KONGU ENGINEERING COLLEGE, PERUNDURAI, ERODE – 638 052

ODD SEMESTER 2016 – 2017

MODULE TEST – III

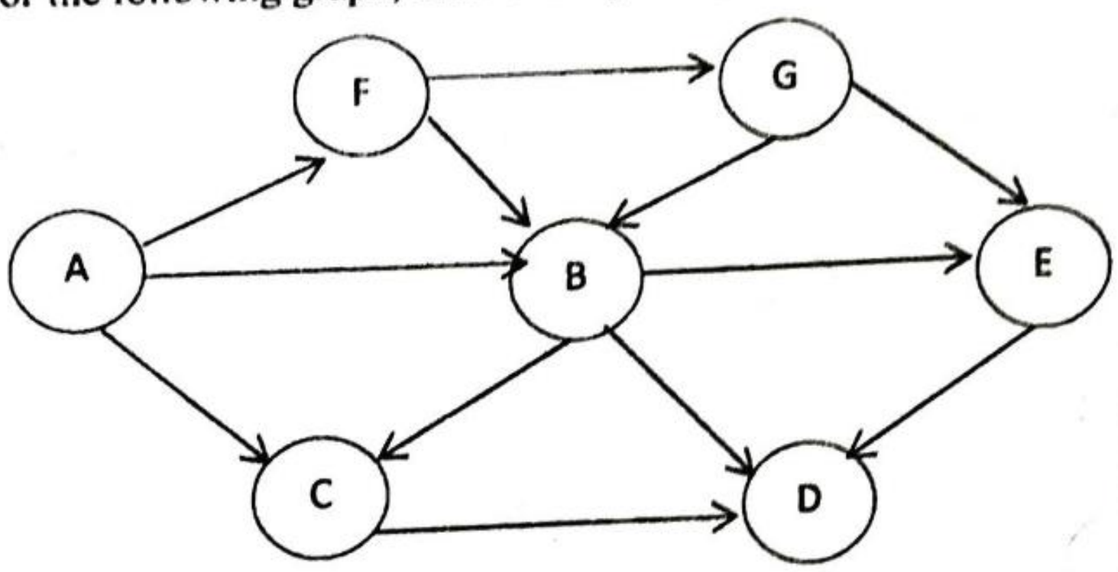
Roll No.....

Programme : B.E Branch : CSE Semester : III	Date : 03.11.2016 Time : 2.30 pm to 4.00 pm
Course Code : 14CST31 Course Name : Data Structure	Duration : 1 ½ Hours Max. Marks : 50

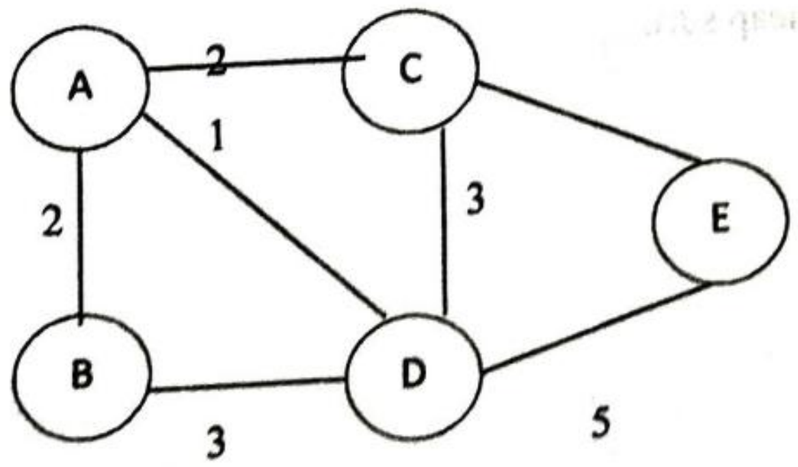
PART - A (10 X 2 = 20 Marks)

ANSWER ALL THE QUESTIONS

1. List the two properties of heap.
2. Mention any two collision resolution strategies in hashing.
3. Define graph.
4. For the following graph, draw the adjacency list representation



5. What is DAG?
6. Define articulation point.
7. Write the routine to perform DFS.
8. Find the MST for the following graph using kruskal's algorithm. Consider A as source.



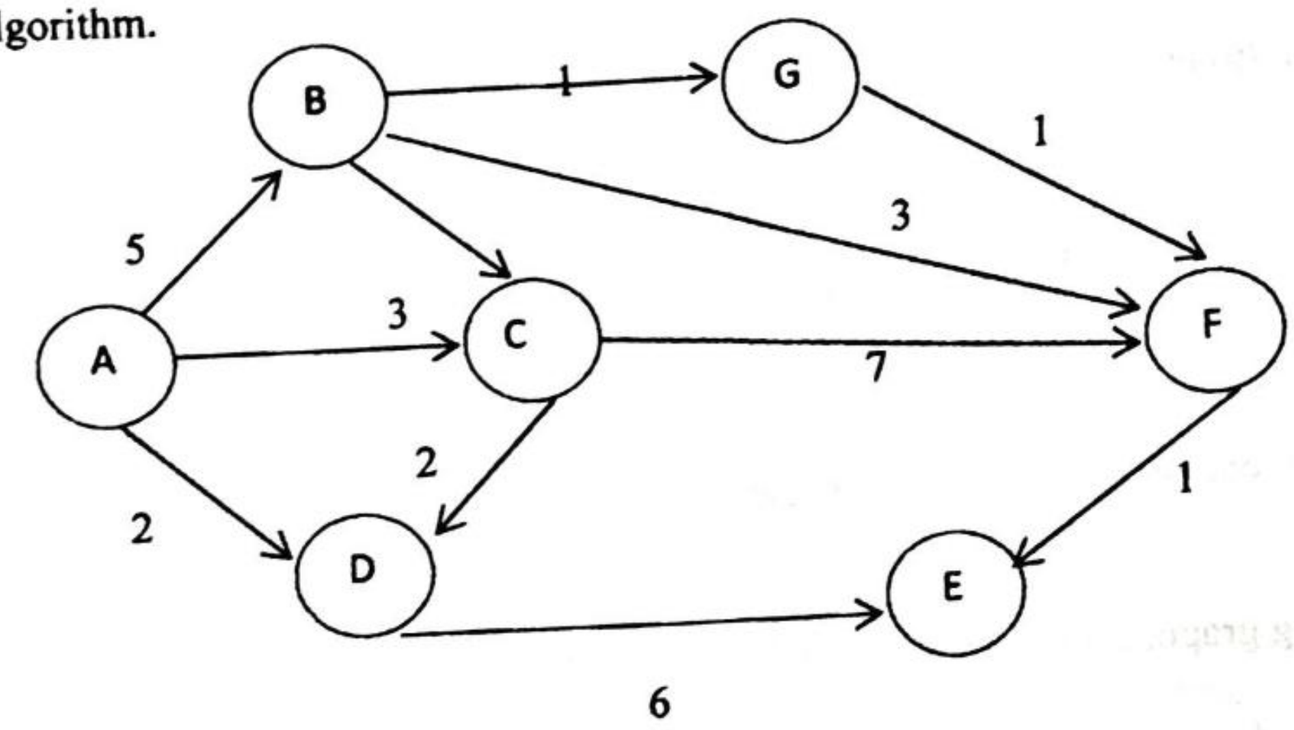
9. How will you sort the vertices in topological sorting?
10. List any four real time applications of graph.

PART - B (3 X 10 = 30 Marks)

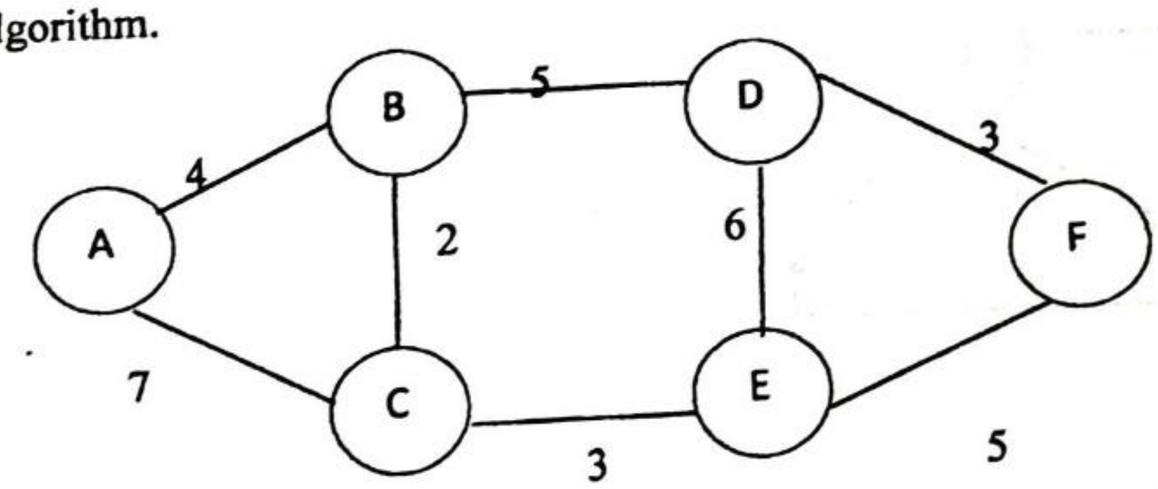
ANSWER ANY THREE QUESTIONS

11. Show the result of inserting 10,12,1,14,6,5,8,15,3,9,7 into an initially empty binary heap. (10)

12. Trace the shortest path from the vertex A to all other vertices using Dijkstra's algorithm. (10)



13. Construct a minimum spanning tree for the following graph using prime's algorithm. (10)



14. i) Given the input {4371, 1323,6173,4199,4344,9679} and a hash function h(x) = x mod 10. Show the result of applying separate chaining hash table. (5)

ii) Write the pseudo code to perform heap sort. (5)

Module Test - 3 - Scheme

PART A

1) Properties of Heap

   1) Structural property

   2) Heap Order Property

2) Collision resolution strategies

   1) Seperate Chaining

   2) Open Addressing

      1) Linear Probing

      2) Quadratic Probing

      3) Double hashing

3) Graph - Data structure

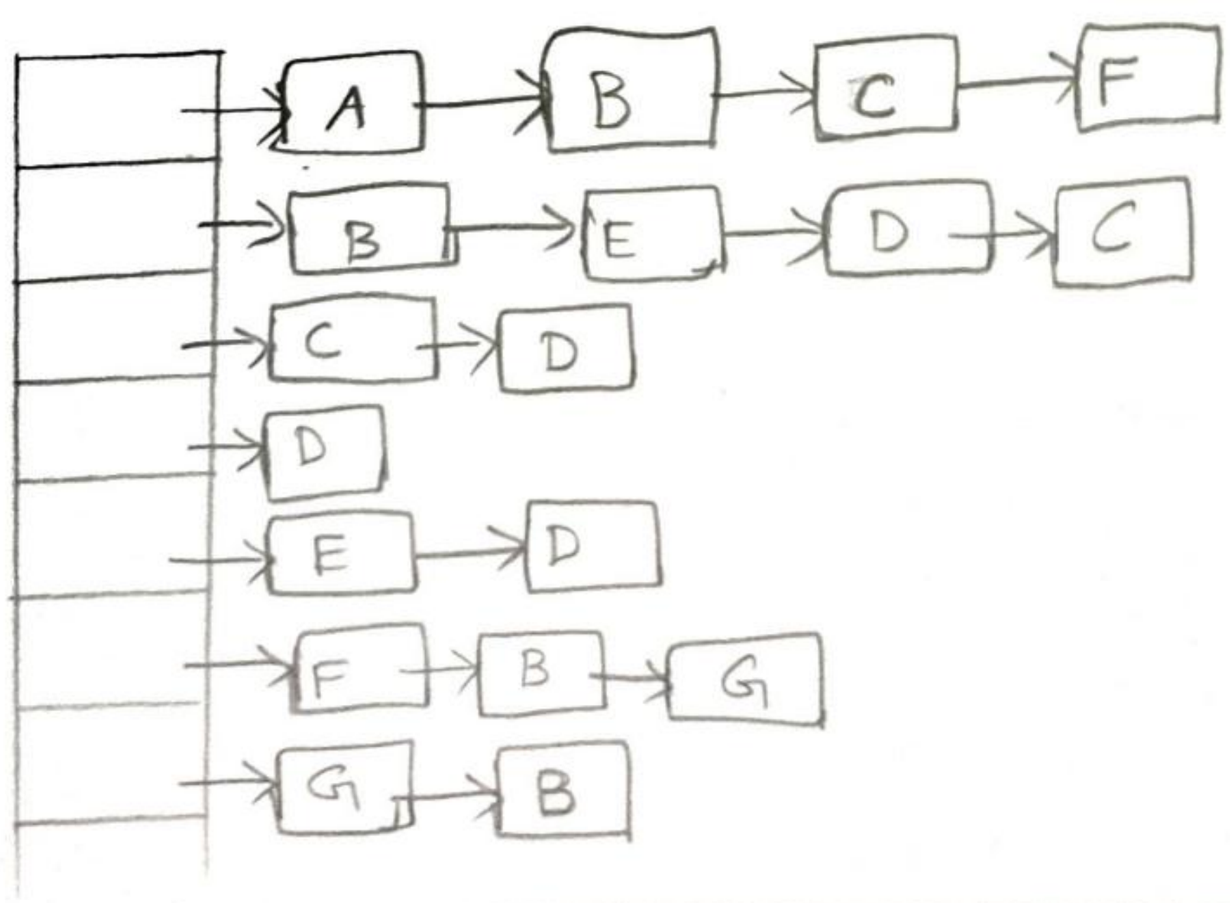
   Collection of vertices v and edges e

5) DAG

   1) Directed Acyclic Graph

   2) NO cycle (or) loop

4)



6) Articulation Point

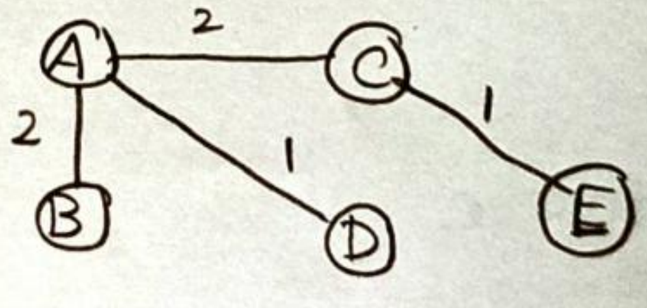
1) Graph is biconnected

2) Removal of vertex, disconnects the graph

7) DFS

```
dfs(1);
void dfs (int v, int a[10][10], int visited[10], int
{
  visited[v]=1;
  for (i=1; i<=n; i++)
    for (j=1; j<=n; j++)
      if (cost[i][j] && !visited[i])
        printf ("%d → %d = %d", v, i);
    dfs(i);
}
```

8)



Minimum Cost = 6

9) 1) Indegree

2) Outdegree

10) 1) Airport system

2) Wiring connection in home

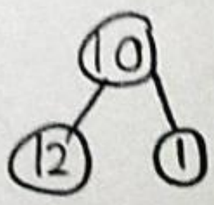
3) Course pre requisite

4) Roads & Railways

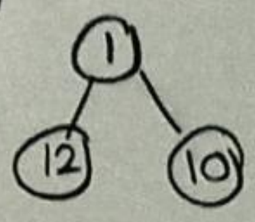
Part B

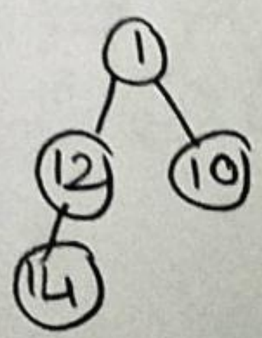
(a)

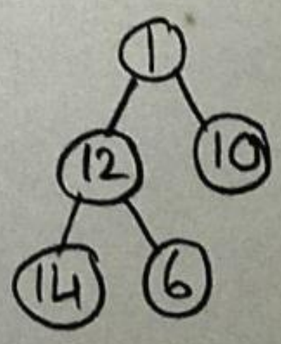
(10)



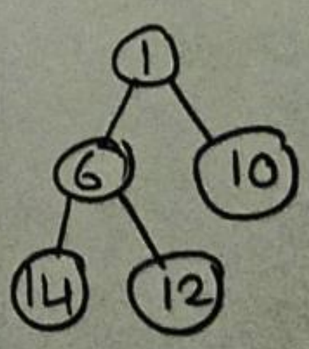
Heap order

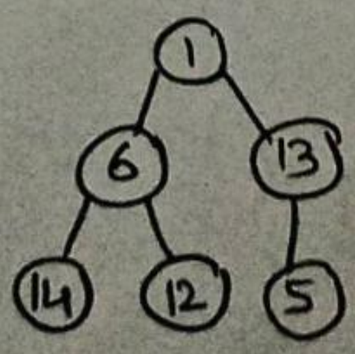


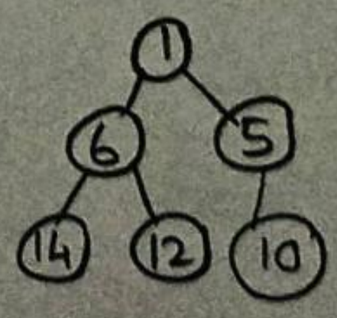


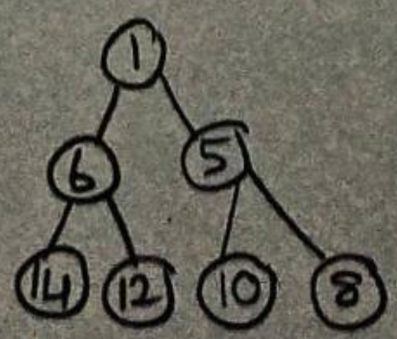


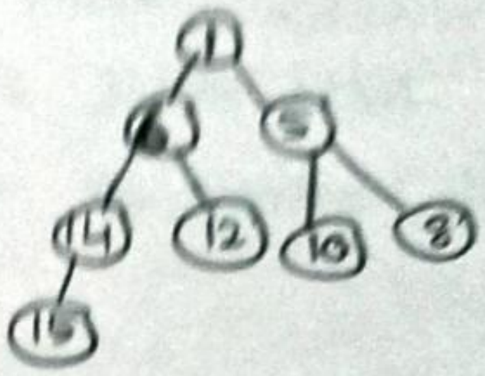
Heap order

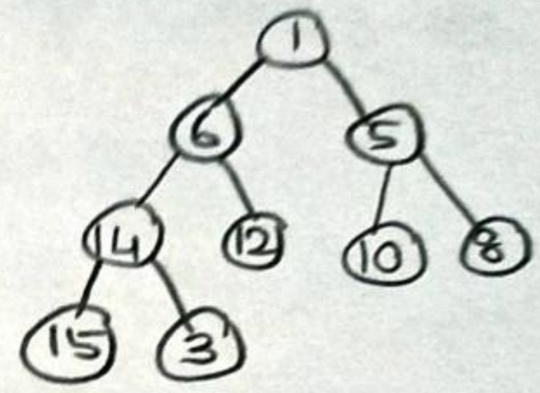




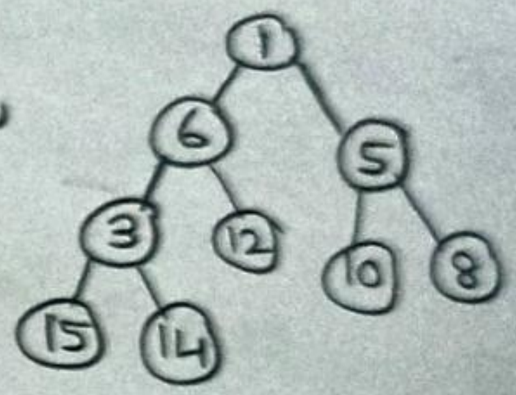


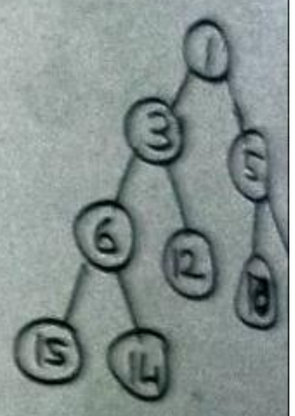


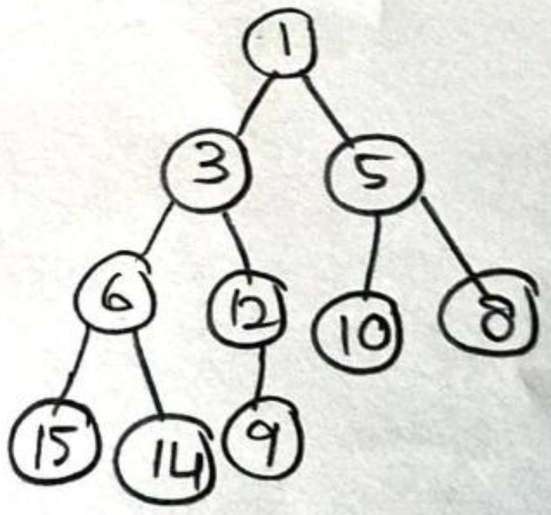




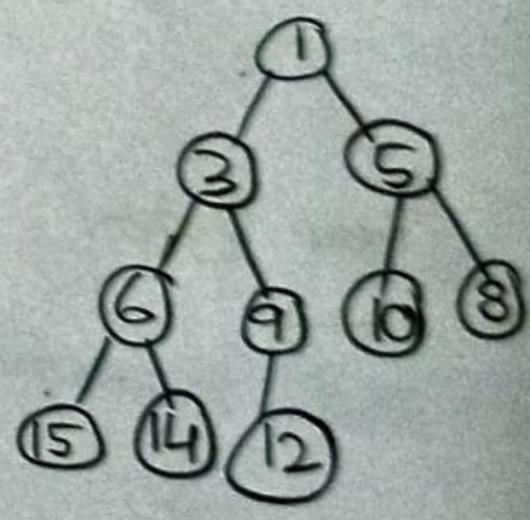
Heap order

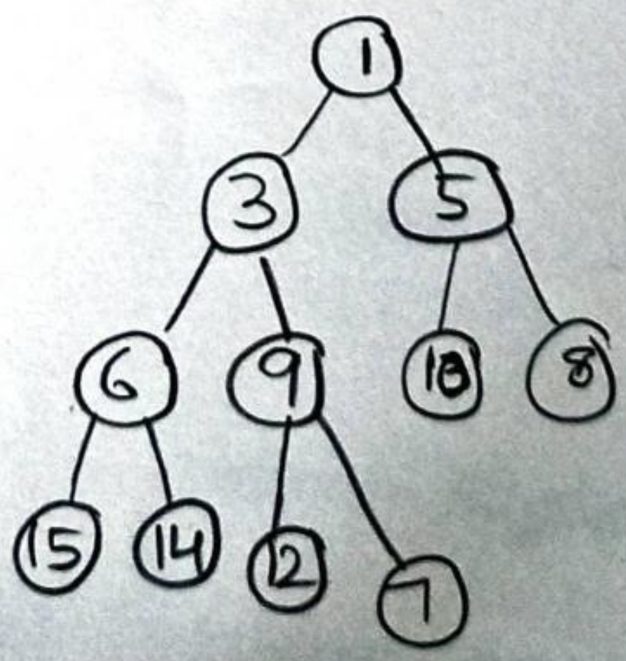




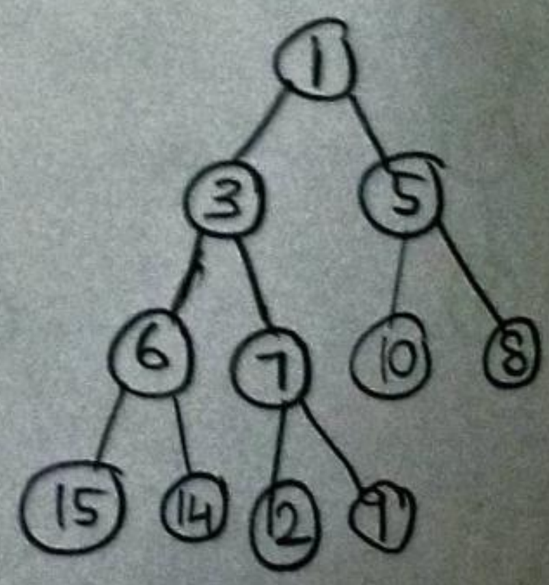


Heap order





Heap order



Initial

V	K	d_V	P_V
A	0	0	0
B	0	∞	0
C	0	∞	0
D	0	∞	0
E	0	∞	0
F	0	∞	0
G	0	∞	0

C

V	K	d_V	P_V
A	1	0	0
B	0	5	A
C	1	3	A
D	1	2	A
E	0	8	D
F	0	10	C
G	0	9	0

A

V	K	d_V	P_V
A	1	0	0
B	0	5	A
C	0	3	A
D	0	2	A
E	0	∞	0
F	0	∞	0
G	0	∞	0

B

V	K	d_V	P_V
A	1	0	0
B	1	5	A
C	1	3	A
D	1	2	A
E	0	8	D
F	0	8	B
G	0	6	B

D

V	K	d_V	P_V
A	1	0	0
B	0	5	A
C	0	3	A
D	1	2	A
E	0	8	D
F	0	8	0
G	0	8	0

G

V	K	d_V	P_V
A	1	0	0
B	1	5	A
C	1	3	A
D	1	2	A
E	0	8	D
F	0	7	G
G	1	6	B

F	K	d_v	P_v
A	1	0	A
B	1	5	A
C	1	3	A
D	1	2	A
E	0	8	D
F	1	7	G
G	1	6	B

E

V	K	d_v	P_v
A	1	0	0
B	1	5	A
C	1	3	A
D	1	2	A
E	1	8	D
F	1	7	G
G	1	6	B

A → B, Cost = 5

A → C, Cost = 3

A → D, Cost = 2

A → E, Cost = 8

A → F, Cost = 7

A → G, Cost = 6

Initial

V	K	d_v	P_v
A	0	0	0
B	0	∞	0
C	0	∞	0
D	0	∞	0
E	0	∞	0
F	0	∞	0

A

V	K	d_v	P_v
A	1	0	0
B	0	4	A
C	0	7	A
D	0	∞	0
E	0	∞	0
F	0	∞	0

B

V	K	d_v	P_v
A	1	0	0
B	1	4	A
C	0	2	B
D	0	5	B
E	0	∞	0
F	0	∞	0

C

V	K	d_v	P_v
A	1	0	0
B	1	4	A
C	1	2	B
D	0	5	B
E	0	3	C
F	0	∞	0

E

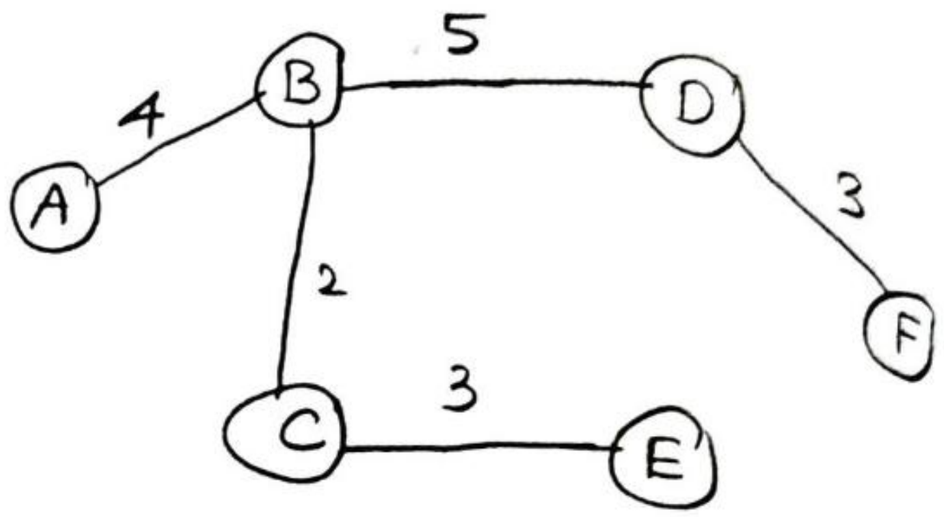
V	K	d_v	P_v
A	1	0	0
B	1	4	A
C	1	2	B
D	0	5	B
E	1	3	C
F	0	5	E

D

V	K	d_v	P_v
A	1	0	0
B	1	4	A
C	1	2	B
D	1	5	B
E	1	3	C
F	0	3	D

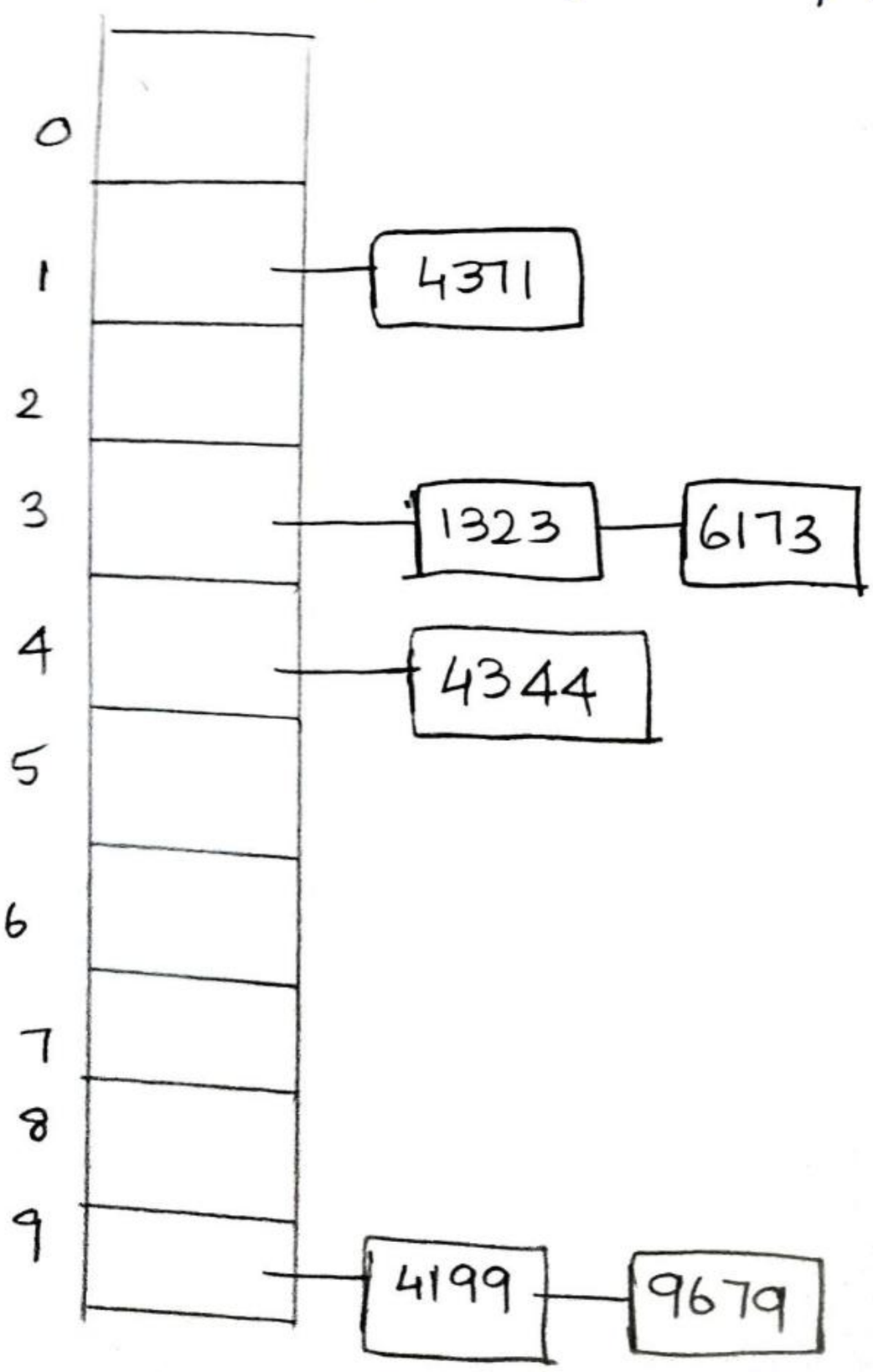
F

V	K	d_V	P_V
A	1	0	0
B	1	4	A
C	1	2	B
D	1	5	B
E	1	3	C
F	1	3	D



Total Cost = 4+2+5+3+3 = 17

4) i) 4371, 1323, 6173, 4199, 4344, 9679



4371 %. 10 = 1

6173 %. 10 = 3

1323 %. 10 = 3

4344 %. 10 = 4

4199 %. 10 = 9

9679 %. 10 = 9

```
#define leftchild(i)   (2*i);
void heapsort (int A[], int N) {
for (i=N/2; i>=0; i--)
  percdown (A,i,N);
for (i=N; i>1; i++)
{
  swap (&A[1], &A[i]);
   percdown (A,1,i);
  }
}
void percdown(int A[], int i, int N)
{
  int child, temp;
  for (temp = A[i]; leftchild (i) <N; i=child)
  {
    child = 2*i;
    if (child != N-1 && A[child+1] > A[child])
      child++;
    if (temp < A[child])
      A[i] =  A[child];
    else
      break;
  }
  A[i] = temp;
}
```

COMPUTER SCIENCE ENGINEERING · KEC

SA	03/11/16
Name and Signature of Hall Supdt. with Date	

KONGU ENGINEERING COLLEGE

PERUNDURAI, ERODE - 638 052.

(Autonomous)

ESTD 1984

Name of the Student	SABARI GOVINDARAJAN.B	Register No.	1	5	C	S	R	1	7	4
Programme	BE	Branch & Semester	CSE & III							
Course Code and Name	14CST31 Data Structures	Date	03.11.2016			No.of Pages Used	12			

MARKS TO BE FILLED IN BY THE EXAMINER

PART - A		PART - B			Grand Total Max. Marks : 50
Question No.	Max Marks : 2	Question No.		Max Marks : 10	
1	2	11	i)	10 Good	49
2	2		ii)		
3	2	12	i)	10	98.1
4	2		ii)		
5	2	13	i)	10	V. Good
6	2		ii)		Sabari
7	2	14	i)	—	
8	2		ii)	—	
9	~~0~~ 1				
10	2				
TOTAL	19	TOTAL		30	

Total Marks in words : FOUR NINE

INSTRUCTION TO THE CANDIDATE

1. Check the Question Paper, Programme, Course Code, Branch Name etc., before answering the questions.
2. Use both sides of the paper for answering questions.
3. POSSESSION OF ANY INCRIMINATING MATERIAL AND MALPRACTICE OF ANY NATURE IS PUNISHABLE AS PER RULES.

S.V.KOGILAVANI — Name of the Examiner

S Kogilavani 3/11/16 — Signature of the Examiner with Date

PART - A

1) Properties of Heap

Structural property - The tree should be complete

Heap order property - Root element of the tree should contain the minimum among all the elements of tree.

2) Collision resolution strategies

1) Separate Chaining
2) Open Addressing
   1) Linear Probing
   2) Quadratic Probing
   3) Double hashing

3) Graph

A Graph is a set of vertices $v \in V$, and edges $e \in E$. It is a data structure which find many application in real life scenarios such as airport systems

5) DAG

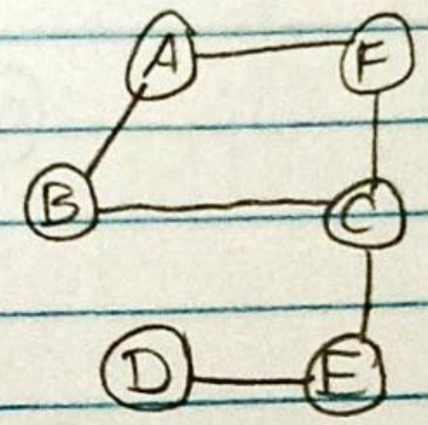
DAG - Directed Acyclic Graph

An acyclic graph is a graph in which the graph does not contain any loop, ie the starting ending vertices should be different. If this property is passed by a directed graph then it is said to be Directed Acyclic graph

6) Articulation Point

Articulation point in a graph refers to a vertex on whose removal, the rest of graph gets disconnected.

Eg:



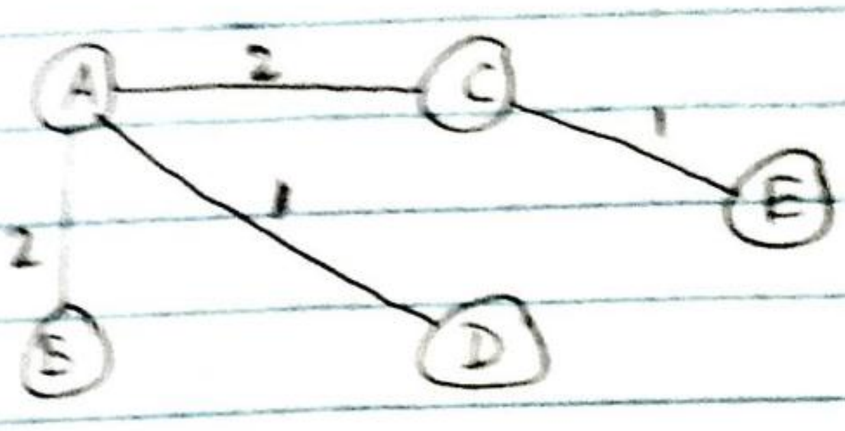
Here E is the articulation point. If vertex E is removed the graph gets disconnected.

7) DFS

```
void dfs (int v, int a[10][10], int visited[10], int n)
{
    visited[v]=1;
    for(i=1;i<=n;i++)
    if (!a[i]
```

```
void dfs (int v, int a[10][10], int visited[10], int n)
{
  visited[v]=1
   for (i=1; i<=n; i++)
   for (j=1; j<=n; j++)
    if (cost[i][j] && !visited[i])
      printf ("%d -> %d = %d", v, i);
   dfs(i);
}
```

8)



A,D - Accepted
C,E - Accepted
A,B - Accepted
A,C - Accepted
C,D - Rejected
S,D - Rejected
D,E - Rejected

Minimum cost = 2+2+1+1
= 6

9) Topological sort

Topological sort in a DAG is used to sort the order of vertices in the visited order. A particular vertex can only be visited if the previous vertex is visited. Queue can be used to implement topological sort.

10) Real time applications of Graph

1) Airport System
2) Wiring Connection in home
3) Course pre requisite in a university

ii) 10, 12, 1, 14, 6, 5, 8, 15, 3, 9, 7

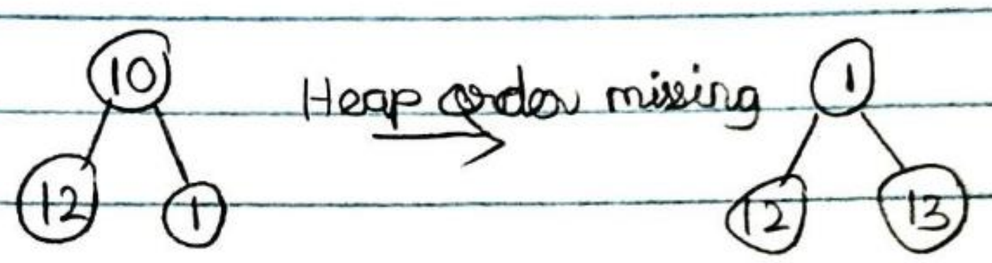
Step 1: (10)

(10)

Step 2:

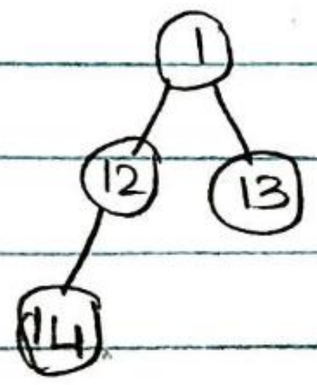
(12)

Step 3:



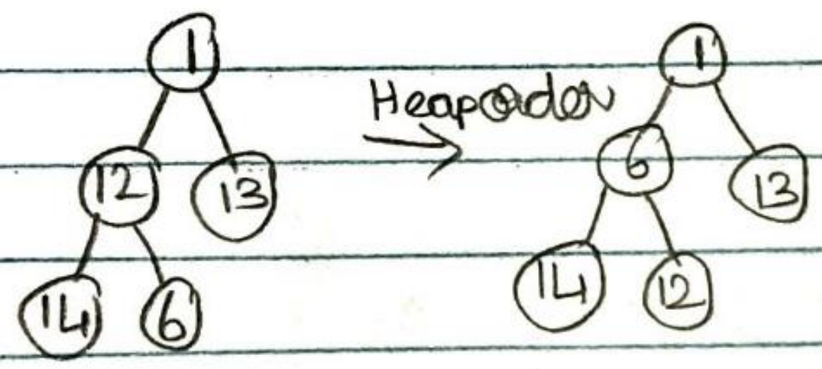
(1)

Step 4:



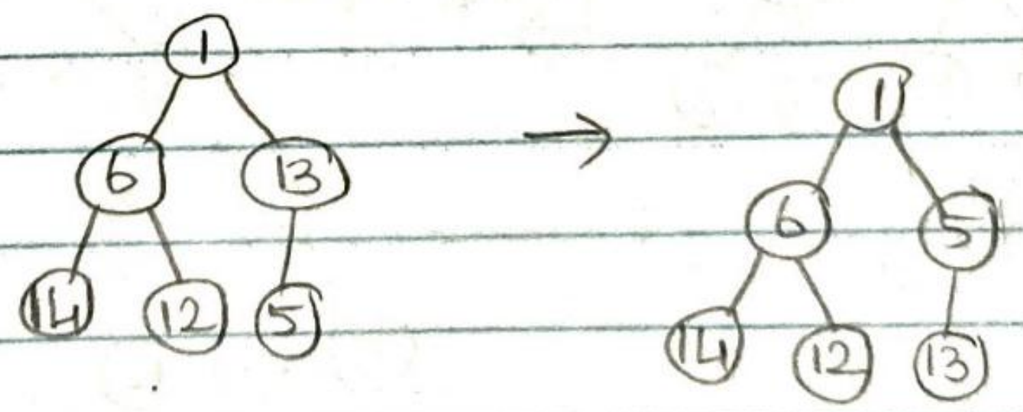
(14)

Step 5:



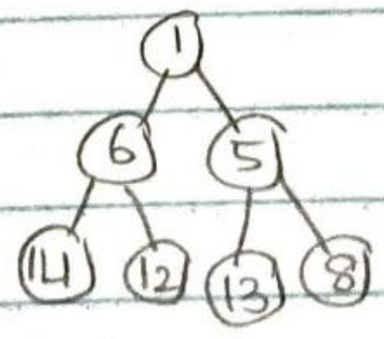
(6)

Step 6:



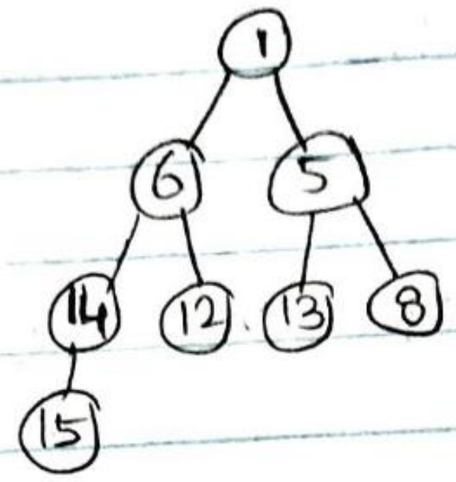
(5)

Step 7:

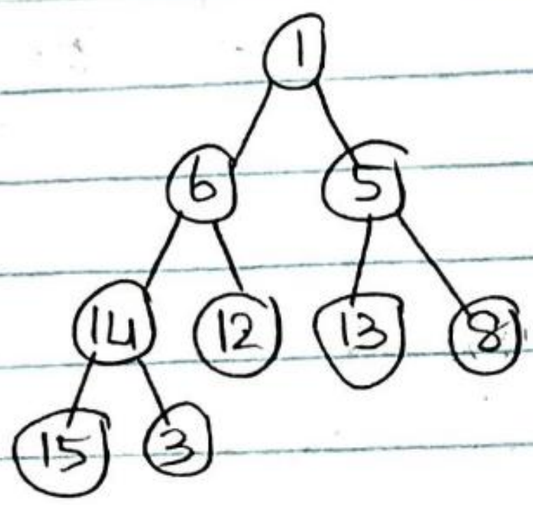


(8)

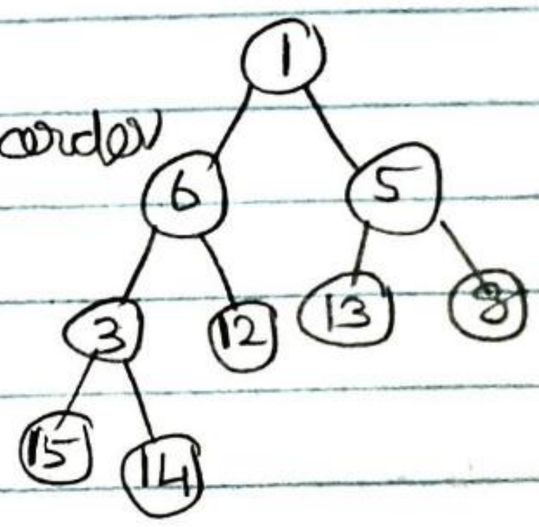
Step 8:
(15)



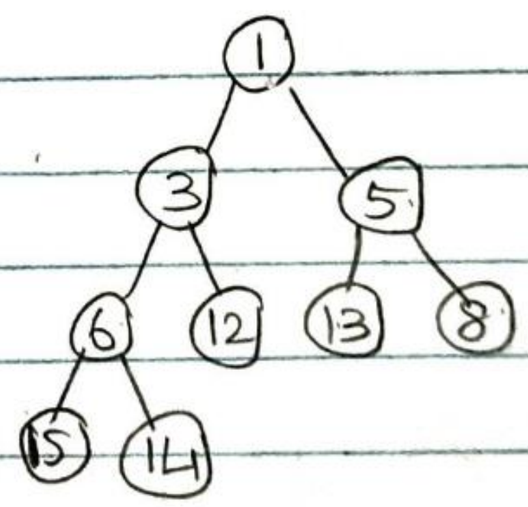
Step 9:
(3)



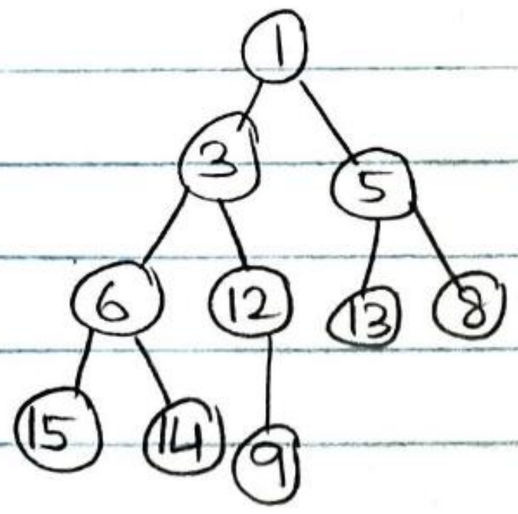
Heap order →



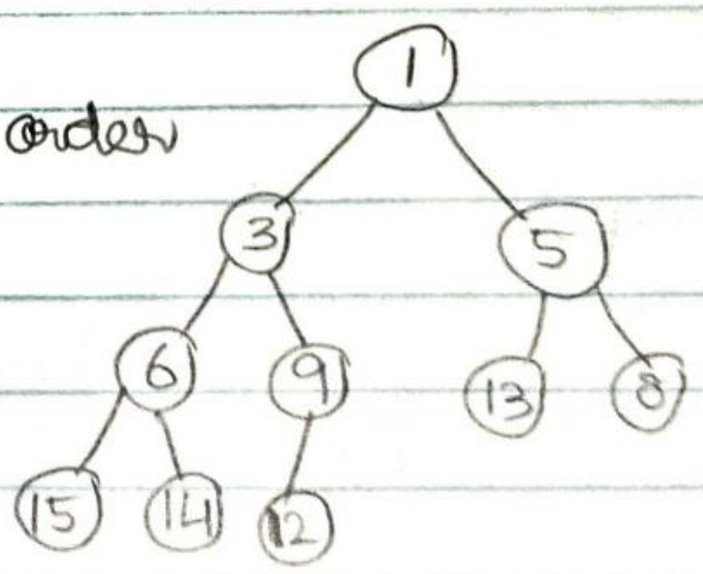
↓ Heap order



Step 10:
(9)

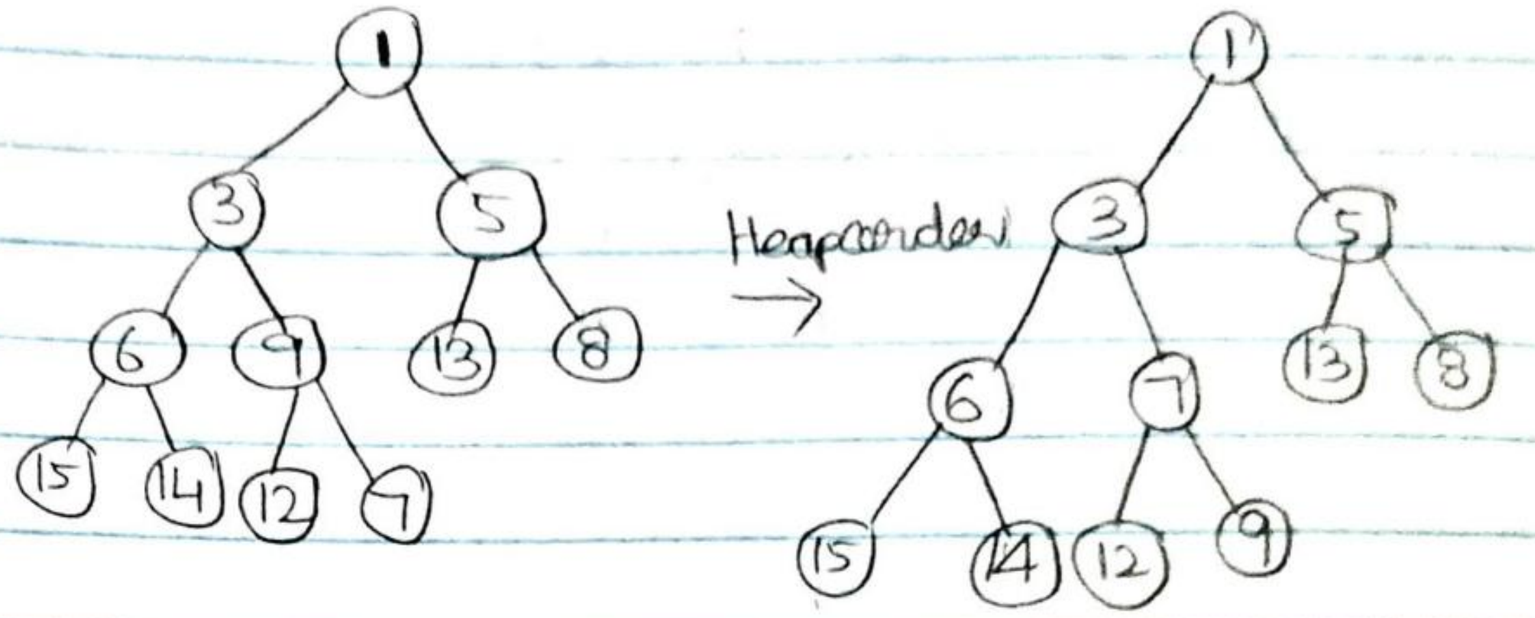


Heap order →

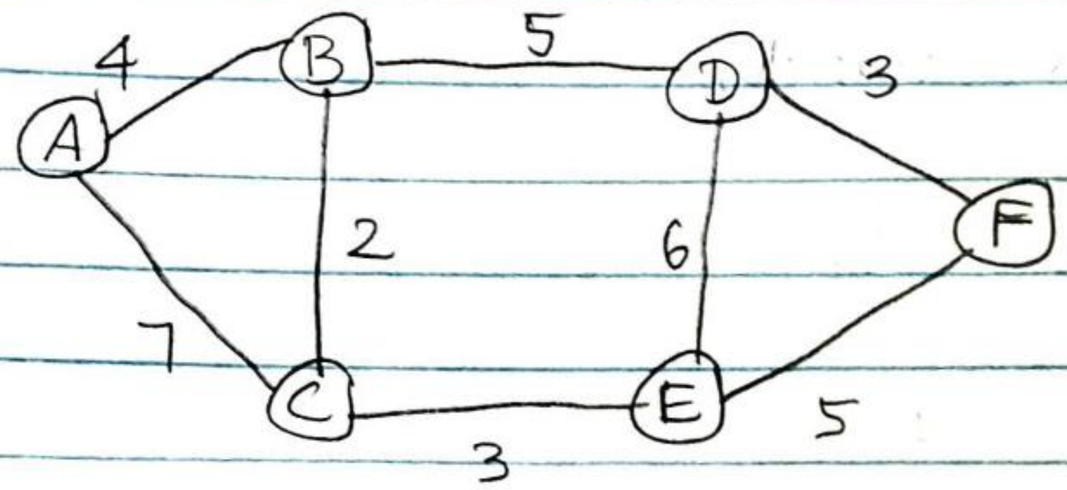


Step 11 :

(7)



13)



Initial Configuration

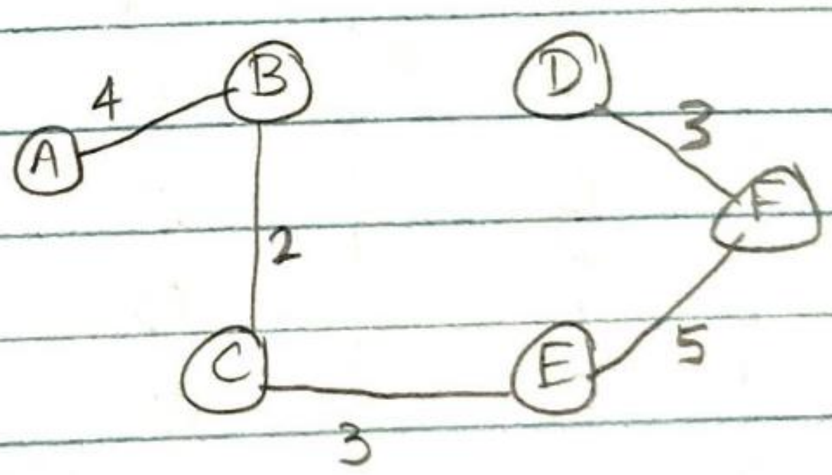
Vertices	known	d_v	P_v
A	0	0	0
B	0	∞	0
C	0	∞	0
D	0	∞	0
E	0	∞	0
F	0	∞	0

After A is selected

Vertices	Known	dv	Pv
A	1	0	0
B	0	4	A
C	0	7	A
D	0	∞	0
E	0	∞	0
F	0	∞	0

After B is selected

Vertices	Known	d_v	P_v
A	1	0	0
B	1	4	A
C	0	2	B
D	0	5	B
E	0	∞	0
F	0	∞	0



After C is selected

Vertices	Known	dv	Pv
A	1	0	0
B	1	4	A
C	1	2	B
D	0	5	B
E	0	3	C
F	0	∞	0

After F is selected

Vertices	Known	d_v	P_v
A	1	0	0
B	1	4	A
C	1	2	B
D	0	5	B
E	1	3	C
F	0	5	E

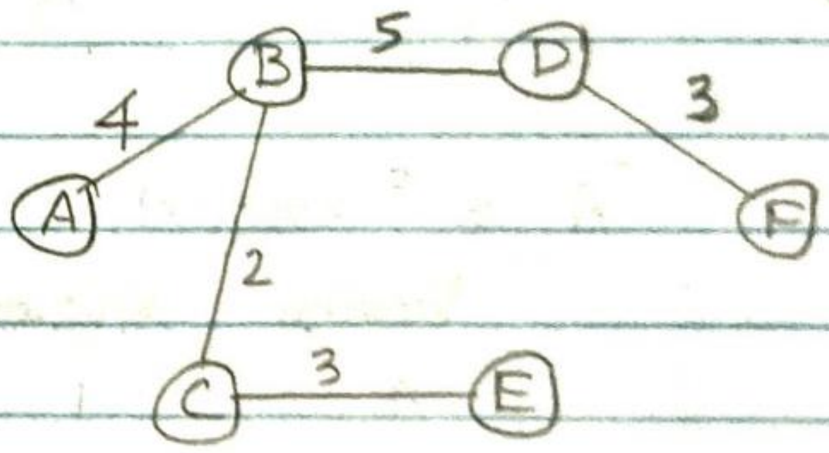
After D is selected

Vertices	Known	d_v	P_v
A	1	0	0
B	1	4	A
C	1	2	B
D	1	5	B
E	1	3	C
F	0	3	D

After F is selected

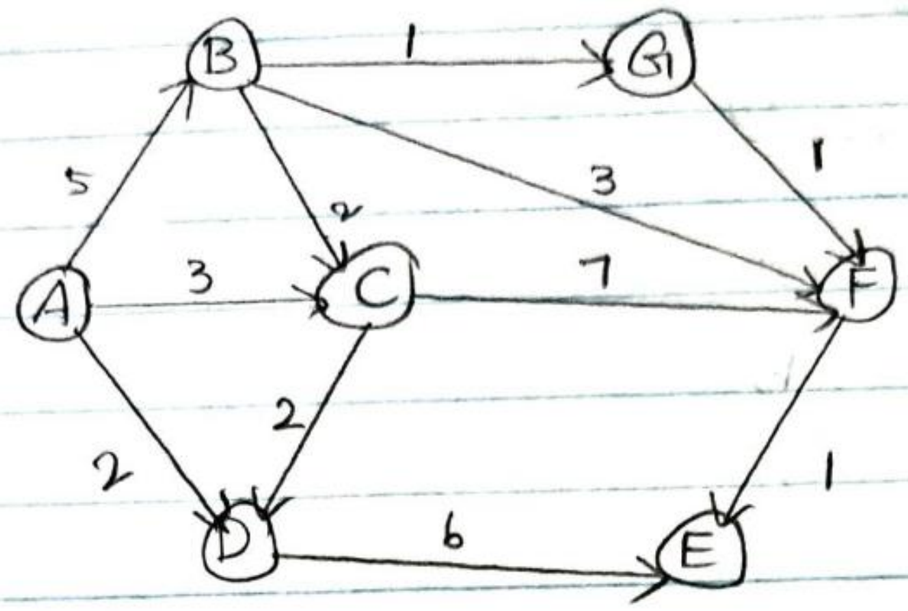
Vertices	Known	d_v	P_v
A	1	0	0
B	1	4	A
C	1	2	B
D	1	5	B
E	1	3	C
F	1	3	D

Output



Total Cost = 4+2+5+3+3 = 17

12)



Initial Configuration

vertices	known	d_v	P_v
A	0	0	0
B	0	∞	0
C	0	∞	0
D	0	∞	0
E	0	∞	0
F	0	∞	0
G	0	∞	0

A is Selected

Vertices	Known	d_v	P_v
A	1	0	0
B	0	5	A
C	0	3	A
D	0	2	A
E	0	∞	0
F	0	∞	0
G	0	∞	0

D is selected

Vertices	Known	d_v	P_v
A	1	0	0
B	0	5	A
C	0	3	A
D	1	2	A
E	0	8	D
F	0	∞	D
G	0	∞	D

C is selected

Vertices	Known	d_v	P_v
A	1	0	0
B	0	5	A
C	1	3	A
D	1	2	A
E	0	8	D
F	0	10	C
G	0	∞	0

B is selected

Vertices	Known	d_v	P_v
A	1	0	0
B	1	5	A
C	1	3	A
D	1	2	A
E	0	8	D
F	0	8	B
G	0	6	B

After G_1 is Selected

Known	Known	d_v	P_v
A	1	0	0
B	1	5	A
C	1	3	A
D	1	2	A
E	0	8	D
F	0	7	G
G	1	6	B

After F is selected

Vertices	Known	d_v	P_v
A	1	0	0
B	1	5	A
C	1	3	A
D	1	2	A
F	0	8	D
F	1	7	G
G	1	6	B

After E is selected

Vertices	Known	d_v	P_v
A	1	0	0
B	1	5	A
C	1	3	A
D	1	2	A
E	1	8	D
F	1	7	G
G	1	6	B

A → B, cost = 5
A → C, Cost = 3
A → D, Cost = 2
A → E, Cost = 8
A → F, cost = 7
A → G, cost = 6

4)