SNS COLLEGE OF ENGINEERING

Coimbatore-107

COURSE NAME: ANALYSIS OF ALGORITHM

II YEAR/ IV SEMESTER

UNIT – V

BACKTRACKING ALGORITHM

Topic

NP Problem

Computational complexity classes:

P NP NP Hard NP Complete Problem Reduction

(i) **P class:** Time Complexity ($O(n)$, $O(n^2)$, $O(n^3)$)

*. Stands for Polynomial Time

*. Collection of decision problems (Yes/No)

*. It can be solved by deterministic (Regular Alg) machine in polynomial time.

*. Easy to solve & Tractable.

*. Solved in theory & practical.

Examples:

find GCD, Searching, find maximum Match

(ii). **NP class:**

*. It is not known, how to solve quickly. (Collection of decision Problems)

*. But if Solution is reached, Verification of Solution is easy.

*. Hard to Solve, But easy to check/verify.

*. It can be verified by Turing machine in polynomial time. But Cannot be Solved in polynomial time.

Example: Sudoku.

- Graph coloring
- Hamiltonian path problem
- Travelling salesman problem.

Note:

Based on given I/P & Graph, it may fall either in NP class or in P class.

(iii) NP Complete (Non Deterministic Polynomial time complete)

⇒ A problem is NP complete if it is both NP & NP hard.

⇒ These are hard problems in NP.

⇒ These are subset of larger class of NP

⇒ Any problem in NP class can be transformed / reduced into NP complete problems in polynomial time. (P = NP)

Example:

1) 0/1 Knapsack
2) Hamiltonian cycle
3)

(iv) NP Hard problems: (Non Deterministic)

*. These are computational problems atleast hard as problems in Np class.

*. Longer time to solve. But Solution can be verified in polynomial time.

*. It is harder than Np Complete problem & Solved by Problem reduction techniques. (Divide & conquer, Backtracking).

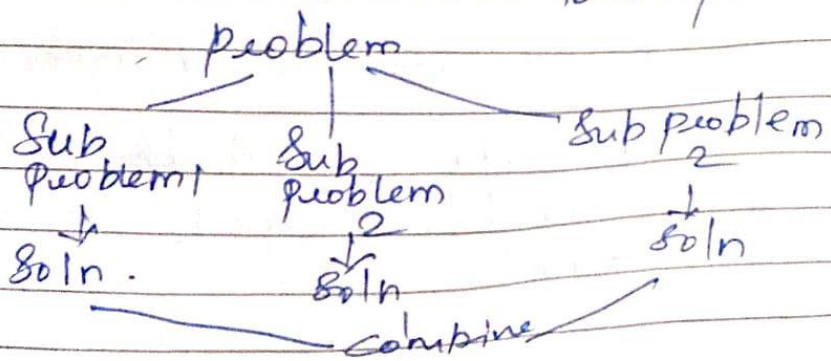
* All NP problems are not in Np Hard.

Example:

Halting problem, No Hamiltonian cycle

(v). Problem Reduction:

→ Strategy is dividing complex problem in to smaller sub problem



Applications:

→ Divider & conquer (Quicksort, Mergesort)
→ Recursion. (Problem → Best cases)
⇒ AI (Reduce complex goal into sub goals)

Algorithm & Analysis of 'n' Queens pu

Algorithm solveNQueens(n)
{
board [n][n] = 0
place Queen (board, row=0)
if (placeQueen returns true
Print ("Solution found").
else
Print (No Solution);
}

Time Complex
O(N!) → for each row; Try all possible
Space Complex
O(N) → single Solution
O(N*K) → for storing all solutions

Algorithm placeQueen (board, row)
{
if (row ≥ n)
return true // Queens placed success
for(col from 0 to n-1
if((IsSafe (board, row, col) == true
place queen at board [row] [col]=1
if place queen(board, row +1) } return true)