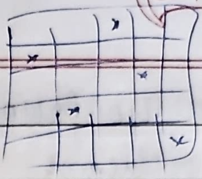
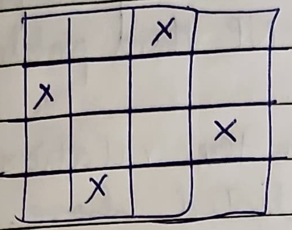
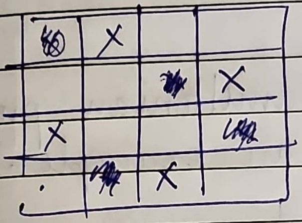
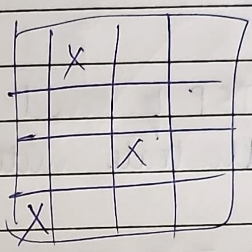
Unit-V Backtracking





n queens problem.

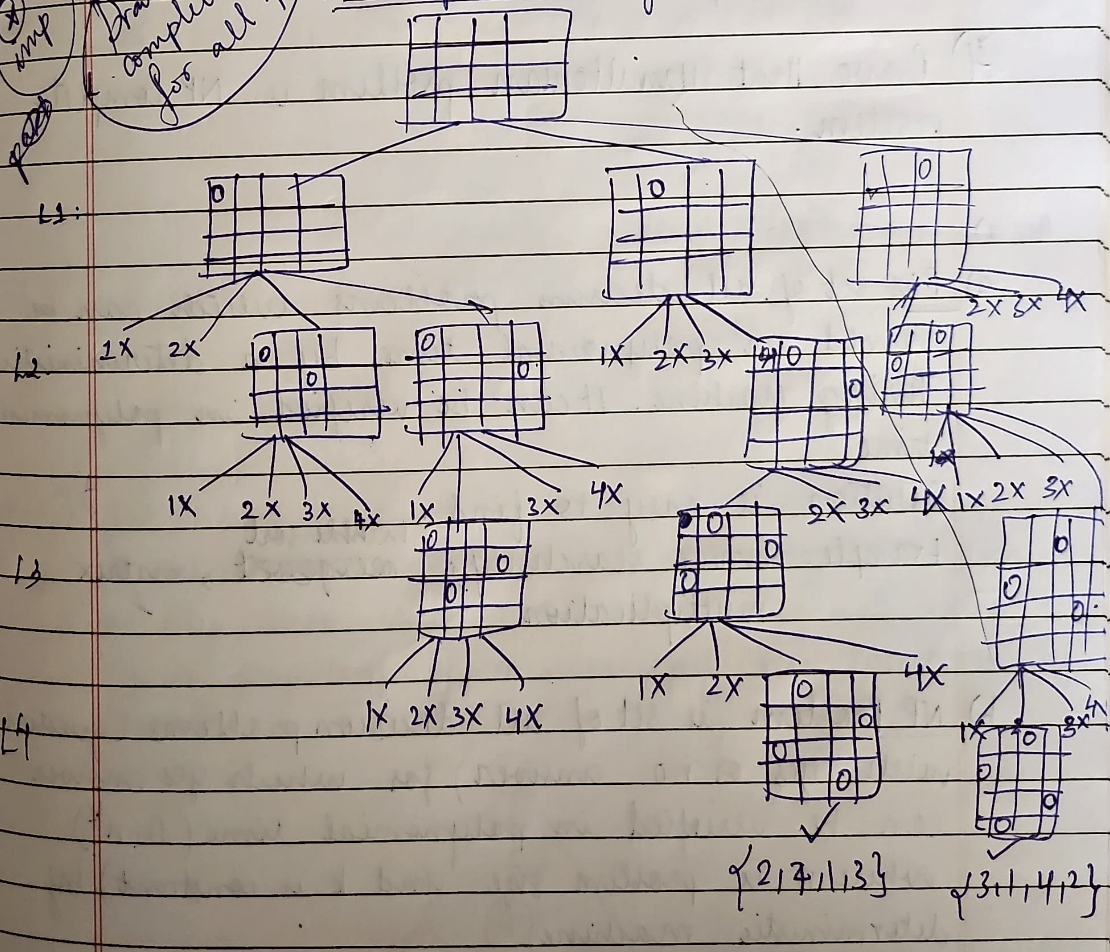




imp

Draw the complete thing for all 4.

State-Space Time of 4 Queens Problem



Q1) Define the following :
a) P probleam
b) NP problem
c) NP complete problem
d) NP Hard problem.

Illustrate with a diagram the relationship b/w the type of problems listed above. (Venn diagram)

Q2) What is TSP ~~p~~ and Hamiltonian problem? ~~List the~~ Illustrate with an example.

Q3) Prove that TSP (Travelling salesman Problem) is NP Complete problem.

Q4) Prove that Hamiltonian problem is NP complete problem.

Ans 1)
a) P is set of all decision problems which can be solved in polynomial time by a deterministic Turing Machine. It can be verified in polynomial time.
Solution is easy to find
Example: Linear searching, ~~mergesort~~ bubble sort, matrix multiplication

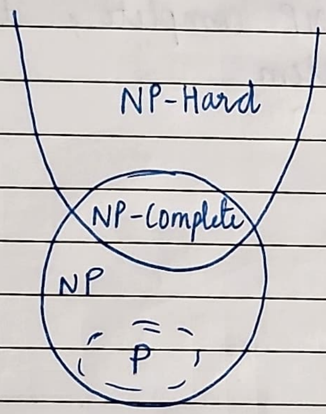
b) NP Problem is set of all decision problems (question with yes or no answer) for which yes answer can be verified in polynomial time ($O(n^k)$ where n is problem size and k is constant) by deterministic machine
Refers to Non deterministic Polynomial time.
Example: find prime factors of large numbers, N-Queens.
~~& Can't be executed in polynomial time~~

classmate
Date
Page

both NP and NP hard

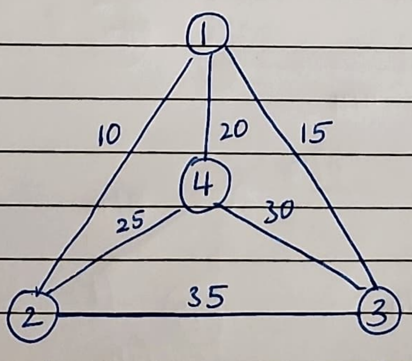
c) NP complete is a problem x in NP iff every other problem in NP can be quickly (ie. in polynomial time) transformed into x.
→ Every problem in NP is reducible to x
eg: Hamiltonian cycle, satisfiablity

d) NP Hard are problems that are atleast as hard as hardest problem in NP. Not all NP-hard problems are NP.
It takes longer time to check if solution is right or wrong
Example: Halting problem, circuit satisfactory



Venn diagram

Ans 2) TSP:
A salesman must visit a set of cities, each exactly once and return to original city, with objective of minimizing the total travel distance.

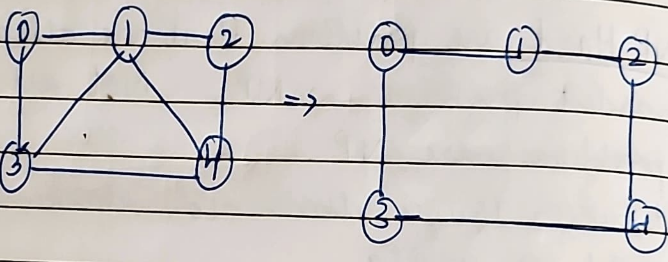


A TSP tour is 1 - 2 - 4 - 3 - 1 with total cost
10 + 25 + 30 + 15 = 80

Hamiltonian problem

To find all hamiltonian cycles in graph, which is a cycle ~~in a graph~~ that visits each vertex exactly once and returns to start vertex.

~~Egg~~ Eg:



$0 \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 0$

Ans 3) To prove TSP is NP complete, it must be NP and NP Hard problem

- NP: Given

Q) Define 3-SAT problem

Conditions: ① $0 \le f(e) \le c(e)$

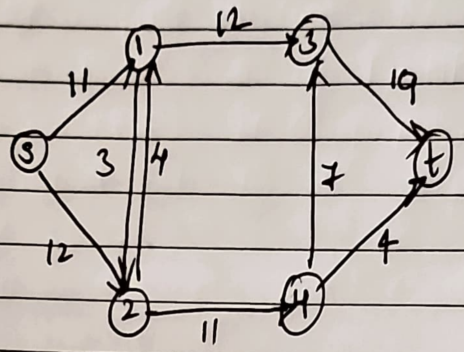
② Flow conservation

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$$ for all nodes except s & t

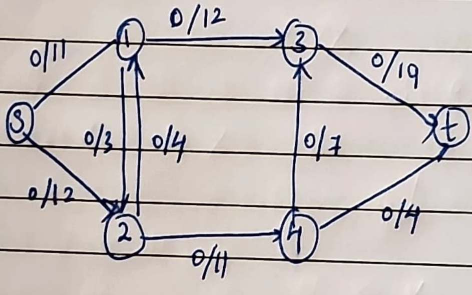
classmate
Date
Page

~~Determine~~

Determine the max. flow for given flow network using Ford Fulkerson Algo.

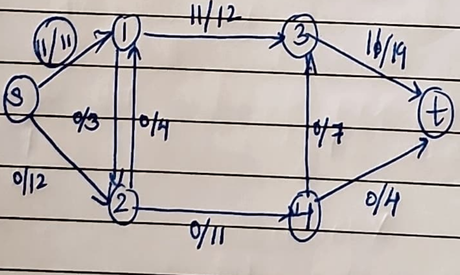


1

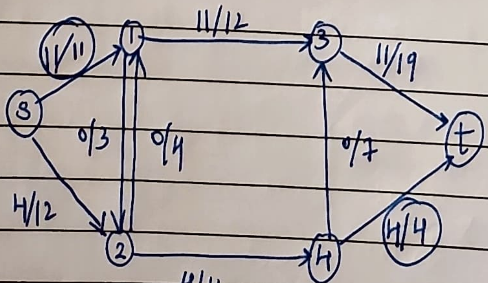


~~Draw the~~ graphs

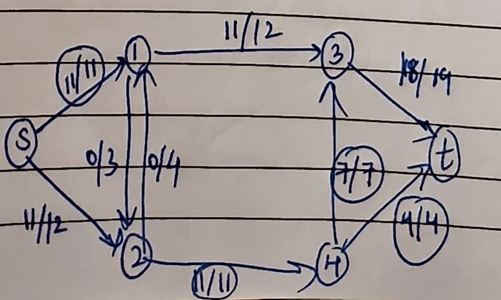
2



3

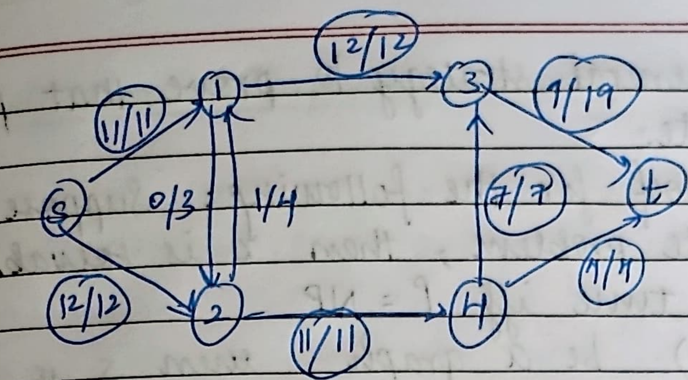


4



classmate
Date
Page

5



1/2 mark

Path no.	Augmenting paths	Flow
1	$s \xrightarrow{11} 1 \xrightarrow{12} 2 \xrightarrow{19} t$	11
2	$s \xrightarrow{12} 2 \xrightarrow{11} 4 \xrightarrow{4} t$	4
3	$s \xrightarrow{8} 2 \xrightarrow{7} 4 \xrightarrow{7} 3 \xrightarrow{8} t$	7
4	$s \xrightarrow{1} 2 \xrightarrow{4} 1 \xrightarrow{1} 3 \xrightarrow{1} t$	1
	~~Max Flow~~ No more augmenting paths. Hence stopped	23

Algorithm

Max Flow (G) {
Initially $f(e) = 0$ for all e in G
while there is s-t path in residual graph G
Let P be a simple s-t path in G
$f' =$ augment (f, P)
update f to be f'
~~Update~~ Update the residual graph G_f to be $G_{f'}$
End while
Return f

UNIT-V

Q1) ~~Eq~~ Explain the general strategy to prove that problems are NP complete.

Q2) Construct the proof for the following: Suppose x is a NP complete problem, then x is solvable in polynomial time iff P = NP

Q3) Let G = (V,E) be a graph, then S is an independent set iff its complement V−S is a vertex cover. (give general & example)

Q4) Explain how the circuit satisfiability is a NP complete problem

Ans 1) Given a new problem X, the basic strategy for proving it to be NP complete is as follows:

i) Prove that X belongs to NP

ii) choose a problem of Y that is known to be NP complete and

iii) Prove that $Y \leq_P X$.

Consider an arbitrary instance S_Y of problem Y and show how to construct in polynomial time with an instance S_X of problem X that satisfies the following properties:

(1) If S_Y is S instance of Y, then S_X is S instance of X.

(2) If S_X is S instance of X, then S_Y is the S instance of Y

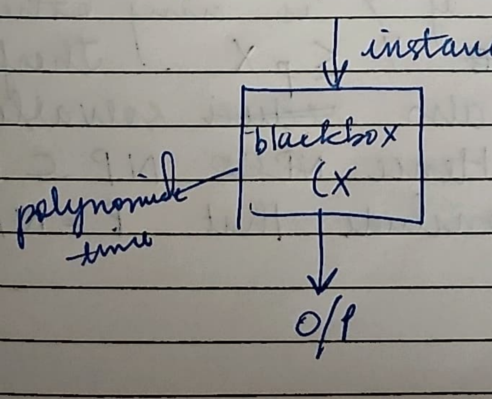
The polynomial time reduction with a special structure wherein asking the blackbox a single question and using its answer and based on the answer, we can query the blackbox multiple times to achieve the notion of polynomial time reduction.

Polynomial Time Reduction

The intention here is to explore the space of computationally hard problems and compare relative difficulty of different problems and formally express the statements
" Problem X is atleast as hard as Problem Y"
We will formalize this through the notion of reduction i.e. we will show that a particular problem X is at least as hard as some other problem Y by arguing that if we had a blackbox capable of solving X, then we can also solve Y.
In other words, X is powerful enough to solve Y. To make it more precise, we add the assumption that if X can be solved directly in polynomial time, then proving that arbitrary instanse of problem Y are solved using a polynomial no. of standard computational steps plus a polynomial no of calls to the black box that solves X, then we can write :

$$Y \leq_p X$$

ie Y is polynomial time reducible to X.



~~Suppose Y is polynomial polynom.~~
Suppose $Y \leq_p X$
if X can be solved in polynomial time then Y can also be solved in polynomial time.

Prove that (independent set $\leq_p$ Vertex cover)

If we have a blackbox to solve vertex cover, then we can decide whether G as an independent set of size atleast k by asking the blackbox whether ~~vecto~~ G as vertex cover has size n-k or |V|-k.

Q) Suppose ~~X~~ as an NP complete problem, the X is solvable in polynomial time iff P = NP.

"A problem is said to be NP complete if it satisfies the ~~pol~~ following two properties:
i) $X \in NP$
ii) for all $Y \in NP$, $Y \leq_p X$ "

Proof:
Claiming if P = NP, then X can be solved in polynomial time since it belongs to NP.
Conversely suppose that X can be solved in polynomial time if Y is any other problem in NP, then ~~Y~~ $Y \leq_p X$. Therefore, it follows that Y is also ~~reduc~~ solvable in polynomial time. Hence ~~NP~~ $NP \subseteq P$ where we can conclude that P = NP