.
public static<T> List unite(List<T> list1, List<T>
list2)
{
    List<T> list3 = new List<T>();
    T item = list1.first();
    while (item != null)
    {
        list3.add(item);
        item = list1.next();
    }
    item = list2.first();
    while (item != null)
    {
        if (!list3.contains(item))
            list3.add(item);
        item = list2.next();
    }
    return list3;
}
```

10. Given the following list of integers:

3, 9, 2, 15, -5, 18, 7 5, 8

a) What is the maximum number of comparisons for a search?
b) What is the average number of comparisons assuming that all entries are searched with equal probability?
c) What is the average number of comparisons if the search probabilities are respectively the following:

0.1, 0.3, 0.05, 0.2, 0.05, 0.1, 0.05, 0.1, 0.05

d) How would you rearrange the list so as to minimize the average number of comparisons?

a) 9

b) $\frac{n(n+1)/2}{n} = \frac{(n+1)}{2} = \frac{(9+1)}{2} = 5$

c) $0.1*1 + 0.3*2 + 0.05*3 + 0.2*4 + 0.05*5 + 0.1*6 + 0.05*7 + 0.1*8 + 0.05*9 = 4.1$

d) Arrange them in decreasing order of probs.

11. You are given the ordered list class. Using the methods in this class, write a method that accepts a given ordered list of Strings and two String keys k1 and k2 and returns a new ordered list with all the elements smaller than or equal to k1 and larger than or equal to k2. You may assume that k1 < k2 but k1 and k2 may or may not be in the ordered list.

For example, if the ordered list is:
A→ C→ F → G →H→ L→ M→ N→ Z and k1 and k2 are, respectively, G and O, then return the ordered list
A→C→F→G→Z

```
public static OrderedList<String> rangeList(OrderedList<String> list1, String k1, String k2)
{
    OrderedList<String> result = new OrderedList<String>();
    int i=0;
    String check = list1.first();
    while (i < list1.size() && check.compareTo(k1)<=0)
    {
        result.insert(check);
        check = list1.next();
        i++;
    }
    while (i < list1.size() && check.compareTo(k2)<0)
    {
        check = list1.next();
        i++;
    }
```

```
while (i < list1.size())
{       result.insert(check);
        check = list1.next();
        i++;
} }     return result;
```

12. a) Write a *recursive* method named `min` to return the smallest number in an array of `double`s.
b) Write a *recursive* method named `sum` to return sum of an array of `double`s.
c) Write a *recursive* method named `log` that returns the number of times its input `int` parameter must be divided by `2` in order to get `0`.

b)

```
public static double sum(double[] a, int first, int last)
{   if (first == last)
        return a[first];
    else
        return a[first] + sum(a, first+1, last);
}
```

c)

```
public static int log (int n)
{   if (n/2 == 0) return 1;
    else return 1 + log(n/2);
}
```

a)

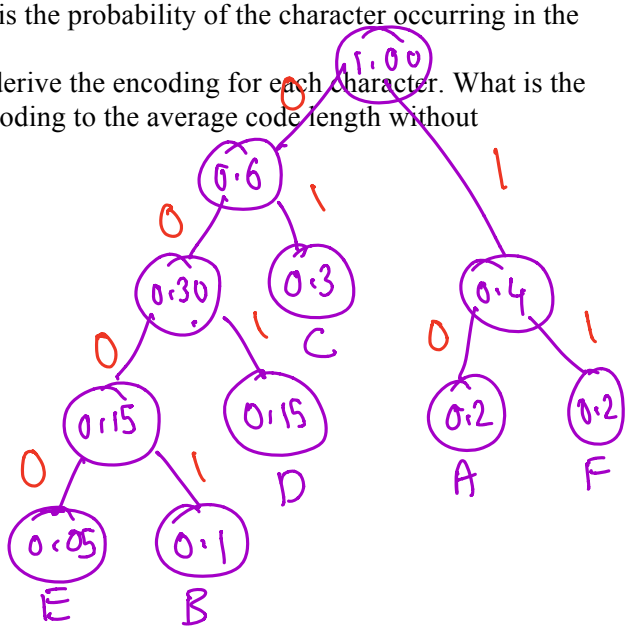
```
public static double min(double[] a, int curr,
                         double currMin)
{ if (curr != a.length-1)
  {  if (currMin > a[curr])
         currMin = a[curr];
     curr++;
     return min(a, curr, currMin);
  }
  else
   {  if (currMin > a[curr])
          return a[curr];
      else
          return currMin;
   }
}
```

Call from main

```
min(a, 0, a[0])
```

13. Given the following set of character-probability pairs:
(A,0.2), (B,0.1), (C,0.3), (D,0.15), (E,0.05), (F,0.2)
(The probability associated with each character is the probability of the character occurring in the text to be compressed.)
Build a Huffman tree for this character set and derive the encoding for each character. What is the ratio of the average code length with Huffman coding to the average code length without Huffman coding?

A 0.2
B 0.1
C 0.3
D 0.15
E 0.05
F 0.2



Codes
E : 0000
B : 0001
D : 001
C : 01
A : 10
F : 11

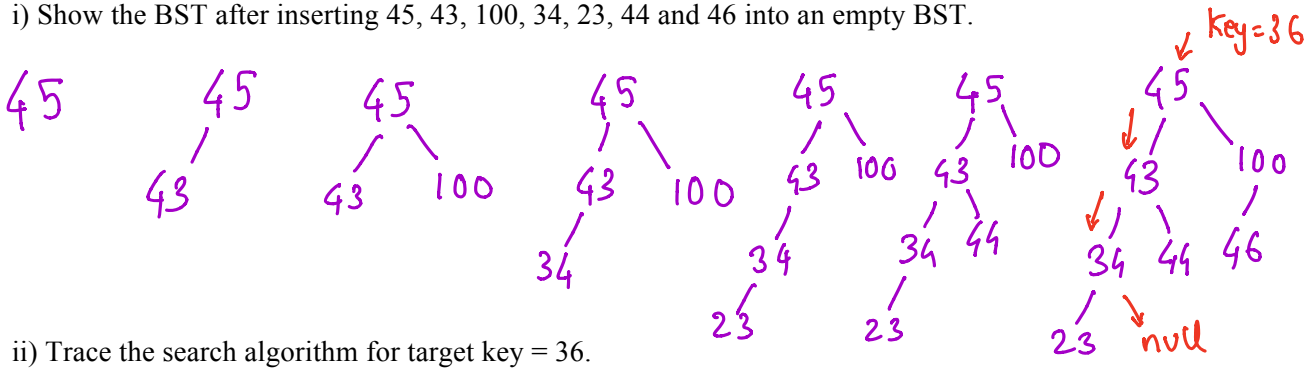
Without Huffman length of each code = 3 bits

With Huffman Average code length =
$2*0.2 + 2*0.2 + 2*0.3 + 3*0.15 + 4*0.1 + 4*0.05 = 2.45$

Compression ratio = $\frac{2.45}{3} \approx 0.82$

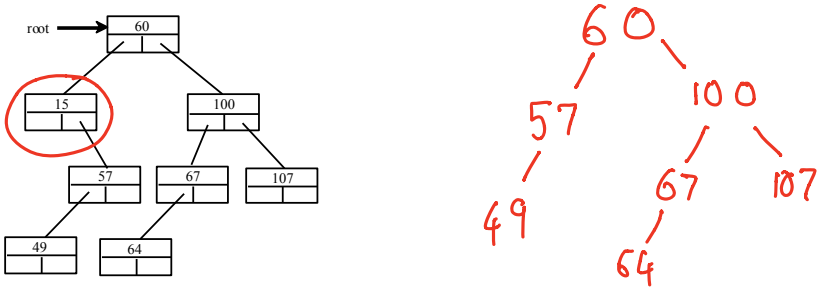
14 (a)

i) Show the BST after inserting 45, 43, 100, 34, 23, 44 and 46 into an empty BST.

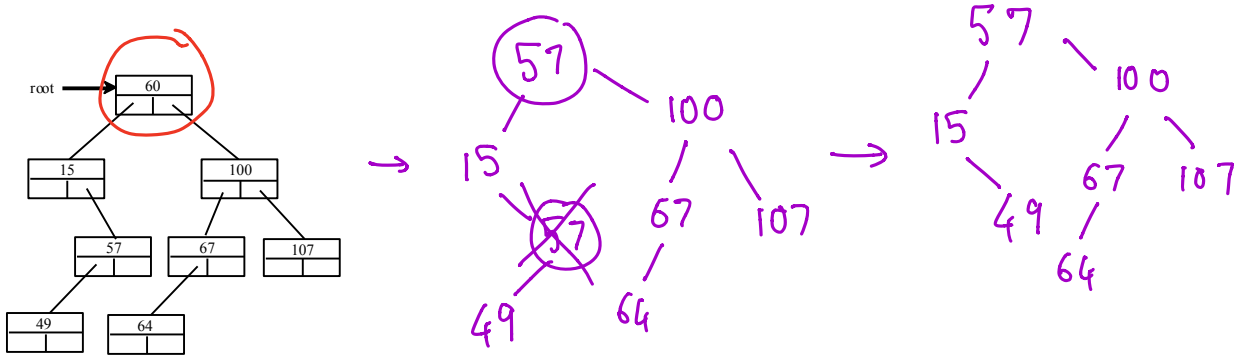


ii) Trace the search algorithm for target key = 36.

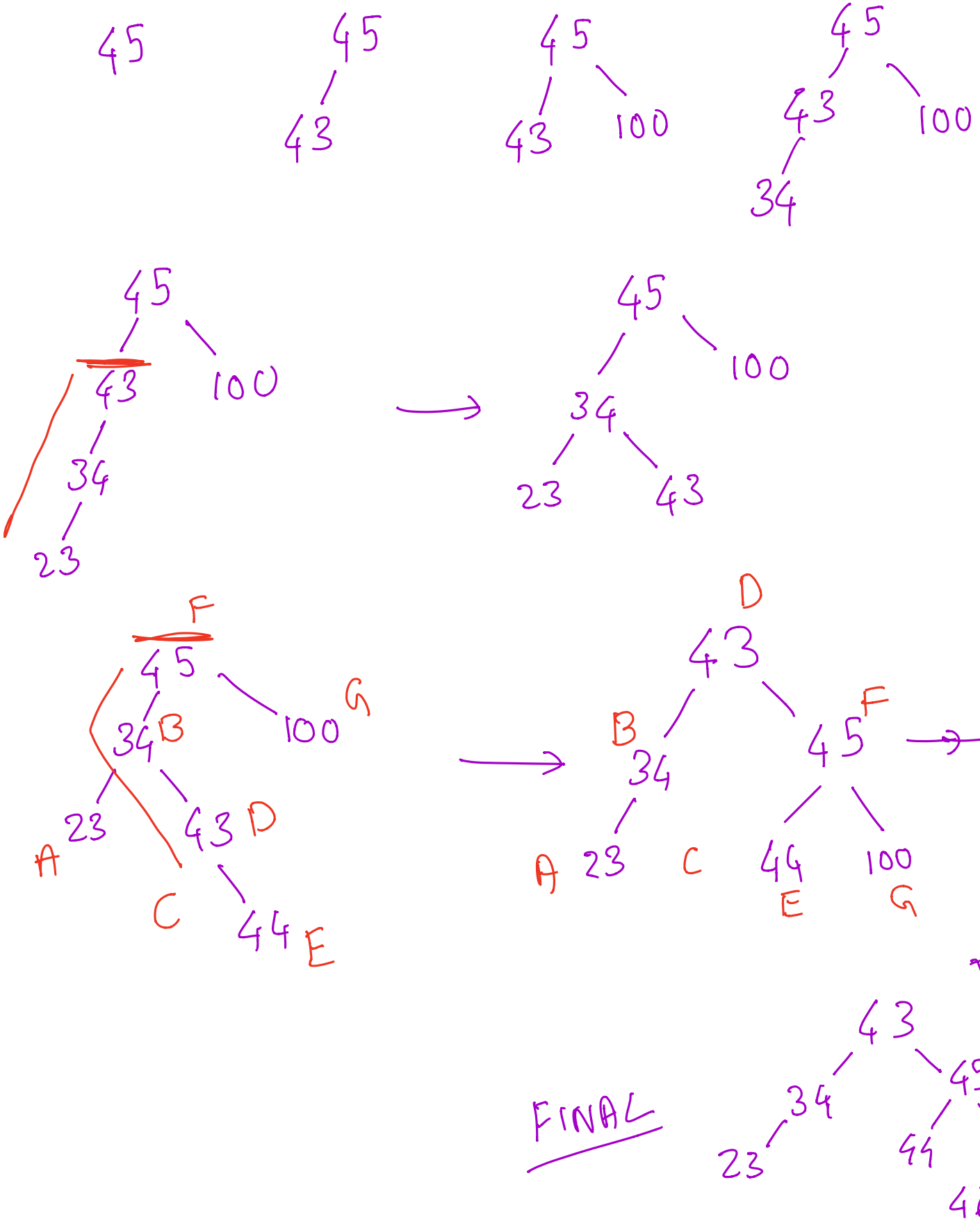
iii) Show the BST after deleting 15 from the following BST.



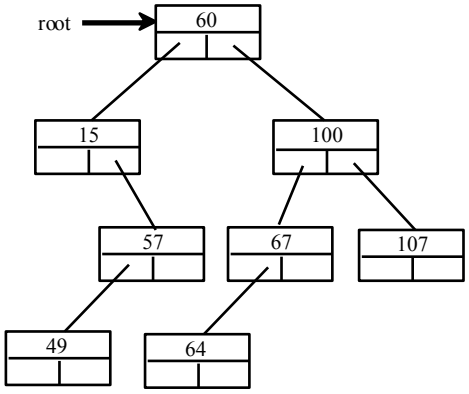
iv) Show the BST after deleting 60 from the following BST.



14 (b) Repeat the above example, but use an AVL tree instead of an ordinary BST. That means, after each operation, perform rotations if necessary.



15. Show the inorder, preorder, and postorder of the following BST.

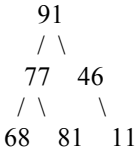


Preorder: 60, 15, 57, 49, 100, 67, 64, 107

Inorder: 15, 49, 57, 60, 64, 67, 100, 107

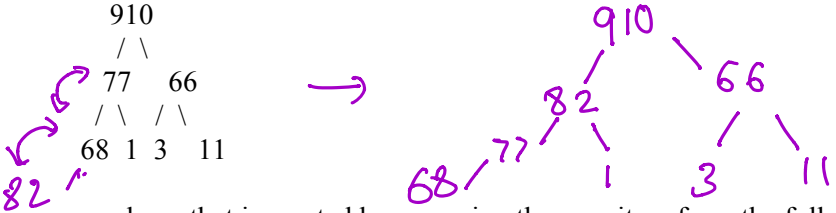
Postorder: 49, 57, 15, 64, 67, 107, 100, 60

16. a) Give two different reasons to explain why the following binary tree is not a heap:

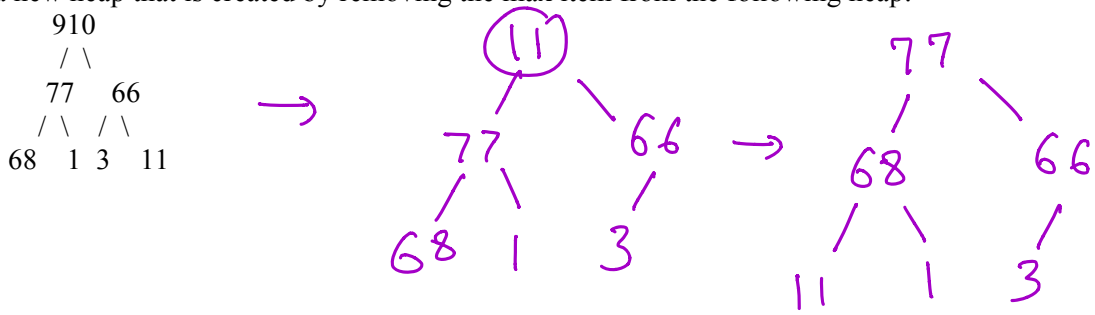


1. Not a complete binary tree
2. 81 > 77

b) Draw a new heap that is created by inserting 82 into the following heap:



c) Draw a new heap that is created by removing the max item from the following heap:



17. Starting from an empty max-heap, do the following operations ***in sequence.*** Show the heap after each operation.

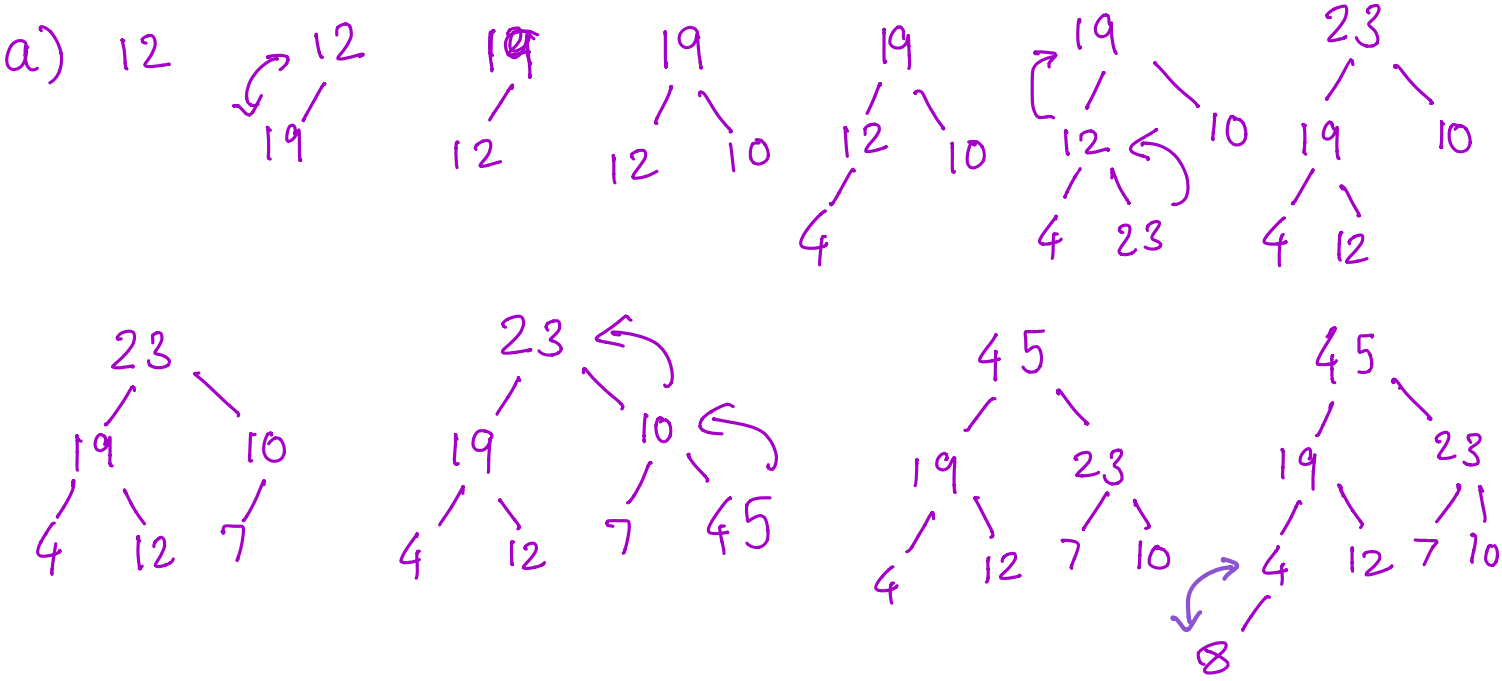
a) Insert 12, 19, 10, 4, 23, 7, 45, 8, 15

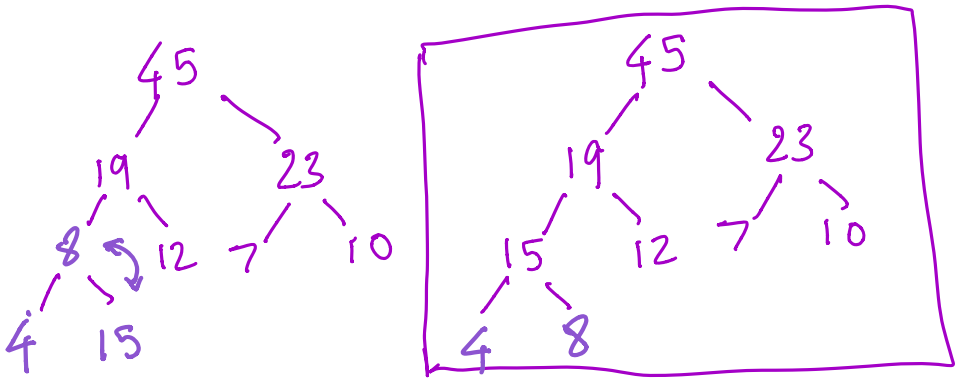
b) Delete_max

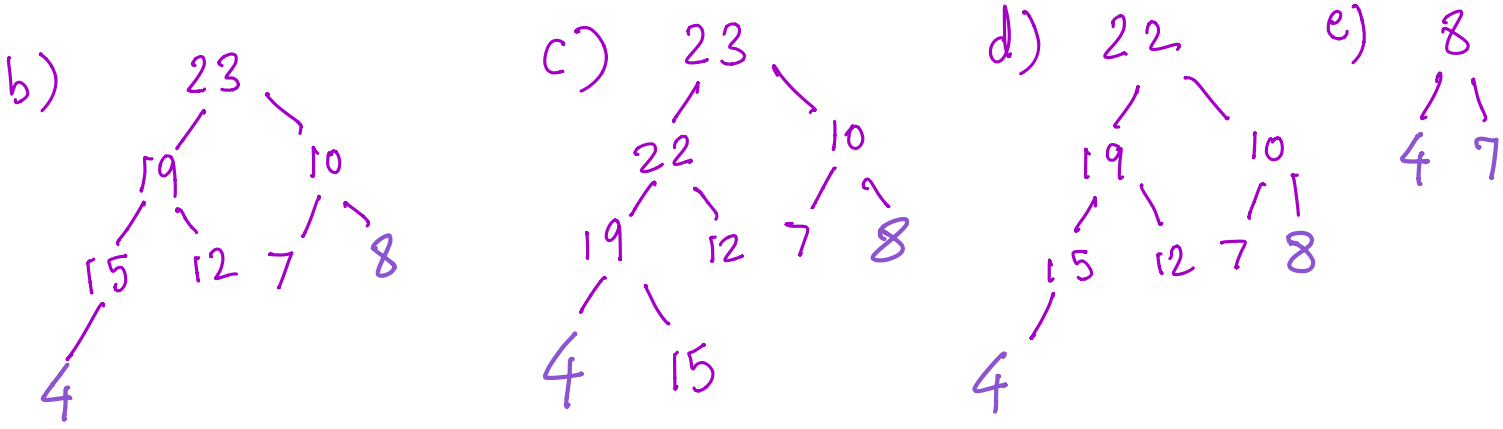
c) Insert 22

d) Delete_max

e) Delete_max five times

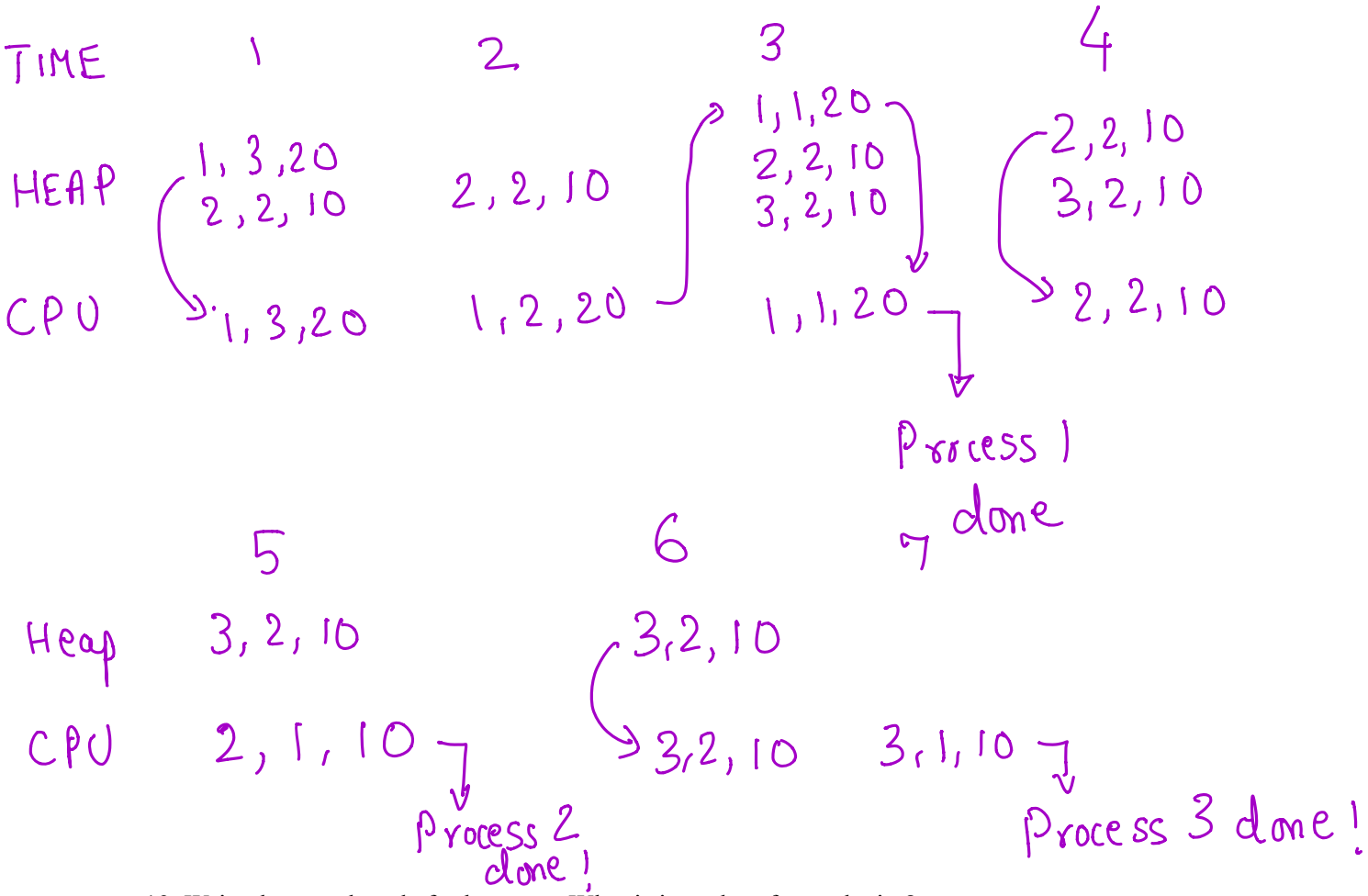






18. The following three processes arrive for processing by the CPU at the time units shown. Show how the heap is used to complete all the processes. Assume that the CPU time slice is 2 time units.
Note: Each process can be represented by a tuple (process id, time units required, priority)

Process	Time of arrival
(1, 3, 20)	1
(2, 2, 10)	1
(3, 2, 10)	3



19. Write the pseudocode for heapsort. What is its order of complexity?

Read items into the heap. → n log n steps

while (heap is not empty)
    delete Max → n log n steps

2n log n steps

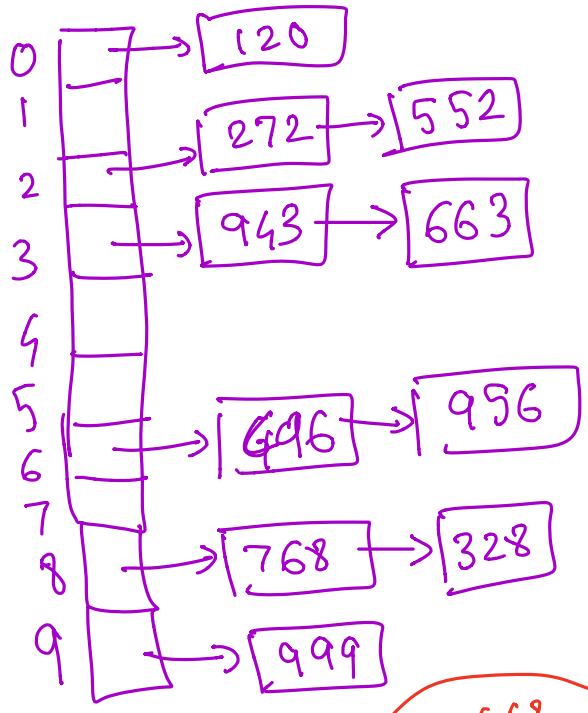
→ O(n log n)

20. You are given the following keys to be hashed into a hash table of size 10.
496, 943, 272, 768, 663, 328, 552, 956, 999, 120

Assuming that the hash function is key mod 10, show the hash table if the hash collision technique used is:

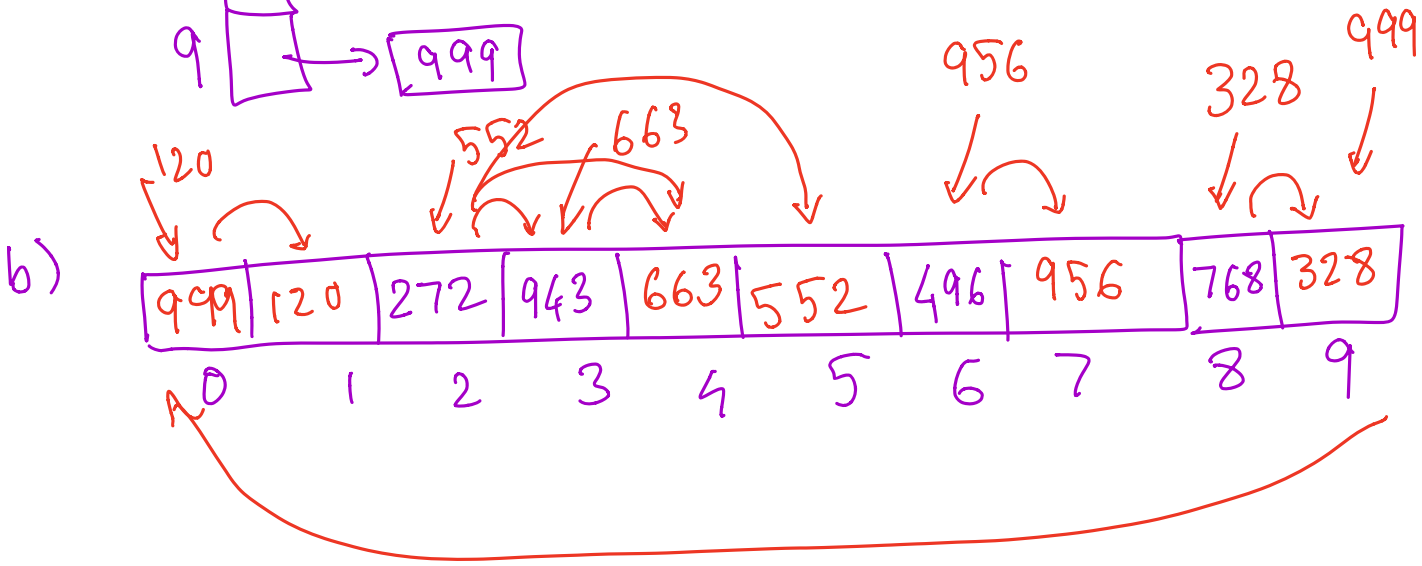
a) Separate chaining
b) Linear Probing
c) How is search performed in a hash table that uses linear probing to store items?

a)



496 % 10 → 6

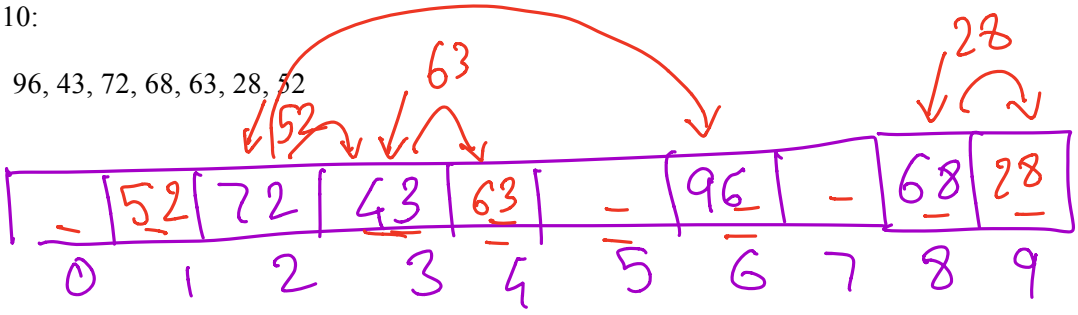
b)



c) Go to i = key mod 10
Search i, i+1, etc until key is found or empty spot is found or all n elements are searched.

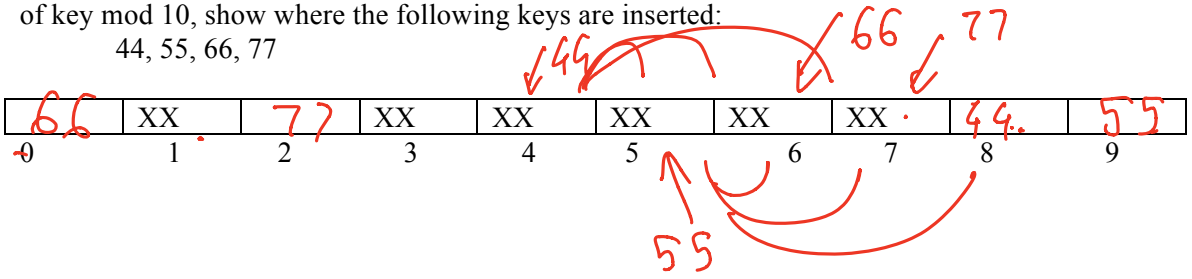
21. Insert the following keys using quadratic probing and a hash function key mod 10 into a table of size 10:

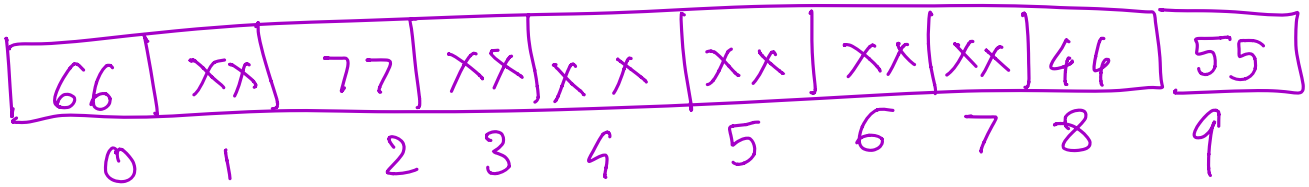
96, 43, 72, 68, 63, 28, 52



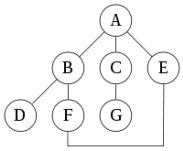
22. The figure below shows the current state of a 10-element hash table, with the occupied cells shown with an XX. Assuming that collisions are handled using linear probing and a hash function of key mod 10, show where the following keys are inserted:

44, 55, 66, 77



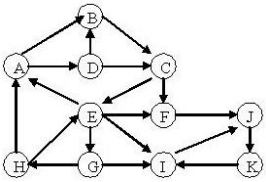


23. For the graph given below
a) State whether the graph is weighted or unweighted, directed or undirected, connected or unconnected, cyclic or non-cyclic.
b) List all vertices with degree = 3.
c) List a cycle in the graph, if it exists.
d) List all possible simple paths from A to D.
e) Determine the DFS and BFS traversals from node A.



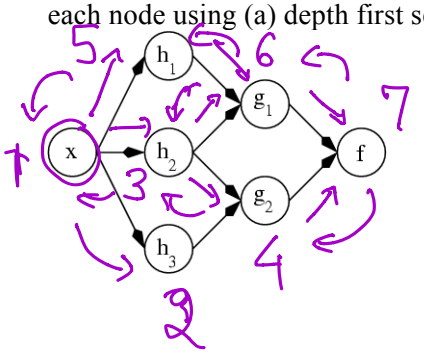
a) unweighted, undirected, connected, cyclic
b) A, B
c) A E F B A
d) A B D, A E F B D
e) DFS: A C G E F B D
BFS: A B C E D F G

24. For the following graph, which node(s) has the highest indegree? What is the highest indegree? Which node(s) has the highest outdegree? What is the outdegree? Is the graph strongly connected or weakly connected?



I: 3
E: 4
No path from F to A
∴ weakly connected

25. For the following dependency graph, derive a total order by giving topological numbers to each node using (a) depth first search and (b) breadth first search



a) DFS
nodes = 7

x	h3	h2	g2	h1	g1	f
1	2	3	4	5	6	7

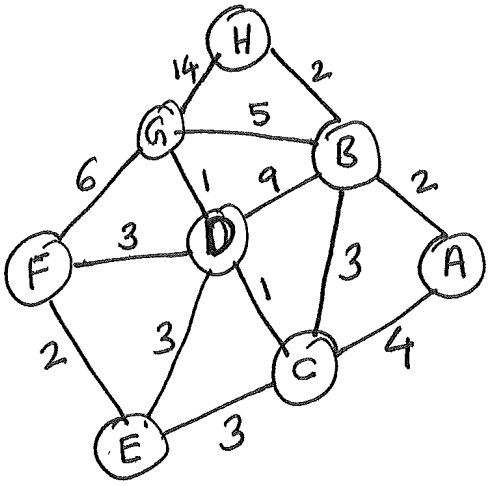
b) BFS

	x	h1	h2	h3	g1	g2	f
pred:	0	x	x	x	2	2	2
		0	0	0	x 0	x 0	x 0

x	h1	h2	h3	g1	g2	f
1	2	3	4	5	6	7

1 2 3 4 5 6 7

26. Using Dijkstra's algorithm, find the shortest paths from node A to all other nodes in the following graph:



Adjacency List

A	B	C	D	E	F	G	H
B, 2	A, 2	A, 4	B, 9	C, 3	D, 3	B, 5	B, 2
C, 4	C, 3	B, 3	C, 1	D, 3	E, 2	D, 1	G, 14
	D, 9	D, 1	E, 3	F, 2	G, 6	F, 6	
	G, 5	E, 3	F, 3			H, 14	
	H, 2		G, 1				

Sl No.	Confirmed (C)	Tentative (T)	Remarks
1	A, 0, -	B, 2, B C, 4, C	Move A to C list Its neighbours to T list
2	B, 2, B	C, 4, C D, 11, B G, 7, B H, 4, B	Move B to C list Retain old C Add D, G, H
3	C, 4, C	D, 5, C E, 7, C G, 7, B H, 4, B	Move C to C list New entry for D Add E
4	H, 4, B	D, 5, C G, 7, B E, 7, C	Move H to C list Retain old G
5	D, 5, C	E, 7, C F, 8, C G, 6, C	Move D to C list New G Add F

6	G, G, C	E, 7, C F, 8, C	Move G to C list
7	E, 7, C	F, 8, C	Move E to C list
8	F, 8, C	—	Move F to C list.