Name: Vidha Pandey
Father's Name: HC Pandey
Univ. Roll No: 2013576
Course: B. Tech (CS-DS)
Sem: VI
Paper Name: Design & Analysis of Algorithm
Paper Code: TCS 610
Type of Paper: Regular.
Signature: Pandey

Question.
1(a)

BFS

Breadth First Search algorithm traverses a graph in a breadth-ward motion & uses a queue to remember to get the next vertex to start a search when dead end occurs in any iteration.

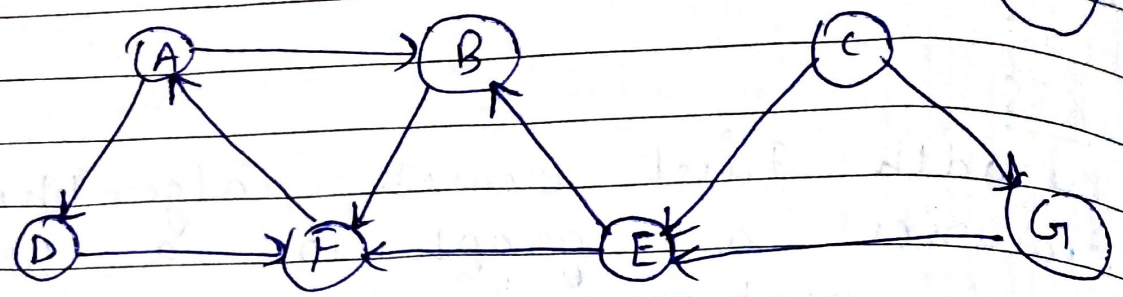
DFS

Depth First Search algorithm traverses a graph in depthward motion & uses a stack to remember to get the next vertex to start a search when a dead end occurs in any iteration.

Parameters	BFS	DFS
Data Structure	queue	stack
Source	BFS is better when target is closure	DFS is better when target is far.
Speed	Slower than DFS	Faster than BFS.
Time Complexity	O(E+V) V → vertices E → edges	O(V+E) V → vertices E → edges

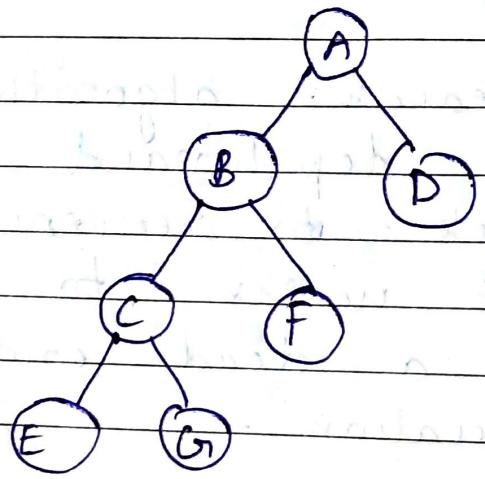
Pandey.

2



A	B	D	C	F	E	G

BFS Tree.



Min hop.

From	A
A	0
B	0
C	1
D	0
E	2
F	1
G	2

Pandey

Question 2. (a)

(a) $T(n) = T\left(\frac{n}{8}\right) + n^{1/3}$

$a = 1 \qquad b = 8 \qquad k = \frac{1}{3} \qquad p = 0$

$a < b^k$

$1 < (8)^{1/3}$

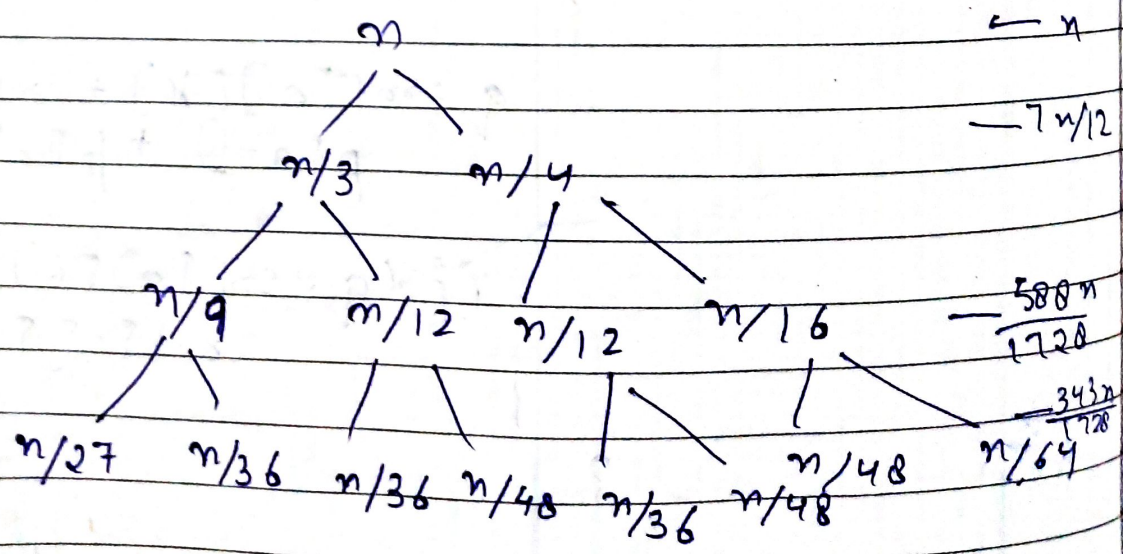
$1 < 2$

$p \geq 0$

Time Complexity $= O(n^{1/3} \log^0 n+1)$

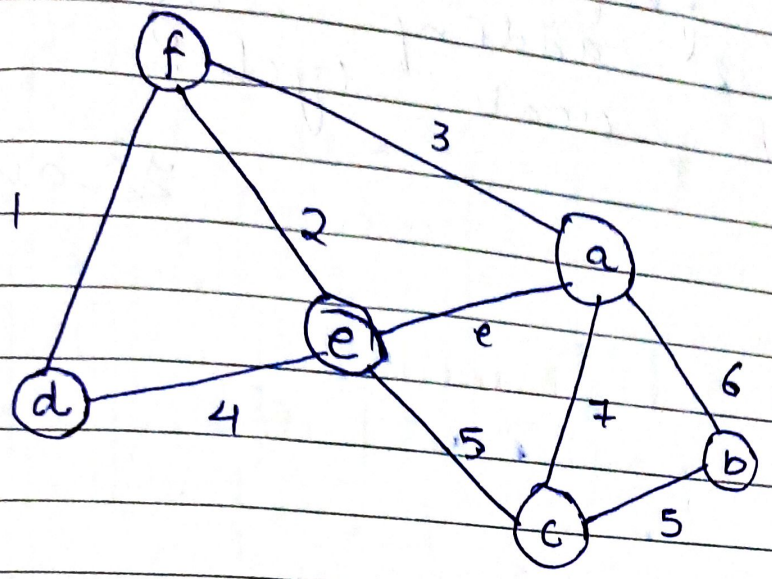
$= O(n^{1/3} \log n)$

(b) $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{n}{4}\right) + 5n$



Sandy.

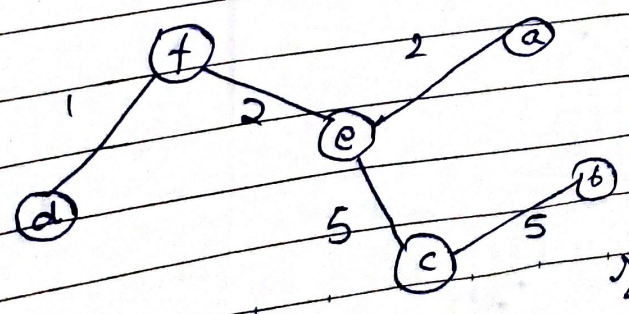
Question 3.
(b)



Step 1.
sorting edges in increasing order of their weights

u	v	w
d	f	1
f	e	2
e	a	2
f	a	3
d	e	4
e	c	5
c	b	5
a	b	6
a	c	7

Step 2.



$1 + 2 + 2 + 5 + 5 = 15$

creating graph with only vertices & adding edges which do not create cycle.

Yandy.

Question
4 (a)


```
int MatrixChainMultiplication (int *p,
int i, int j)
{
    int n=j+1;
    int sol[n][n];
    int a, b, k, l, q;

    for(a=1; a<n; a++)
        sol[a][a]=0;

    for (l=2; l<n; l++)
    {
        for(a=1; a<n-l+1; a++)
        {
            b = a+l-1;
            sol[a][b] =INT_MAX;
            for(k=a; k<=b-1; k++)
            {
                q =sol[a][k] + sol[k+1][b] +
                    p[a-1] * p[k] * p[b];

                if (q<sol[a][b])
                    sol[a][b]=q;
            }
        }
    }
    return sol[1][n-1];
}
```

Yfandy.

Question 5 (b)

p-problems	np problems.
(1) set of problems which can be solved in polynomial time by deterministic algorithms.	problems which can be solved in non-deterministic polynomial time.
(2) The problem belongs to class p if its easy to find the solution.	If its easy to ~~find~~ check a solution that may have been very tedious to find.
(3) can be solved & verified in polynomial time	solution to np problem cannot be obtained in polynomial times, but if its solution is given it can be verified in polynomial time.
(4) subset of np problems.	superset of p problems.
eg; selection sort, linear search.	TSP, knapsack.

Pandey.

Subset sum problem:

```
# include<stdio.h>
bool subsetsum(int set[], int n,
        int sum)
{
    if (sum==0)
            return true
    if (n==0)
            return false
    if (set[n-1] > sum)
            return subsetsum(set,
                n-1, sum);

    return subsetsum(set, n-1, sum) ||
        subsetsum(set, n-1, sum-set[n-1]);

}

int main()
{
    int set[]={3, 34, 4, 12, 5, 2};
    int sum=9.
    int n = sizeof(set)/sizeof(set[0])
    if (subsetsum(set, n, sum) == true)
            printf("found subset");
    else
            printf("not subset found");
    return 0;
}
```

o/p => found subset.

Pandey

Question 6.

(a) Hashing is the process of mapping large amount of data item to smaller table with the help of hashing function.
It is a technique to convert a range of key values into a range of indexes of an array.
It is used to facilitate the next level searching method when compared with the linear or binary search.

Collision occurs when more than one keys (hash functions) map to the same location of hashes. In this situation, two or more data elements qualify to be mapped to the same location in hash table.
Solution to hashing ⇒ choosing a better hash function.

To handle collision we use collision resolution techniques.
The process of finding an alternate location for a key in the case of collision is called collision resolution.

Pandey

(1) Chaining (Open Hashing / Separate chaing)

→ technique creates a linked list to the slot for which collision occurs.
→ the new keys is then inserted in the linked list.
→ These linked lists to the slots appear like chains.

$$\text{Load factor}(\alpha) = \frac{\text{no. of elements in hash table}}{\text{total size of hash table}}$$

(2) Open addressing

In open addressing all the keys are stored in the hash table itself. This approach is also called closed chaining.

(1) Maximum no. of keys are directly proportional to size of the table.

(2) Load factor [0, 1]

(3) If there is a collision, then

Pandey

with minor modification in hash function, a different index slot is created.

(4) Probing is finding the slot index, so in open addressing we can probe many no. of time.

Techniques used:

- Linear Probing:
  → we linearly probe for next bucket.
  → we keep probing until an empty bucket is found.

- Quadratic Probing:
  → we probe for i^2th bucket in i^{th} iteration.
  → we keep probing until empty bucket is found.

- Double Hashing:
  → we use another hash function hash2(x) and look for i^{th} hash2(x) bucket in i^{th} iteration.
  → It requires more computation time as two functions need to be computed.

Pandey