Set-IV

9. 3 / 4 — Travelling sales-10

5 / 6 — a) P class NP class (5) } Different
b) NP hard, NP Complete (5)

Set-III

1 / 2 — a) algorithm for Connected Components & spanning trees
b) procedure to identify Connected components

3 / 4 — a) Control abstraction } Branch & bound
b) general method

Set-II

1 / 2 — a) in, pre, post
b) Compare prim's, krushkals

9. 3 / 4 — 0/1 knapsack using branch bound

Set-I

1 / 2 — a) BFS
b) DFS } 4

5 / 6 — a) Cook's theorem
b) 3SAT problem in NP

Set-IV (5Q)

(b)

NP Complete	NP hard
* NP Complete problems are decision problem	* NP hard problem might not be decision problem
* harder problems	* hardest problems
* it is in NP	* may not be in NP
* problem X is NP Complete if NP prblm y is reducible to X in polynomial time	* problem x is NP hard if NP Complete problem y is reducible to x in Polynomial time
* Ex: 3SAT prblm	* Ex: Halting problem

(a)

P class	NP class
* if a prblm can be solved in time polynomial in the size of the input it is Considered an "easy" problem	* Technically "if a prblm can be solved in polynomial time by a non-deterministic turing machine, then it is in class NP
* class P contains problems which are solvable in polynomial time	* class NP Consists of those problem that are verifiable in polynomial time

Set-III

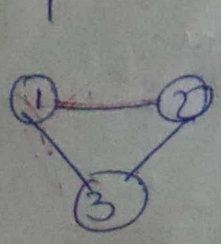
(1a)

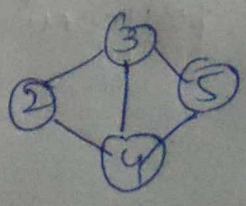
Connected Components

```
Algorithm Conn Components (G,?)
{
for i ← 1 to n do
{
 visited [i] ← 0
}
for i ← 1 to n do
{
if (visited [i] == 0) then
DFS(i);
output the newly visited
vertices with adjacent edges;
}
}
```

(1b)

* A undirected graph is called Connected graph if there is a path b/w any 2 vertices
* A Connected Component is a subgraph in which any two vertices are connected to each other by paths & which is connected to no additional vertices of the super graph





(3a) set-II

Control Abstraction:

```
Algorithm branch&bound ()
{
E ← new(node);
while (true)
{
if (E is a final leaf) then
{
write ("path from E to the root");
return;
}
Expand (E);
if (H is empty) then
{
write ("there is no soln");
return;
}
E ← delete_top(H);
}
}
```

(3b) General method (matter)

Set - II

(11a)

preorder - pLR
Inorder - LpR
postorder - LRp

Algorithm

```
tree node := record
{
type data;
tree node* lchild;
tree node* rchild;
}
```

Common for all

pre order

```
Algorithm preorder(t)
{
if (t≠0) then
{
visit(t);
preorder(t→lchild);
preorder(t→rchild);
}
}
```

Inorder

```
Algorithm inorder(t)
{
if (t≠0) then
{
inorder(t→lchild);
visit(t);
inorder(t→rchild);
}
}
```

post order

```
Algorithm postorder(t)
{
if (t≠0) then
{
postorder(t→lchild);
postorder(t→rchild);
visit(t);
}
}
```

(1b) Set-11

Prims	Kruskals
* it starts to build to minimum spanning tree from any vertex in the graph	* it starts to build the minimum spanning tree from the vertex carrying minimum weight in the graph
* it traverses one node more than one time to get minimum distance	* it traverses one node only once
* Time Complexity - $O(V^2)$	* time Complexity - $O(E \log V)$
* it generates the minimum spanning tree starting from the root vertex	* it generates the minimum spanning tree starting from the least weighted edge
* Applications 1. Travelling sales person 2. Network for roads and rail tracks connecting all cities	* Applications 1. LAN Connection 2. TV network

Set - 1

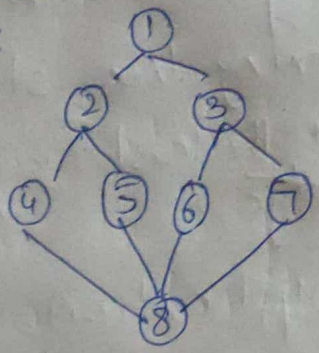
1@ Graph traversal
- BFS, DFS

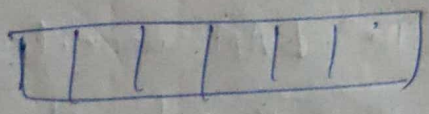
BFS

Breadth first search

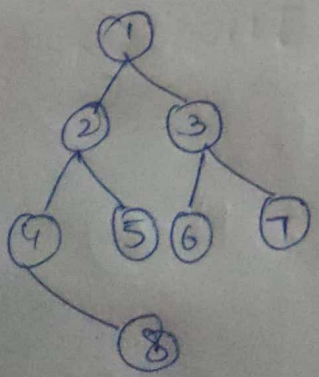
Algorithm BFS(G,n)
{
for i:=1 to n do
visited[i]:=0;
for(i:=1 to n)do
if(visited[i]==0)then
BFS(G,i);
}

Ex:





O/P:



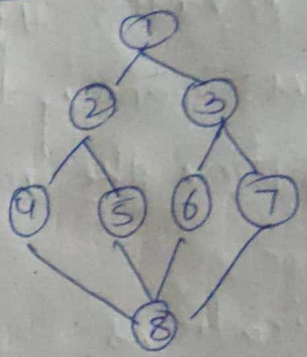
1 2 3 4 5 6 7 8

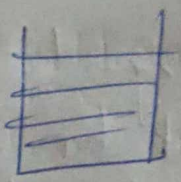
1b

DFS

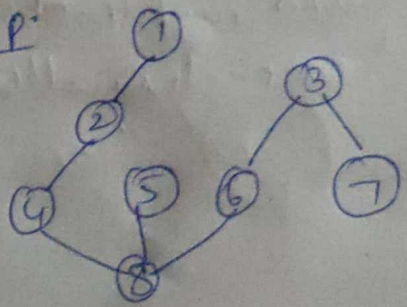
depth first search

Algorithm DFS(G,v)
{
v.visited:=true;
for each v∈G Adj[v];
if(v.visited:=false)
DFS(G,v);
}





O/P:



5 7 3 6 8 4 2 1

Set-1

6 a

Cook's theorem:

Scientist steaphen Cook in 1971 stated that boolean satisfiability problem is NP-Complete

proof:

This algorithm can also determine the values of expression for corresponding assignment and can accept the entire expression is true

this algorithm is composed by

1. input type
2. Read/write head
3. each cell continues only one symbol at a time
4. Computation is performed in no. of states
5. The algorithm terminates when it reaches the accept state

The conjuction clauses for boolean expressions are given in following table

Clauses	meaning	Time Complexity
T_{ijo}	cell i of i/p tape contains symbol j	$O(p(n^2))$
Q_{30}	intial state	$O(1)$
H_{00}	intial position of tape head	$O(1)$
$T_{ijk} : T_{ij(k+1)} \vee H_{ik}$	tape remains unchanged unless written	$O(p(n^2))$
$Q_{qk} \rightarrow \sim Q_{qk}$	one state at a time	$O(p(n))$
$H_{ik} \rightarrow \sim H_{ik}$	one read/write head position at a time	$O(p(n^2))$

Note: H – head
Q – states
T – tape

disjunction clauses for the boolean expression

clauses	meaning	Time Complexity
$(H_{ik} \wedge Q_{qk} \wedge T_{ijk}) \rightarrow H_{(i+j)(k+1)} \wedge Q_{q(k+1)} \wedge T_{ij(k+1)}$	possible transitions	$O(p(n^2))$
disjunction of all clauses Q	moving to accept state	$O(1)$

if B is satisfiable then there is an accepting state in the algorithm. This the proof shows that boolean satisfiability problem solved in polynomial time. Hence all prblm in NP could be solved ϕ in polynomial time and hence class NP could be equal to P

Note: H - head
Q - states
T - tape

disjunction clauses for the boolean expression

clauses	meaning	Time Complexity
$(H_{ik} \wedge Q_{qk} \wedge T_{ijk}) \rightarrow H_{(i+j)(k+1)} \wedge Q_{q(k+1)} \wedge T_{ij(k+1)}$	Possible transitions	$O(p(n^2))$
disjunction of all clauses Q_i	moving to accept state	$O(1)$

if B is satisfiable then there is an accepting state in the algorithm. This the proof shows that boolean satisfiability problem solved in polynomial time. Hence all prblm in NP could be solved in polynomial time and hence class NP could be equal to P

Set - II

(6b)

3SAT problem

3 SAT problem which takes a boolean formula 's' in CNF form with each clause having exactly three literals and check weather S is satisfied or not

~~follow~~ formula for instance of 3SAT prblm

$S = (\bar{a} + b + \bar{g}) \cdot (c + \bar{e} + f) \cdot (\bar{b} + d + \bar{f}) (a + e + \bar{b})$

theorem: 3 SAT prblm in NP

proof: let 's' be the boolean formula having 3 literals in each class for which we can construct a simple non-deterministic algorithm which can guess an assignment if boolean values to 's' is evaluated as 1 then 's' is satisfied and we can say that 3SAT ϕ is NP Complete