Q.1 Describe problem definition of multi commodity flow in the network. State & prove the Ford fulkerson's theorem.

Ans- Multi Commodity Flow :- It is a flow network with multiple goods/commodities flowing through the network, with different source and sink nodes.

- Multi-commodity flow network is a directed graph $G(V,E)$ such that :-
- For every edge $(u,v) \in E$ a non-negative capacity is associated, denoted by $c(u,v)$. If (u,v) is not in E, $c(u,v) = 0$.
- There is a distinct set S of vertices called sources. A vertex with no incoming edge, only outgoing edges, is called sources $(S_1, S_2 --- S_n)$.
- There is distinct set T of vertices called sink. T is proper subset of V, that is $T \in V$. In this set, all sinks are indexed, that is if we have n sinks, then $\{S_1, S_2 — S_n\}$.

Ford-Fulkerson's Method :- The method says that find a path from sources to sink & send max. flow through possible through it. Repeat this until no such path is left.

```
Ford-fulkerson (G, c, s, t)
  where G = flow network
        c = capacity function of G
        s = source , t = sink
Step-1  for each edge (u, v) in E
            f(u, v) = 0
            f(v, u) = 0
Step-2      G = Gf
Step-3  While there is a path p from s to t in Gf
        Repeat step 4 to 5
Step-4    cf(p) = min { cf(u,v) : (u, v) is in p }
Step-5    for each edge (u, v) in p
          f(u, v) = f(u, v) + cf(p)
          Cf(u, v) = Cf (u, v) - cf(p)
          Cf(u, v) = cf (v, u) + Cf (p)
Step-6    Return f.
```

- Briefly Describe flow shop scheduling & network capacity assignment problem.

Flow Shop Scheduling :— The objective of flow shop scheduling is to schedule operations of jobs so that:
Time consumption is less than or equals to some constant T,
Time consumption is smallest possible.

The FSS is specified by 3 fields: α, β, γ
α = the structure of problem
β = It cumulates a set of explicit constraints
Γ = Indicates objective to be optimized

Variants of shop scheduling:—

- Workshops with one machine
- Flowshop
- Jobshop
- Openshop
- Mined Workshop

Network Capacity Problem :— It is generally described as a special type of model, which is used in linear programming.
It includes problems such as:
- Transportation Problem
- Assignment Problem
- Shortest Path Problem
- Maximum Flow Problem

Terminology of NCP:—
(i) Nodes & Arcs
(ii) Arc Flow
(iii) Source
(iv) Sink
(v) Edge

Quadratic Assignment Problem (QAP):– The objective of QAP is to assign n facilities to n locations in such a way as to minimize the assignment cost. The assignment cost is the sum, over all pairs of the flow b/w a pair of facilities multiplied by distance b/w their assigned locations.

QAP can thus be written as $A = (a_{ij})$ & distance matrix $B = (b_{ij})$

$$\pi \in S_n < = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i)\,\pi(j)} b_{ij}$$

$$C_{ij} = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} x_{ij} \quad \text{or} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \cdot d_{\phi(i)\phi(j)}$$

$$x_{ij} = \begin{cases} 1, & \text{job assigned} \\ 0, & \text{otherwise} \end{cases}$$

$$\boxed{\sum_{i,j,k,l=1}^{n} f_{ik} d_{jl} x_{ij} x_{kl}}$$

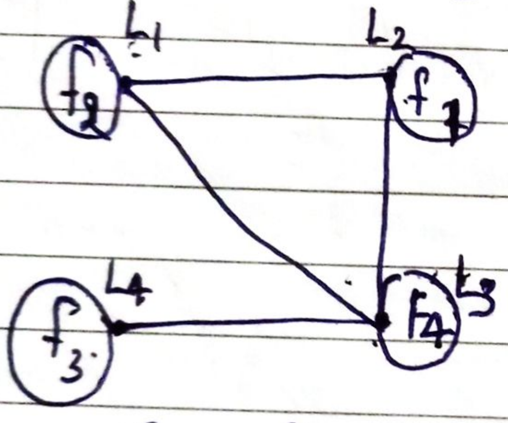
Q.

f_i	f_j	Flow(i,j)
1	2	3
1	4	2
2	4	1
3	4	4

l_i	l_j	distance (i,j)
1	3	53
2	1	22
2	3	40
3	4	55

$\{2, 1, 4, 3\}$

① ② ③ ④



$$\boxed{C = \sum_{i,j}^{n} f_{ij} d_{ij}}$$

$C = f(1,2) \cdot d(2,1) + f(1,4) \cdot d(2,3) + f(2,4)\, d(1,3)$
$+ f(3,4) \cdot d(3,4)$

$C = 3(22) + 2(40) + 1(53) + 4(55)$

QAP

$\boxed{C = 419}$ Ans

Q·1 Explain the term P, NP, NP hard & NP completeness. Also explain the relationship b/w them.

Ans- P class:-

This class of problem can be solved in polynomial time. "P" stands for "polynomial". The complexity class P is the set of all decision problems that can be solved in worst case polynomial time.

Suppose size of input to problem is 'n' then problem can be solved in the time $O(n^k)$ for some constant k.

A decision problem D is solvable in the class P, if there exists an algorithm A such that:

- A takes instance of D as inputs.
- A always outputs the correct answer "yes" or
- There exists a polynomial p such that the execution of A on inputs of size 'n' always terminates in $p(n)$

NP class:- NP stands for "Non-deterministic Polynomial time". NP is the class of decision problem that can be solved using a non-deterministic turing machine & a polynomial amount of computation time.

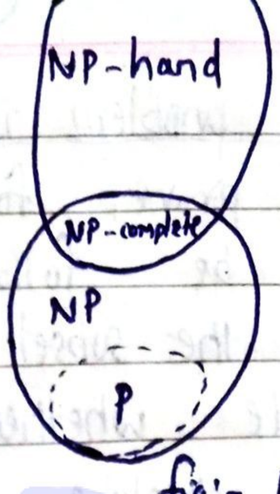
P is subset of NP

Some well known problems in NP are
- Finding Hamiltonian circuit through a graph
- Finding all subsets of set
- Partition problem
- TSP
- Graph Coloring

<u>NP complete class:—</u> It is class of problems which are both NP-hard & themselves member of NP. NP complete problems are hardest problems in NP set. A decision problem L is NP-complete if:

- L is in NP
- Every problem in NP is reducible to L in polynomial time.

<u>NP-hard class:-</u> NP hard therefore means "at least as hard as any NP problem" although it might, infact be harder. If an NP-hard problem can be solved in polynomial time, then all NP-complete problem can be solved in polynomial time.

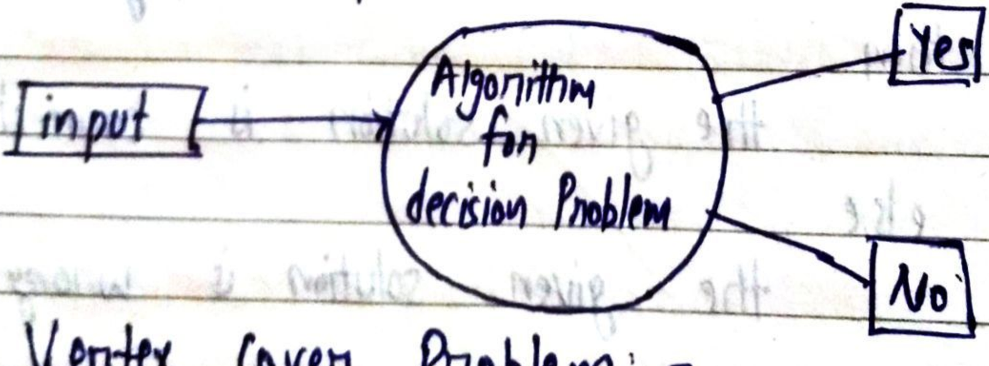


This diagram assumes $P \neq NP$

fig:- P vs NP vs NP-hard vs NP-complete

Q.2 What are decision problem? Prove that vertex cover problem is NP-complete.

Ans- A decision problem is one that solve problem only in "yes" or "no" form.
These problems are based for studying complexity theory. It asks only for yes or no answer. That means, output for algorithm for decision problem is either 1 or 0.



<u>Vertex cover Problem:-</u>
It is to find a vertex cover of minimum size in a given undirected graph.
A vertex cover of an undirected graph G(V,E) is subset of V' of V.

Vertex Cover problem is a NP problem.

In order to prove, following steps are supposed to be followed:

- First, take the subset V' as the certificate and decide whether it covers all the edges by checking all edges in polynomial time.

We can check whether the set V' is vertex cover of size k using the following strategy

```
Let count be an integer
set count to 0
for each vertex v in V'
  remove all edges adjacent to v from set E
    increment count by 1
    if count = k & E is empty
  then
          the given solution is correct
   else
          the given solution is wrong
```

It is plain to see that this can be done in polynomial time. Thus the vertex cover problem is in the class NP.

Q.3 Explain the cook's theorem with suitable example.

Ans- Cook's theorem states that the Boolean satisfiability problem is NP- complete. The complexity of theorem proving procedure following are 3 theorem by Stephen Cook -

Theorem-① If a set S of string is accepted by some non-deterministic TM within P time, then S is P- reducible to PNF teutologies.

Theorem-② The following are P-reducible to each other & hence each has the some polynomial degree of difficulty (teutologies), {PNF teutologies} {sub graph prise}

Theorem③ If set of strings are accepted by a non-deterministic m/c within time $T(n) = 2^n$ & if $T_Q(k)$ is an honest function of type P, then there is ∞ constant k, so S can be recognized by a deterministic m/c within time $T_Q(k\ 8^n)$

- First, Emphasized the significance of P time reducibility.
- Second, he foursed attention on class of NP of decision problems that can solven in P time by a NP complete.

Q.4 Prove that Hamiltanion cycle problem in NP-complete

Ans- To prove Hamiltanion cycle is NP-complete we first need to prove that HAM_CYCLE NP.

(i) To show that HAM_CYCLE ∈NP: If we can find an order of vertices such that each vertices is counted once except the starting vertex which is also the end vertex. Thus it verified that there is an edge b/w each consecutive vertices. This can be done in polynomial time so,

HAM_CYCLE ∈NP

(ii) Second step is to prove that vertex cover ∈P

Given a graph G & integer k, we redraw the graph below as follows:—

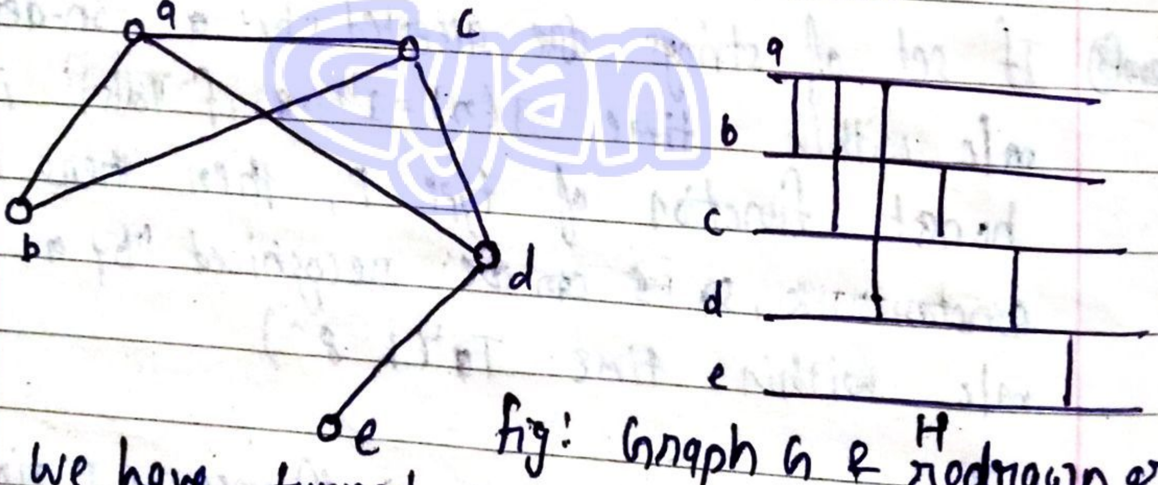


fig: Graph G & redrawn graph

We have turned every vertex in graph G to P horizontal line segment in it. The edges are represented by vertices lines called gates. Vertex cover in G of size k where k is horizontal lines that all gates in H. So how, pick k disjoint paths through that graph visits all gates

But prob. is that path might change end point so we need to make sure that all vertices are visited by path the don't start or end inside graph itself.

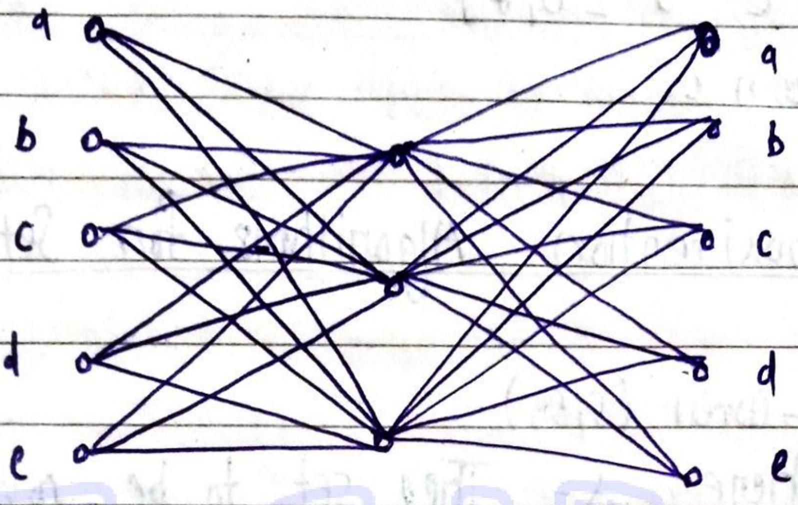


fig: Graph with Hamilton Cycle

Q.5 Explain Approximation algorithm for vertex & set cover problem.

Ans -

Approximation Algorithm for Vertex Cover ⇒

The aim is to cover the edges so alternatively we pick an arbitrary edge, put its end-vertices in cover; along with removal of those edge which have either of the vertices as end point.

- Vertex _ Cover (G)
where G = Graph with vertices V & edges E
C = Vertex Cover

1. C = ϕ
2. Until there is no edge left on E, repeat step 3 to 5

```
3.   Pick an edge (u, v) arbitrarily.
4.   Remove all edges from E of the form
       (u, w) or (v, w)
5.     C = C ∪ {u, v}
6.     Return C
```

Approximation Algorithms for Set Cover:—

```
   Set_cover (S, Sn)
     where S = The set to be covered
1.   C = ∅
2.   U = S
3.   Until U is empty, repeat step 4 to 6
4.     Pick set Si with max {Si ∧ U}
5.       U = U - Si
6.       C = C ∪ {Si}
7.   Return C
```

Q.6 Prove the circuit Satisfiability problem is NP Complete.

Ans CIRCUIT_SAT is NP complete

A problem that is in NP & has property that every problem is NP in polynomial time reducible to it is called NP-complete

Assume that we have a very simple model of a computer with register & a PC etc.

1. Let Q be a problem in NP
2. A certificate for a instance of Q can be written down in $O(s^i)$ bits, where i is the bit length of desc. of p→b instance.

3. There is a program A that runs on our simple m/c that takes a certificate for an instance of Q & verifies it in time $O(s^j)$ where $j > i$

4. The total time is $O(s^j)$ which means that space requirement is atmost $O(s^j)$ as well.

5. A complete description of machine running this would be in PC + register + "Program" + prog. instance + certificate + memory.

6. To figure out what the state will be at next clock-tick. we have some circuiting that has each bit of PC + reg. + Pro. + Prog. ins. + certificate + memory as i/p & o/p.

7. Now, to capture whole computation, we need to represent each of $O(s^j)$ steps, which would

mean that many copies of m/c configuration and circuiting b/w steps. So that total size of this circuit would be

$O(s^j) * O(s^{(j*k)}) = O(s^{(j*(k+1))})$.

This is big, but still polynomial in input.

8. Now we have ~~[illegible]~~ a single circuit that produce the entire computation of our simple m/c & its program that verifies certificates. And this circuit is polynomial in size! & we can construct it in polynomial time!

So we can construct it in polynomial time! So we have a polynomial time reduction of problem Q to CIRCUIT-SAT!

Hence Proved