☆ Algorithm ⇒ finite set of steps to solve a problem is called Algorithms.

☆ Analysis ⇒ It is a process of comparing two Algo.

★ Priory → execution से पहले की Analysis. (Approx. value)

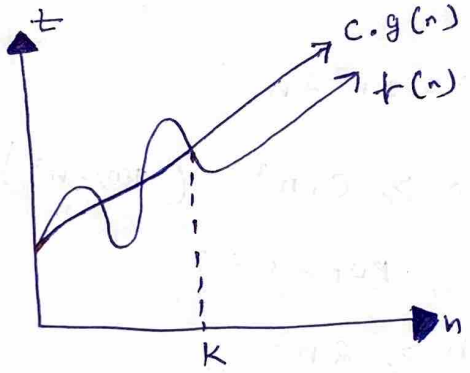
★ Posterion → execution के बाद की Analysis. (give exact value.)

☆ Asymptotic Notation

→ It is the methamatical way to Represent a time complexity.

① Big-oh (O)

-> worst case
-> upper Bound (At most)
least



$F(n) = O\ g(n)$

$f(n) \leq c \cdot g(n)$

$\begin{Bmatrix} C > 0 \\ n \geq K \\ K \geq 0 \end{Bmatrix}$

For ex.

$f(n) = 2n^2 + n$

$f(n) = O(\ )$ ($n = n^2$)

$2n^2 + n \leq C \cdot g(n^2)$

put $C = 3$

$2n^2 + n \leq 3n^2$

$n \leq 3n^2 - 2n^2$

$n \leq n^2$

(divide by $\frac{1}{n}$)

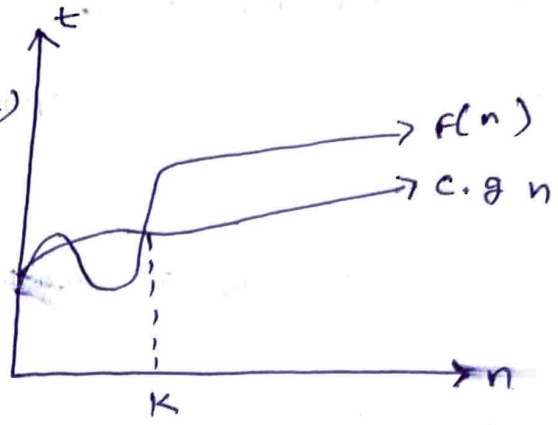
$1 \leq n$, $n \geq 1$

For all the value of n

(2.) Big - Omega (Ω)

→ Best case

→ Lower Bound (atleast)



$f(n) = \Omega\, g(n)$

$f(n) \geq c.g(n)$

for ex.

$f(n) = 2n^2 + n$

$2n^2 + n \geq c.n^2 \quad (g(n) = n^2)$

Put $c = 2$

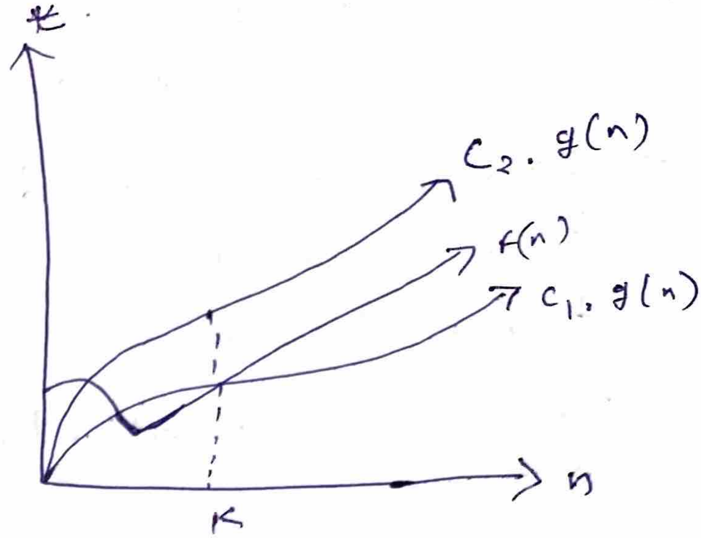
$2n^2 + n \geq 2n^2$

$n \geq 2n^2 - 2n^2$

$n \geq 0$

(3.) Theta (Θ)

Averag case
exact time



$$C_1 \cdot g n \le f(n) \le C_2 \cdot g(n)$$

$$2n^2 \le 2n^2 + n \le 3n^2$$

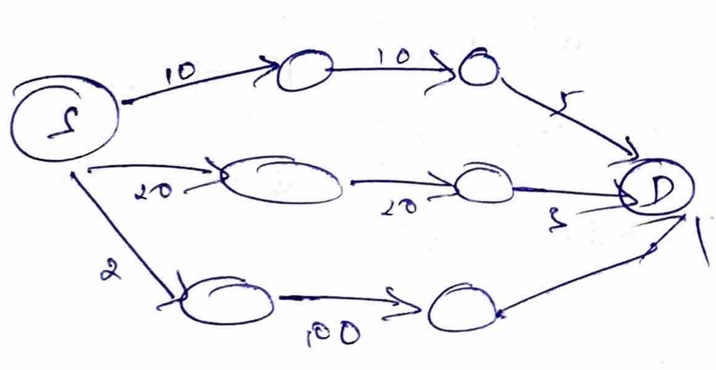
☆ Dynamic Programming

→ It is used to solve optimization problems

→ It divide a problem into series of overlapping sub problems.

Two features : (1.) optimal substructure
(2.) overlapping subproblems

→ It always give the optimal answer



Dynamic में सारे check करेगा ~~Solution~~ path [illegible]

connected
in and prove with

ndwidth
10^8m/sec
twork?

agreed tc
mitted by
ther corr

in Dynamic सारे path check होते then ही optimal solution मिलेगा.

-> Dynamic Programming में हम sub problems के solution को एक Table में store कर लेते हैं ताकि in future same sub problem के लिए direct solution मिल जाए without solve them.

☆ Greedy Technique

-> Follows local optimal choice of each stage with instead of finding global optimum.

-> Min cost
-> Max profit
-> Min Risk

-> feasible solution
-> optimal solution

Application:

1. Knapsack Problem
2. Job sequencing
3. Minimum Spanning tree
4. optimal merge pattern
5. Huffman coding
6. Dijkstra algo.

-> Dynamic and greedy Both work on optimization techniques. (का न minimum find करना हो तो min का max.)

	Greedy Method	Dynamic Prog.
(1.)	There is no principle of optaimality.	There is a principle of Optimality.
(2.)	It is a deterministic approach to get a solution for a problem.	It is not a deterministic approach to get a solution of a problem there we consider all the possible solution and be act the best solution or optimal solution.
(3.)	Greedy method never looking back or revising previous choices.	Dynamic prog look back or revising previous choices.
(4.)	More efficient compare to D.P	Less efficient compare to G.M.
(5.)	Knapsack problem	

Cooks Theorem

-> It stats that The Satisfiability problem (SAT) is NP-complete.

What is (SAT) ?

Cooks Theorem -> SAT is NP-Complete.

SAT-A propositional logic formula (Q) is called a satisfiable if there is a some assignment to its variable that makes it evaluate to true.

-> $p \wedge q$ is satisfiable (if $p=1$ $q=1$).

* Abstract decision problem = It can be viewed as a function that maps the instance set (I) to the solution set $\{0, 1\}$

★ P class Problem

→ Problem which can be solved on a Polynomial time is known as P-class Problem. (and also verified in P Polynomial time.)

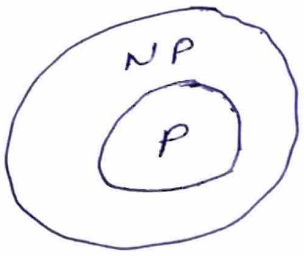
(ex.) All sorting and searching Algo

★ NP class Problem

→ Problems which cannot be solved on a Polynomial time but is verified in polynomial time is known as Non-Deterministic polynomial or NP-Class problem.

(ex.) Su-Do-Ku, Prime factor, Scheduling, Travelling Salesman

(in this solving a problem is difficult but verifing is easy)

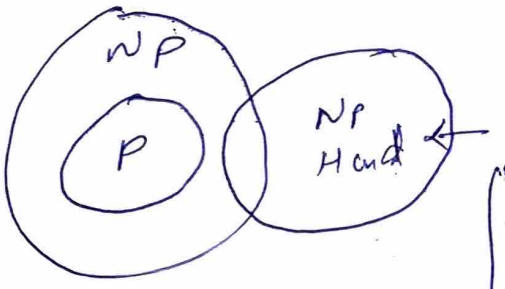


Those problems which are come under P are called Tractable problems

Those problems which are come under NP are called intractable problems

★ NP Hard problem

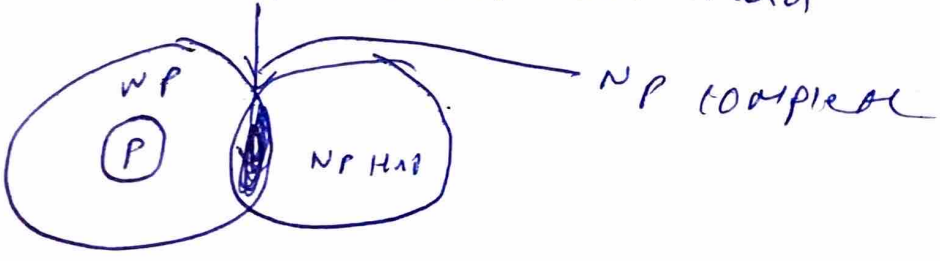
→ A problem is NP Hard if every problem in NP can be polynomial reduced to it,



intersection of NP Hard and NP is NP complete

★ NP complete problem

→ The problem is NP complete if It is in NP and it is in NP Hard



NP complete problems are decision problems

NP Hard problems are optimization problems.

All the NP complete problems are NP Hard. But All the NP Hard are not NP complete

☆ Linear Search	Binary Search
(1.) In linear search technique the array of element is sequentially traversed to locate the given item.	(1.) In Binary search technique the given item is first compared with mid item of array, If given item is not equal to mid item then searching is Performed once half of array. [above or below the mid item].
(2.) In linear search elements can store Randomly in Array list.	(2.) In Binary search elements should be stored in Sorted Array.
(3.) It is slower searching technique.	(4.) It is faster searching technique compared to linear search.

(4.) Worst case time complexity of linear search is $O(n)$	(4.) Worst case time complexity of binary search is $O(\log_2 n)$.
(5.) Best case time complexity of linear search is $O(1)$	(5.) Best case time complexity of Binary Search is $O(1)$
(6.) It can be implemented on Array and Linked list	(6.) It cannot be directly implemented on linked list
(7.) Iterative in Nature	(7.) Recursive in nature and based on divide and conquer Technique.
(8.) Easy to use and no need for any ordered element.	(8.) Tricky algorithm and elements should sorted in order
(9.) Less coding to write	(9.) More coding to write.

☆ Topological sort

जब भी हम graph की बात करेंगे for (ex.)

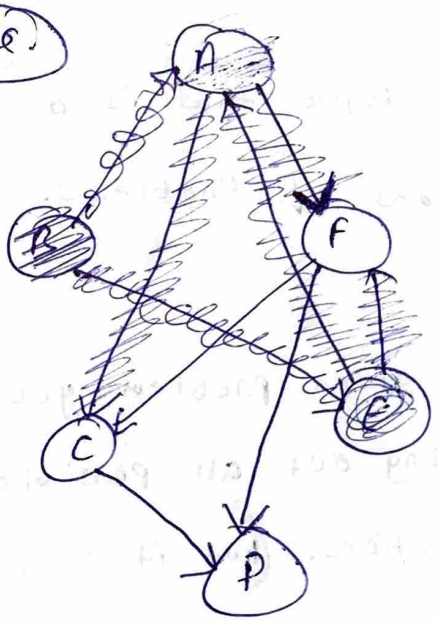
(u) ⟶ (v) And जब भी हम उसका order write करेंगे तो (u) हमेशा (v) से पहले आना चाहिए।

-> Applicable on DAG (Directed Acylic graph)

-> Linear Running Time complexity

[For (ex.) 2nd year में तब जाएंगे जब first year complete होगा]

(ex.)



Firstly

जो भी Arrow जिस भी node पे अंदर नहीं आ रहा एक भी Arrow इसे सबसे पहले दिखेंगे like (B)

फिर उसके जो भी Arrow पार हो वो हटा लेंगे then same for other node

B E A F C D

☆

(1.) State Space tree = It is a tree that Represent all of the possible states of the problem from the root as an initial state to the leaf as a terminal state.

(2.) Problem state = Each Node in the tree defines a Problem state.

(3.) Answer state = Answer state are those solution state (s) for which the path from the Root to (s) Defines a tuple that is a Member of the set of solutions of Problems.

(4.) Brute force Approach = for any given problem you should try out all possible solution and pickup desired solutions. (but it is not for Optimization problem)

(5.) Abstract problem = A function that maps the instance set I to the solution set O {0, 1}

(8.) Search space = It is a set of all possible valid moves.

(9.) ~~Search~~

(9.) Abstract Problem => Q is a binary Relation in a set(I) of a problem instance and a set(S) of problem solutions.

(9.) Dead node => It is not to be expander.

(10.) Dynamic tree => It is a set of dynamically changing rooted tree.

(11.) solution state => those state leading to a tuple.

Backtracking

Meaning = Go back the same way you come
पिछले रास्ते से वही रास्ते वापस जाना

-> In Backtracking we search depth first for solution.

-> In Backtracking technique we backtrack to the last valid path as soon as we hit a deadend.

-> Backtracking Reduce the search space since we no longer have to follow down any path we know are invalid.

-> Backtracking is an algorithmic technique that consider searching every possible combination in order to solve an optimization problem.

-> Backtracking is an approach to solving constraint satisfaction problem without trying all possibilities.

-> Backtracking technique is used for those problems which have many candidate solution but the majority of which donot satisfy the given constraints.

-> DFS Used.

-> जहाँ multiple solution आप ढूंढ रहे हो वहाँ Backtracking का use करना है।

⇒ Backtracking is the depth first search with some bounding function

⇒ In Backtracking method the solution can be expressed in a form of tuples.

$(x_1, x_2, x_3 \cdots x_n)$

☆ Branch and Bound

→ The Branch and Bound Algo. is similar to Backtracking but is used for ~~Activity~~ optimization problem.

⇒ Branch bound is usually work on minimization.

→ It perform a graph traversal on the state space tree by using BFS

→ There are three B-Bound Technique

FIFO Branch and Bound Technique (Queue)

LIFO Branch and Bound Technique (Stack)

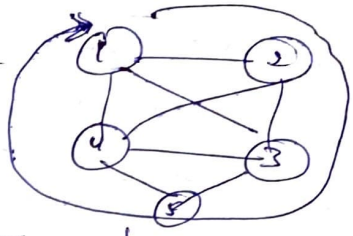
- Least Cost Branch and Bound.

vices connected
and prove with

width
m/se
rk?

ed t
ed b
corr

Hamiltonian Cycle



-> It is the (एक) of backtracking

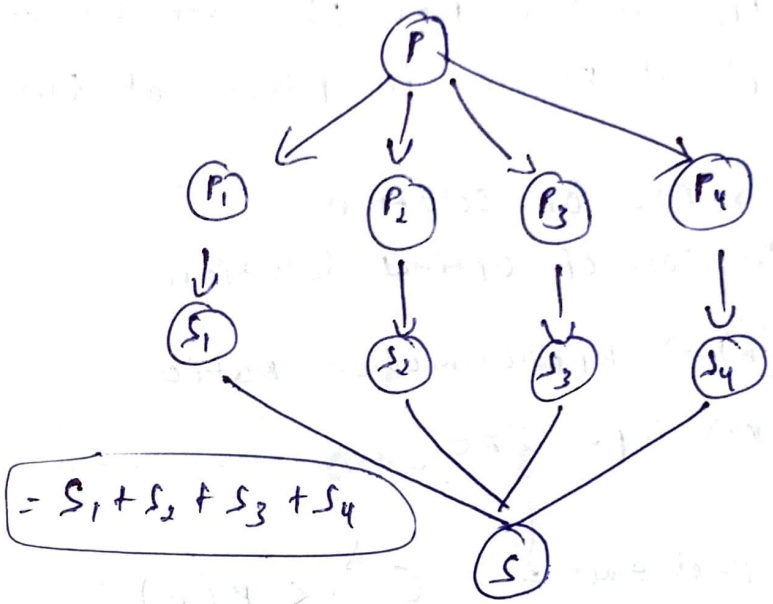
-> (i) Node से start करता है और सारे node cover करके दोबारा (i) Node पे end करता है।

-> Hamiltonian a cycle is a path in a given graph that visit each vertex exactly once and there is edge path b/w first and last vertex.

-> There may be any number of Hamiltonian cycle exist in a graph

★ Divide and Conquer

→ In this Approach we divide a big problem into sub problems then find a solutions of sub problems and then combine sub problems solution.



→ It is used to solve decision problem

→ Follows Top-down Approach

→ Subproblems are interdependent

→ Solution of sub problems complete Recursively More than one.

→ Recursive in Nature

★ Approximation Algo

→ This Algorithm is used to deal NP-Completeness for optimization problem.

→ The Goal of Approximation Algo is come close as possible to optimal solution in polynomial time.

C → cost of solution
C^* → cost of optimal solution
P(n) → Approximation Ratio
n → input size

Maximization problem = $\frac{C^*}{C} \leq P(n)$

Minimization problem = $\frac{C}{C^*} \leq P(n)$

$P(n) \geq 1$

☆ Vertex Cover Problem

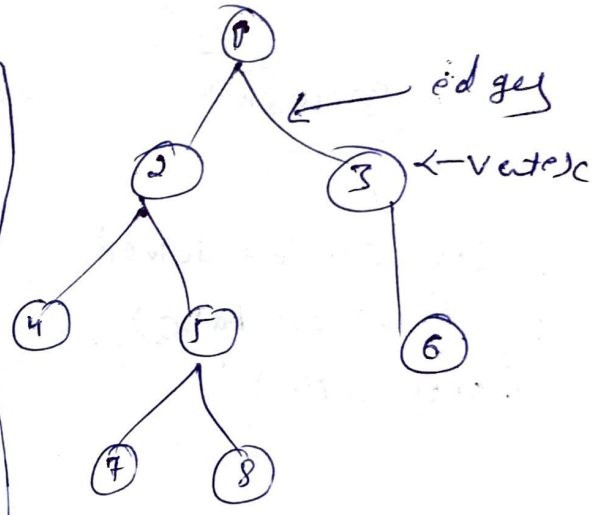
-> Vertex Cover Problem is the example of (App. Algo)

-> (हमें कम से कम Vertex Find करना है जो सारी edges को cover करे) को सभी को graph में।

-> A vertex cover of a graph is a subset of vertices which cover every edges.

-> Vertex cover Problem is the minimum size vertex cover.

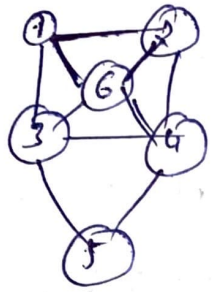
```
APPROX-VERTEX-COVER (G)
1  C ← ϕ
2  E' ← E[G]
3  while E' ≠ ϕ
4      do let (U,V) an arbitary
           edge in E'
5      C ← C ∪ {U,V}
       Remove from E' every
       edge incident on either
       U or V
6. return C
```



W = { 1, 3, 2, 5 }

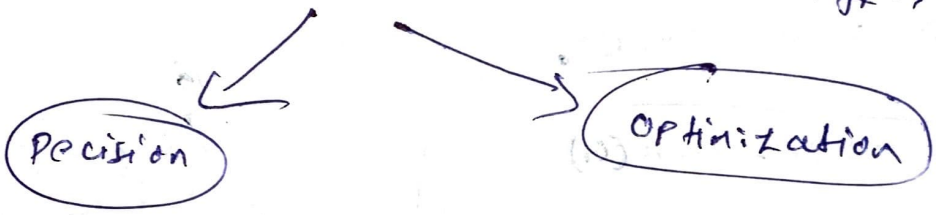
✡ Clique Problem

→ Complete sub graph of a graph is clique.



1-2-6 is a complete sub graph
It is a clique of size 3

→ A clique problem has two types



It can be solved by (True or False) (yes or NO)

It can be solved by maximum clique. size of graph and minimum.

→ NP completness of clique problem :

(1.) Clique problem is in NP

(2.) clique problem is NP-Hard

★ Stable Sorting

Consider An integer array that contains same integers, we are sorting the given elements of array by using a sorting algorithm, If the sorting algorithm does not change the sequence of similar elements in which they appear, then that type of sorting is called stable sorting.

ex.

	0	1	2	3	4	5	6	7	8	9	
	35	32	42	10	14	19	26	44	26	31	Before sorting
							A		B		
	10	14	19	26	26	31	33	35	42	44	after sorting
	0	1	2	3	4	5	6	7	8	9	

prove with

Knapsack Problem

→ Greedy about both
weigh
~~Profit~~ Profit

★ Union Find

=> T.C = O(log N), where (n) is the no. of nodes

S.C = O(n), where (n) is the no. of nodes.

★ Find: Suppose player (1) in the orange team is our friend, and we want to find them on the field during the Match, so we will skim through all the players on the field to find them,

similarly for two disjoint sets, the find operation is used to find particular element.

★ Union: Lets assume that the purple and orange teams are the best polo teams from our state, Now for a national competition we need to send eight players to represent the state,

since we know purple and orange teams

we best, we will combine them and send them as a team

This Describe the union operation.

★ In Degree out degree

Time complexity : $O(V+E)$

V → vertices
E → edges

★ Transitive closure (warshall algo used)

T.C = $O(N)$

★ TC of Quick sort

Best case $O(n * \log n)$

Worst case $O(n^2)$

Average case $O(n * \log n)$

★ Kruksal Algo

TC = $O(e \lg V)$

SC

★ selection sort T.C

worst and Best are same $O(N^2)$ $\theta(N^2)$ $\Omega(N^2)$

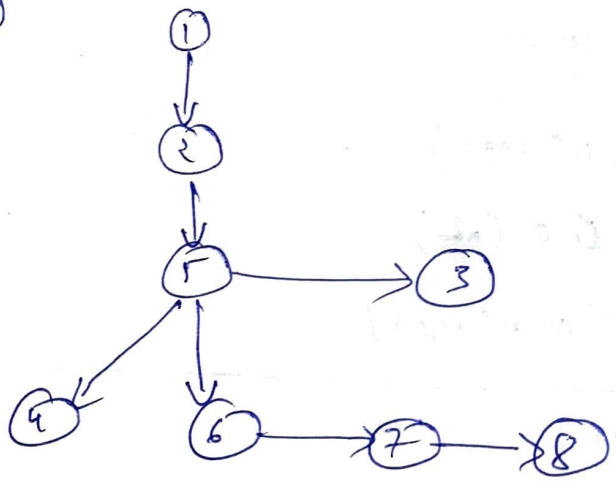
Graph

(1.) Forward edge => It is a non-tree edge (u,v) such that there exist a path b/w (u) to (v) in the tree.

(2.) Cross edge => It is a non-tree edge (u,v) which connect two nodes such that Neither of these is ancestor of the other in the Tree.

(3.) Back edge => It is a non-tree edge (u,v) such that (v) is ancestor of node (u) in the tree.

Ex.



Forward edge -> (1->3), (2->4)

Cross edge -> (4->6) (4->8) (3->7), (3->8)

Back edge -> (4->5)