Question 1 [14 Marks]

State True or False and justify your answer:

a) The maximum number of nodes in a tree (where each node has maximum B children) that has L levels is B^L (B to the power L)

b) A queue should be used when implementing Breadth First Search (BFS).

c) In a binary tree, every node has exactly two children.

d) Binary Search Tree (BST) operation like insert typically run in O(d) time where d is the number of nodes in the tree.

e) In a heap, the left child of a node is always less than the right child of a node.

f) The linked-list implementation of a graph is more efficient than matrix representation in finding whether two vertices are directly connected or not.

g) The order in which elements are inserted in a binary search tree in unimportant.

h) An inorder traversal always processes the elements of a tree in the same order, regardless of the order in which the elements were inserted.

i) The running time of finding the maximum element in a heap is O(logN) where N is the number of elements in the heap.

j) The largest value in a binary search tree (BST) is always stored at the root of the tree.

k) An O(logN) algorithm is slower than an O(N) algorithm.

l) Adding an element to existing heap -and reheaping- takes O(N) time.

m) To delete a dynamically allocated tree, the best traversal method is postorder

n) Implementing a priority queue using a heap is more efficient than using a linked-list.

Question 2 [18 Marks]

Given the following infix expression ((x-4)*y+3)/2

a) Represent the expression using binary tree that contains either operand like x, 4, 3 or operator like * , + [do not represent the parentheses] **[4 Marks]**

b) If x= 10 and y = 2, evaluate the expression **[2 Marks]**

c) Translate the expression into postfix **[4 Marks]**

d) Show how to use the expression tree to generate postfix translation of the given expression **[4 Marks]**

e) Using stack show step by step how to evaluate this expression **[4 Marks]**

Question 3 [16 Marks]

It is required to count the nodes of a binary search tree of integer values.

a) Write down a class for the single node of the binary search tree **[4 Marks]**

b) Write down a recursive function in C++ to find the total number of nodes **[4 Marks]**

c) Write down an iterative function in C++ to find the total number of nodes **[4 Marks]**

d) Compare recursion with iteration in terms of (i) big O and (ii) simplicity **[4 Marks]**

Question 4 [20 Marks]

Write C++ code or an algorithm to

a) Check if a given array is a heap or not **[5 Marks]**

b) Add the values inside the nodes of a linked list of integers **[5 Marks]**

c) Print a linked list content in a reverse order **[5 Marks]**

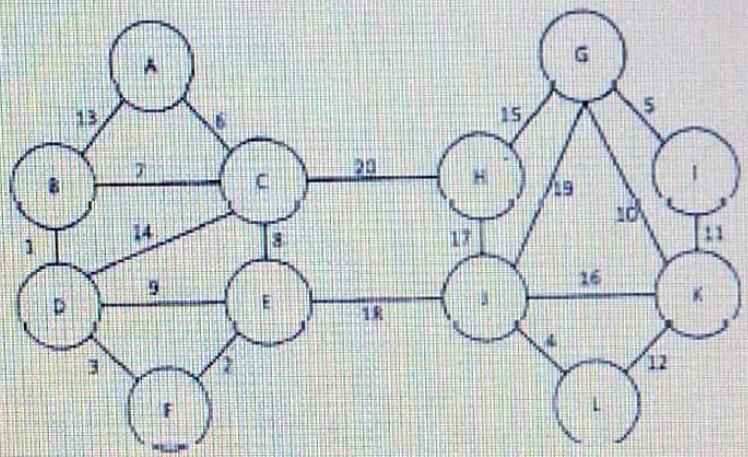
d) Print a binary tree in postorder **[5 Marks]**

Question 5 [24 Marks]

Given the following undirected graph

a) Represent this graph using matrix or linked list **[4 Marks]** and justify your choice **[2 Marks]**

b) Using **Prim** algorithm, show step by step how to find Minimum spanning tree for the given graph **[6 marks]**



c) Starting from node A, Traverse this graph using breadth first search (BFS), show the order of the nodes visited and the queue contents (assume nodes appear alphabetically i.e. A→B→C→D→E→F→G→H→I→J→K→L) mark the node once you add it to the queue, do not add already visited/marked nodes **[6 marks]**

d) Using **Dijkstra** algorithm, find the shortest path starting from node B and ending at node K **[6 marks]**

Question 6 [18 Marks]

Complete the following statements with short answers or choose one answer from between parentheses. In your answer sheet write the short answer and the question number.

a) In a single linked list, the operation of deleting the last node has big O of _____

b) Perfect hash table should have find operation with big O of _____

c) Two different binary search trees (with the same elements but different arrangement) have the same _____ traverse

d) Suppose that a heap containing 85 elements and it is stored in an array. The heap root is in index 0, the left child of node with index 5 is _____ and the parent of the child with index 37 is at index_____ the first leaf node is found at index_____.

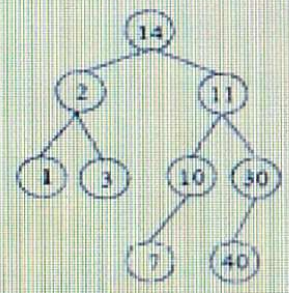
e) The largest value of a binary search tree is always stored at _____

f) A complete directed graph with 8 vertices has _____ edges.

g) The height L of a balanced full binary tree with 15 nodes is _____.

h) A hash table with the hash function **h(i) = i – Offset**, where offset is a constant is has a disadvantage of _____

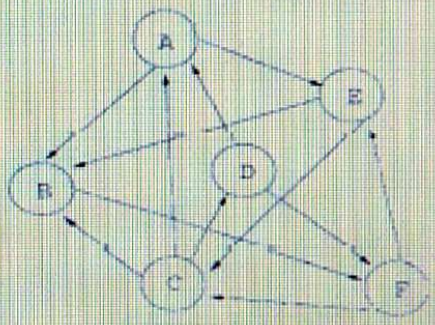
i) Given the following binary tree



The depth of the node with value 11 is _____
This binary tree is (left justified, balanced, heap, binary search tree)

j) Given the following graph:

- This is a (connected, complete, connected and complete) graph it is (directed, undirected) graph
- Draw its linked-list representation (store the vertices in alphabetical order).
- Draw one possible spanning tree starting from node A



2015

Q1

a) F Ex. Binary (node = $2^L - 1$) where B=2. , so it doesn't Hold.

b) T traversing nodes in the same level First.

c) F (None, left child, right child, two children)

d) F O(log d) but worstCase is O(d)

e) F BST

f) F finding adjacent vertices.

g) F Generally, when the root changes, tree structure also changes.

h) T when using inorder traversal, the elements will be sorted.

i) F finding O(1) , find & remove O(logN)

j) F rightMost of the Tree.

k) F O(N) algorithm is slower than O(logN) algorithm.

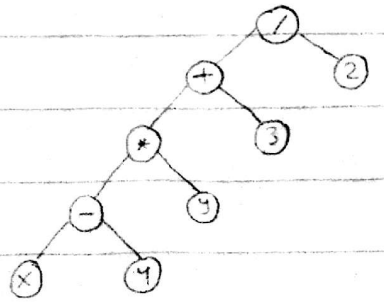
l) F O(log N)

m) T postorder [Fist delete children then parent]

n) T Insert O(logN) - getMax O(logN)

Q2 ((x-4) * y + 3) / 2

a)



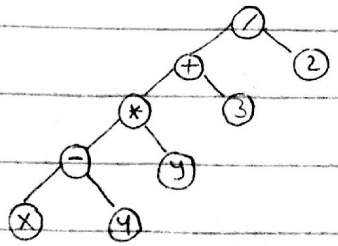
b) ((10-4) * 2 + 3) / 2 = $\frac{15}{2}$

(c)

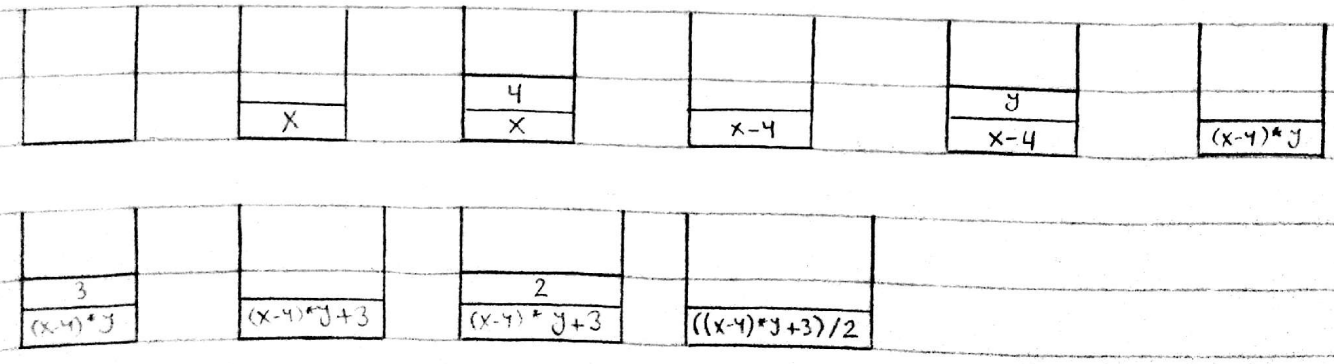
(	(	
(	((	
x	((	x
-	((-	x
4	((-	x4
)	(	x4-
*	(*	x4-
y	(*	x4-y
+	(+	x4-y*
3	(+	x4-y*3
)	—	x4-y*3+
/	/	x4-y*3+
2	/	x4-y*3+2
	—	x4-y*3+2/

(d) Post Fix [left - right - Parent]

x4-y*3+2/



(e)



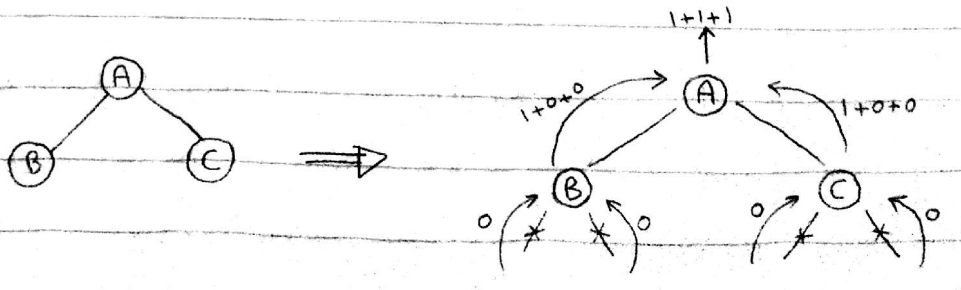
Q3 (a)

```
class node {
    int data;
    node * left;
    node * right;
    public:
    node() { data = 0;  left = NULL;   right = Null; }
    node(int d) { data = d;  left = Null;   right = Null; }
    void setData(int d) { data = d; }
    void setleft(node* L) { left = L; }
    void setRight(node* R) { right = R; }
    int getData() { return data; }
    node * getLeft() { return left; }
    node * getRight() { return right; }
    friend class BST;
};
```

(b)

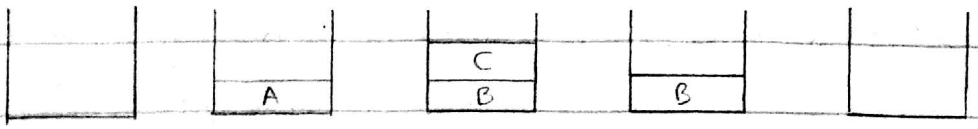
```
int CountNodes(node * t) {
    if(t == Null) { return 0; }
    else { return 1+ CountNodes(t->getRight()) + Countnodes(t->getLeft()); }
}
```

current — rightsubtree — leftsubtree



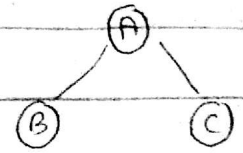
(c)

```
int CountNodesIteratively(node * t) {
    if (t == Null) { return 0; }
    int count = 0;
    stack S;
    S.push(t);
    while (! S.isempty()) {
        node * p = S.pop();
        count++;
        if (p->getLeft() != NULL)
            S.push(p->getLeft());
        if (p->getRight() != NULL)
            S.push(p->getRight());
    }
    return count;
}
```



count = 0 count = 0 count = 1 count = 2 count = 3

p = A p = C p = B



(d)

- Recursive case is simpler to implement & doesn't require extra data structure (to hold nodes) as in case of iterative case but there is more overhead due to stack usage because of successive calls (require saving parameters, values and return address In General)
- both needs to visit (traverse) each node so both O(n).

ⓐ

```
bool IsHeap(int arr[], int size){
    for(int i=0 ; i<size/2 ; i++){    // don't need to check leaf nodes.
        int Parent = arr[i];
        if (2*i+1 < (n-1)) {
            int left = arr[2*i+1];
            if (left> Parent)
                return false;
        }
        if( 2*i+2 < (n-1)) {
            int right= arr[2*i+2];
            if (right > Parent)
                return false;
        }
    }
    return true;
}
```

ⓑ

```
int GetSumOFNodes(node* p){
    int sum = 0;
    while( p != NULL){
        sum += p->getData();
        p = p-> getNext();
    }
    return sum;
}
```

(c)

```
void printAll(node* p) {
    if (p != Null) {
        printAll(p->getNext());
        cout << p->getData() << "\t";
    }
}
```

(d)

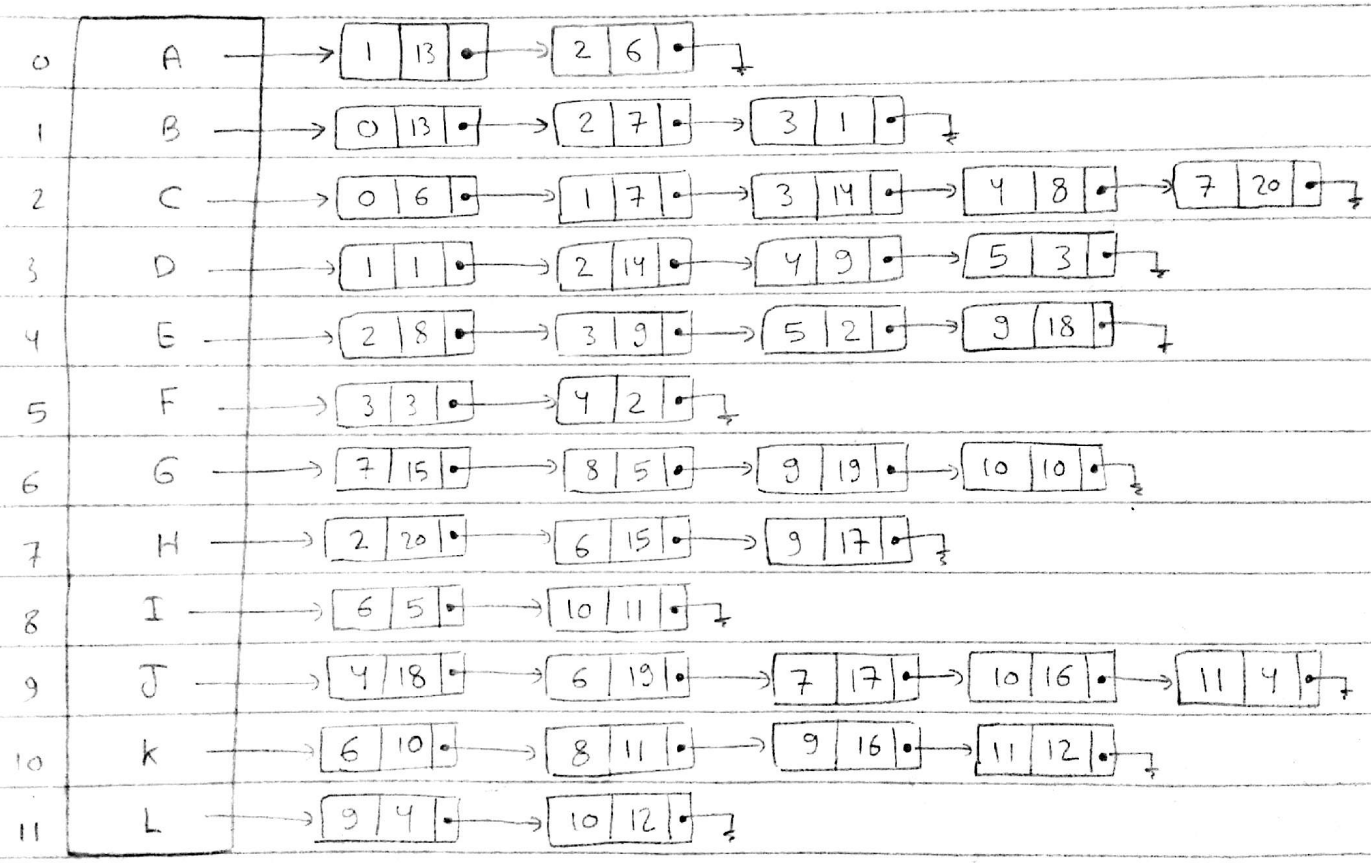
```
void printPostOrder(node* p) {
    if (p != NULL) {
        printPostOrder(p->getLeft());
        printPostOrder(p->getRight());
        cout << p->getData() << "\t";
    }
}
```

Q5 (a) Nodes = 6 + 6 = 12

Edges = 6 + 3 + 2 + 6 + 3 = 20

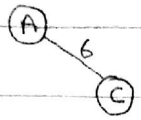
- min no of edges for connected undirected graph of 12 node is 12 − 1 = 11 (sparse)
- Max no of edges ~ ~ ~ ~ ~ is $\frac{(12)(11)}{2}$ = 66 (dense)

as we have 20 edges, it is better to use linkedlist.

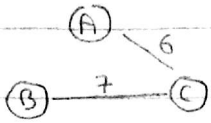


(b) prim (start with node A)

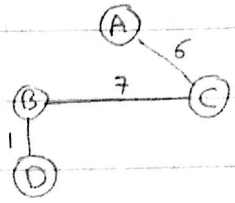
~~6~~, 13



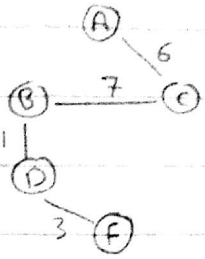
13, ~~7~~, 14, 8, 20



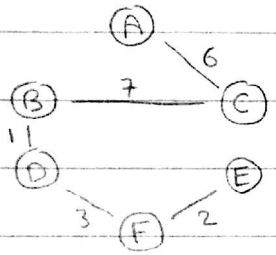
13, 14, 8, 20, X



13, 14, 8, 20, 9, ~~7~~



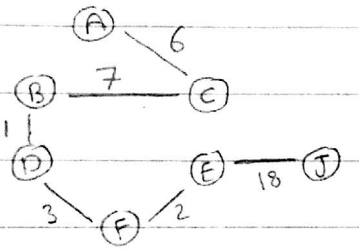
13, 14, 8, 20, 9, ~~2~~



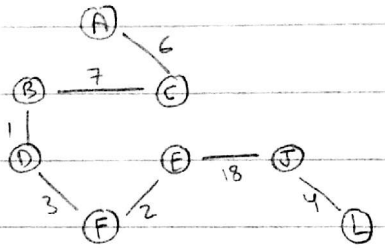
13, 14, 8, 20, 9, ~~18~~

↓ ↓ ↓ ↓

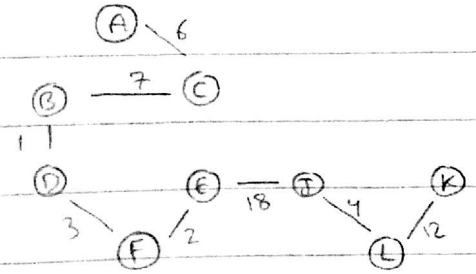
cycle cycle cycle cycle



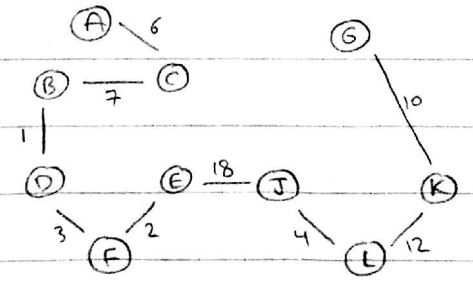
20, ~~4~~, 16, 19, 17



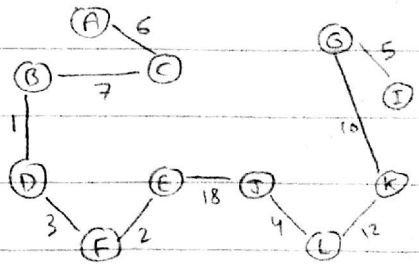
20, 16, 19, 17, ~~12~~



20, 16, 19, 17, 11, ~~16~~.



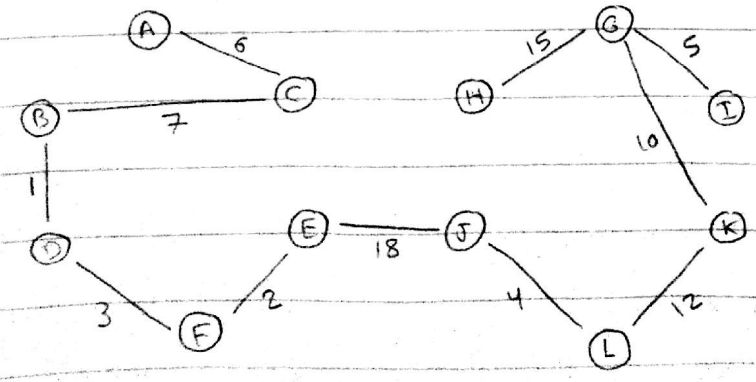
20, 16, 19, 17, 11, ~~5~~, 15



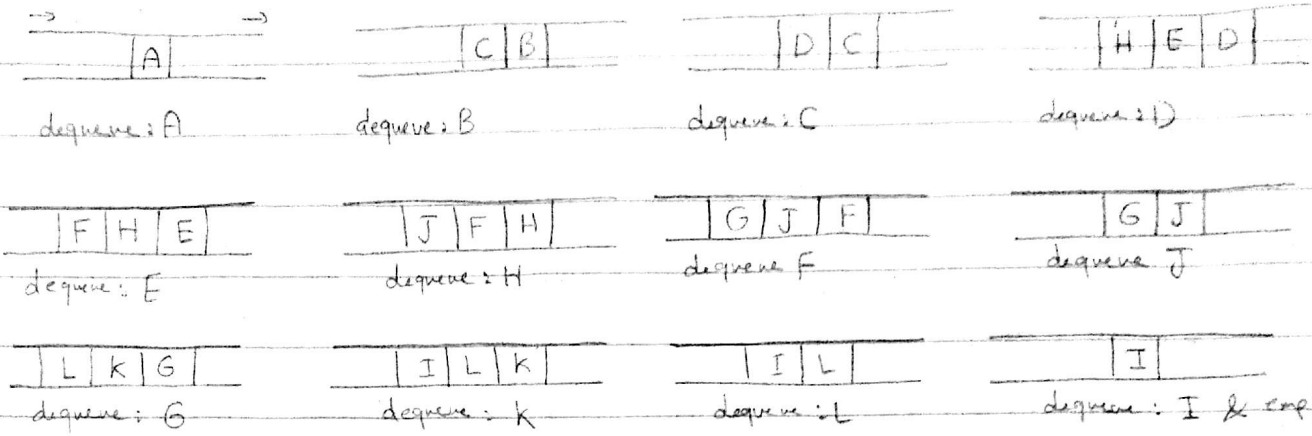
20, 16, 19, 17, 11, ~~18~~

↓

cycle

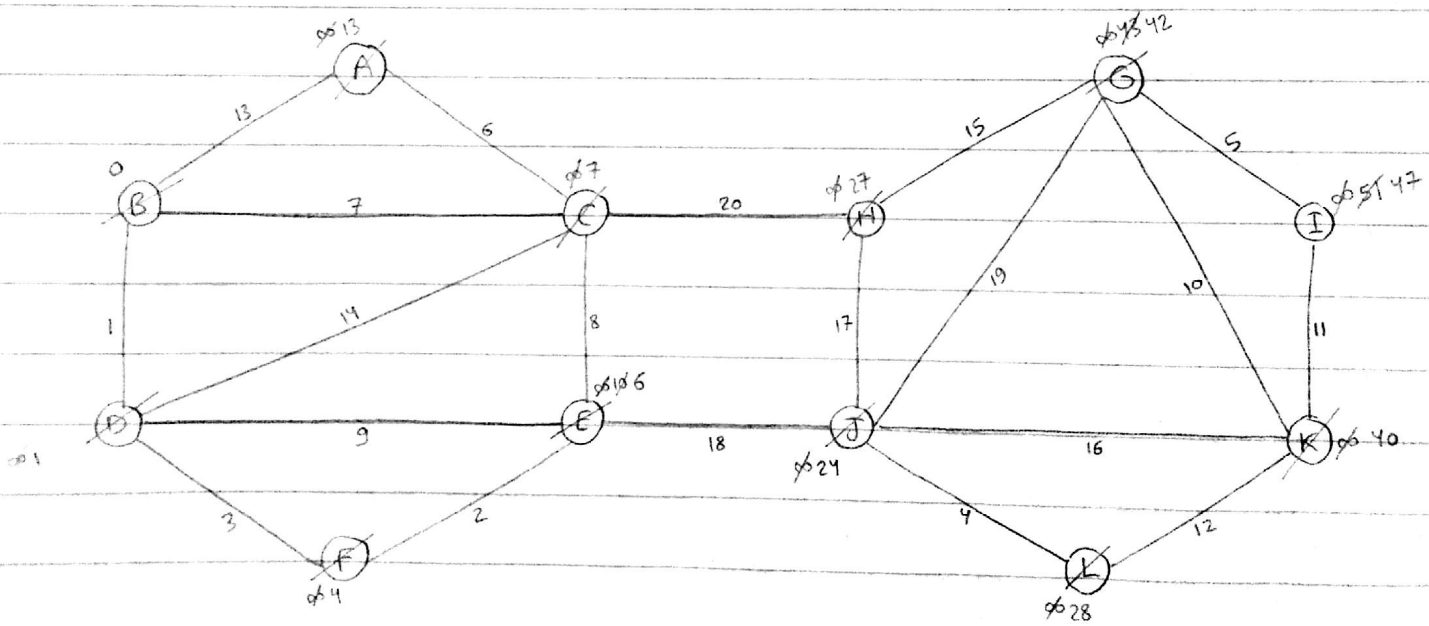


Q5 (c)

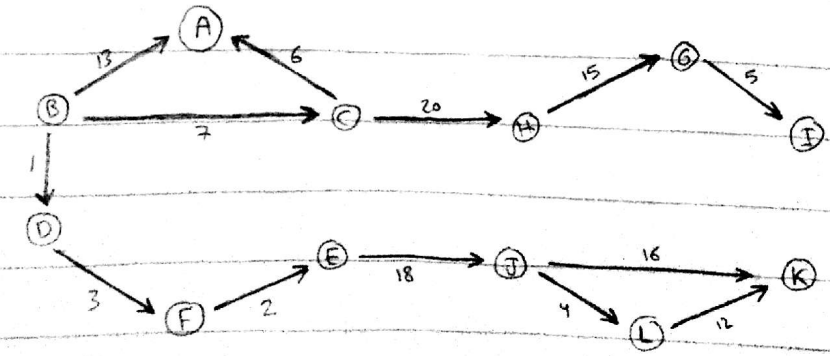


- order of the nodes visited : A B C D E H F J G K L I

(d) From B to K



sequence : B , D , F , E , C , A , J , H , L , K , G , I



paths { B D F E J K (cost = 40)
B D F E J L K (cost = 40)

Q6

(a) O(n)

(b) O(1)

(c) Inorder

(d) $5*2+1=11$, $\lfloor \frac{37-1}{2} \rfloor = 18$, 42

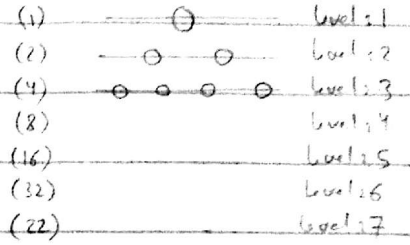
- get parent of index 85 $\Rightarrow \lfloor \frac{85-1}{2} \rfloor = 42$

OR

- Level 7 has only 22 elements that must be children of the left 11 elements in Level 6 & other elements $32-11=21$ in Level 6 are leaf nodes

total leaf nodes = 21 (level 6) + 22 (level 7) = 43 $\lceil \frac{85}{2} \rceil$ ✓

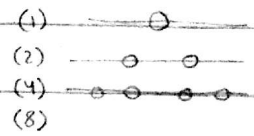
First leaf node no is $(1+2+4+8+16+11)+1 = 43$



(e) right Most

(f) 56 $(N*(N-1))$

(g) 3

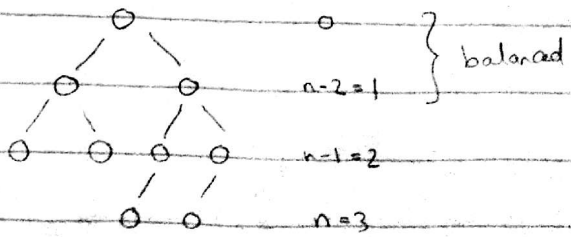


(h) $h(i) = i - \text{offset(const)}$

requires a very large table to Map each element in different location.

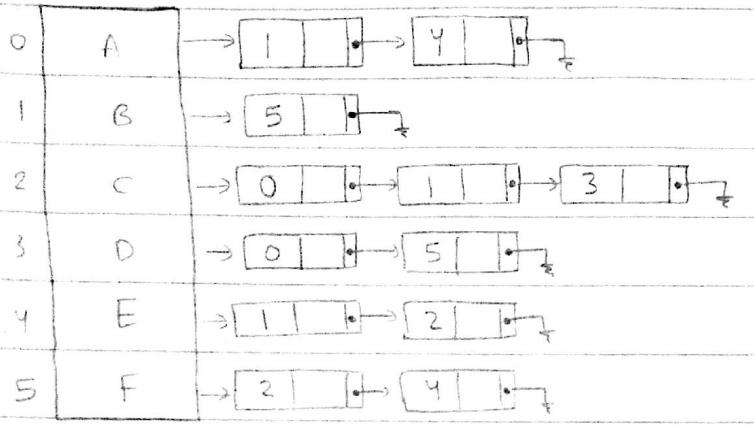
also key(i) must be always > offset

(i) depth = 1 & Balanced



(1) • This a connected graph, it is directed graph. Edges = 11 to be complete Edges should be 30

•



•