Design & Analysis of Algorithms (D.A.A)

Assignment - 2

Name:- B. Ravi Kumar
class:- CSE-B
Branch:- Computer science engineering (CSE)
Roll no:- 20A71A05B5
Date:- 24/06/2022.
Marks allotted:-

Date
24/06/2022

1) Define Cooks theorem and Judge.

Scientist stephen 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 active expression is true

this algorithm is composed by

1. input type
2. Read / write head
3. each cell contains only one symbol at a time
4. computation is performed into no. of states
5. The algorithm terminates cooks pt reaches the accept state

The conjuction clauses for boolean expressions are given in following

Clauses	Meaning	Time Complexity
T_{ij0}	Cell i of i/p tape contains symbol j	$O(p(n^2))$
Q_{s0}	Initial state	$O(1)$
H_{00}	Initial position of tape head	$O(1)$
$T_{ijk}=T_{ij(k+1)}^{\neg H_{ik}}$	tape remains unchanged unless written	$O(p(n^2))$
$Q_{qk}\rightarrow\neg Q_{qk}$ $H_{ik}\rightarrow\neg H_{ik}$	one state at a time one read/write head position at a time	$O(p(n))$ $O(p(n^2))$

Note:- H-head
Q-states
T-tape

disjunction clauses for the boolean expressions

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 problem in NP could be solved in polynomial time and hence class NP could be equal to P.

2) What are Graph traversal techniques and explain?

Graph traversal is a technique to visit the each nodes of a graph G. It is also use to calculate the order of vertices in traverse process. We visit all the nodes starting from one node which is connected to each other without going into loop.

Basically in graph it may happen some time visiters can visit one node more than once. so, this may cause the going the visiters into infinite loop. so, to protect from this infinite loop condition we keep record of each vertices. Like if visiters visited vertex then the value will be zero if not, then one

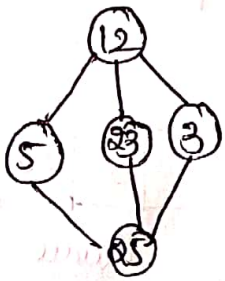
There are two graph traversal techniques

1. Breadth-first search
2. Depth-first search

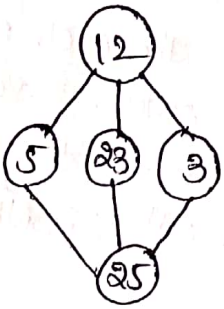
1. Breadth-first search:-

Breadth-first search graph traversal techniques use a queue data structure as an auxiliary data structure to store nodes for further processing. The size of the queue will be the maximum total number of vertices in the graph.

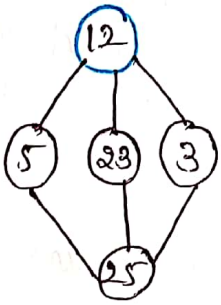
Examples

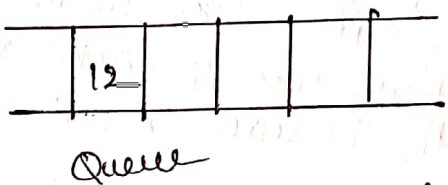


Initialize the queue

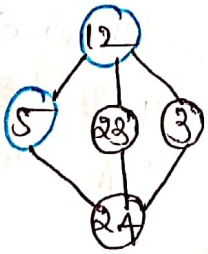


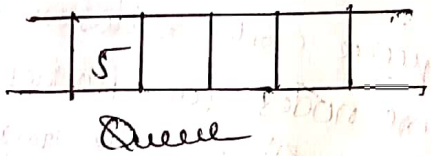
We start visiting from vertex 12 that can be considered as starting node, and mark it as visited



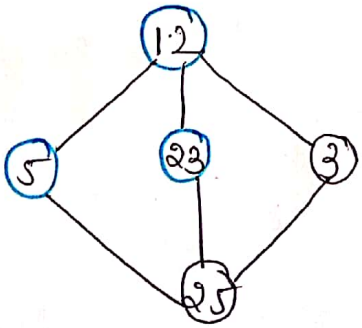


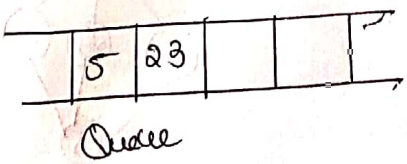
Now we will see an unvisited adjacent node from 12. In this example, we have three nodes and we can visit anyone. Here we are traversing from left to right. We choose 5 and mark it as visited and enqueue it



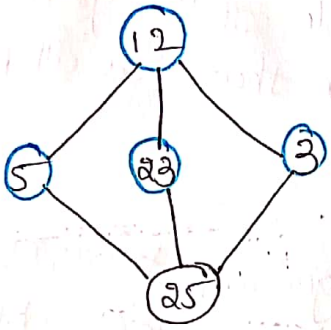


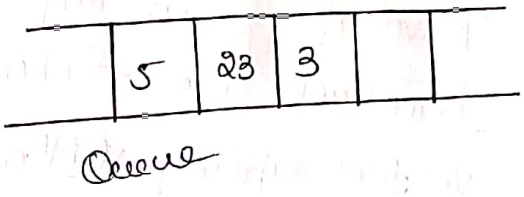
Next, visit the unvisited adjacent node from 12 to 23. We mark it as visited and enqueue it





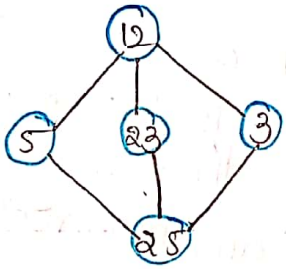
Next, the unvisited adjacent node from 12 is 3. we mark it as visited and enqueue it

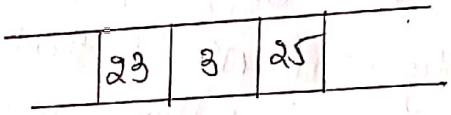




Queue

Now, all the connected adjacent node from 12 is traversed and no unvisited adjacent nodes left. so, we dequeue and find all connect vertex from 5.





Queue

From 5 we have 25 as unvisited adjacent node. we mark it as visited and enqueue it.

Now. If all node visited, we will dequeue all nodes from queue

So BFS Traversal output is: 12, 5, 23, 3, 25.

Complexity Analysis of BFS:-

Time Complexity: $O(V+E)$, where V is the number of vertices and E is the number of edges in the graph.

Space Complexity: $O(V)$

Since, an extra visited array is needed of size V.

Depth-first search:-

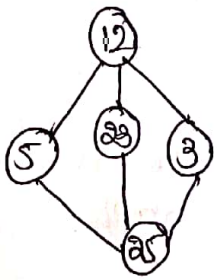
DFS stands for Depth First Search, is one of the graph traversal algorithms that use stack data structure. In DFS Traversal go as deep as possible of the graph and then backtrack once reached a vertex that has all its adjacent vertices already visited.

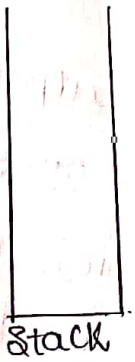
Depth first search (DFS) algorithm traverses a graph in a depthward motion and uses a stack data structure to remember to get the next vertex to start a search when a dead end occurs in any iteration.

DFS traversal for tree and graph is similar but the only difference is that a graph can have a cycle but the tree does not have any cycle. So in the graph we have additional array which keeps the record of visited array to protect from infinite loop and not visited again the visited node

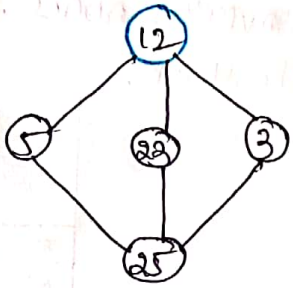
Graph Traversal using DFS Technique

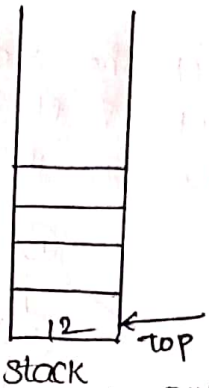
initialize the stack



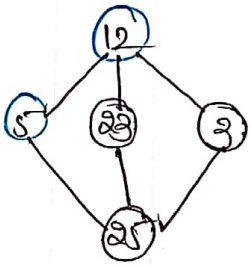


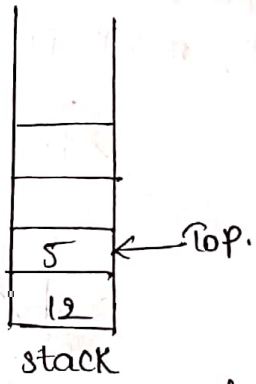
Mark 12 as visited and push into stack. Now explore any unvisited adjacent node from 12 in our example we have three nodes. we can pick any of them. Here we are going to pick 5



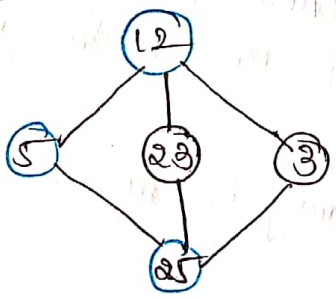


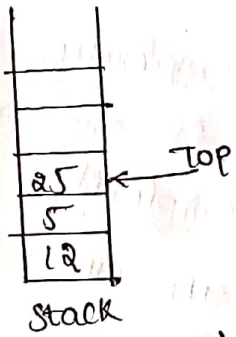
Mark 5 as visited and put it onto the stack. explore any unvisited adjacent node from 5 Both 12 and 25 are adjacent to 5 but we are concerned for unvisited nodes only.



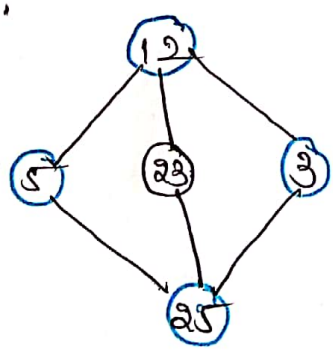


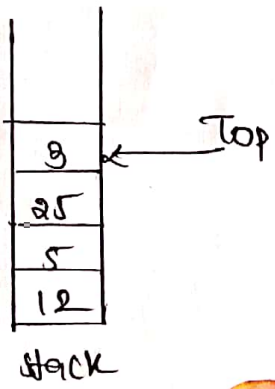
visit 25 and mark it as visited and put onto the stack, Here, we have 23 and 3 nodes, which are adjacent to 25 and both are unvisited.



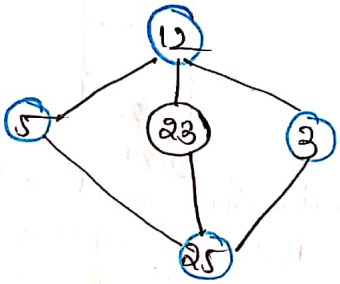


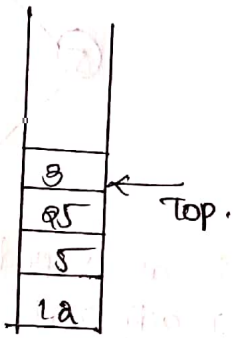
we choose 3, mark it as visited and put onto the stack. Here 3 does not have any unvisited adjacent node. So, we pop 3 from the stack.



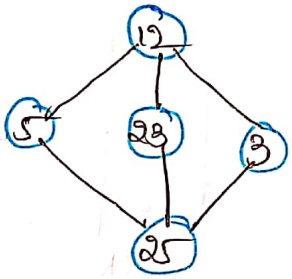


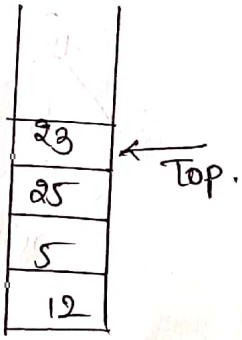
we check the stack top for return to the previous node and check if it has any unvisited nodes. Here, we find 25 to be on the top of the stack.





only unvisited adjacent node is from 5 is 23 now. So we visit 23, mark it as visited and put it onto the stack





Output of DFS Traversal will be: 3, 23, 25, 5, 12

Complexity Analysis:-

Time complexity: $O(V+E)$, where V is the number of vertices and E is the number of edges in the graph.

Space Complexity:- $O(V)$

Since, an extra visited array is needed of size V.

3. Explain the method of reduction to solve travelling sales person problem using branch and bound.

Procedure:

1. Find out reduced cost matrix from a given cost matrix this can be obtained as
   Fanout
   i) row reduction
   ii) column reduction
   iii) row wise reduction sum
   iv) column wise reduction sum
   v) commulative reduction ($\hat{c}$) = row wise reduction sum + column wise reduction sum

2. For starting node we will take commulative reduction as lower bound and infinity as upperbound.
   a) If path A(i,j) is considered then change all entries in row i & column j of A to ∞ (infinity).
   b) set A(j,1) to ∞ (infinity).
   c) Apply row reduction, column reduction except for rows & columns containing infinity
   d) Commulative reduction ($\hat{c}$) need to calculate then
      calculate $\hat{c}(s) = \hat{c}(r) + A(i,j) + \gamma$
      $\hat{c}(s) = \hat{c}(r) + A[i,j] + \gamma$
      $\hat{c}(r)$ = The lowerbound of r^{th} node in (i,j) path

repeat step-2 until all nodes are visited.

Let $A =$

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 4 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix} \begin{matrix} 10 \\ 2 \\ 2 \\ 3 \\ 4 \end{matrix}$$

21

row reduction

$$\begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 3 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix} = 4$$

min 1 0 3 0 0

Column reduction

$$\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 4 & 0 & 0 & 12 & \infty \end{bmatrix}$$

①

reduction cost = 21+4 = 25

from 1→2

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & 2 & 0 \\ 0 & \infty & \infty & 0 & 2 \\ 15 & \infty & 12 & \infty & 0 \\ 11 & \infty & 0 & 12 & \infty \end{bmatrix}$$

②

$C(1,2) + C(1) + \hat{r} = 10 + 25 + 0 = 35$

from 1→3

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 15 & 3 & \infty & \infty & 0 \\ 11 & 0 & \infty & 12 & \infty \end{bmatrix}$$

$\hat{r} = 11$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 4 & 3 & \infty & \infty & 0 \\ 0 & 0 & \infty & 12 & \infty \end{bmatrix}$$

③

17+25+11 = 53

from 1→4

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ 0 & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix}$$

④

cost = 4+0+25 = 29

from 1→5

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & 2 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 15 & 3 & 12 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix} 2$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 10 & \infty & 9 & 0 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 12 & 0 & 9 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix}$$

3

5

⑤

$= C(1,5) + C(1) + \hat{r}$

$= 1 + 25 + 5 = 31$

cost 1→4 (cost) is (25) less, so prefer 1→4 root

from 4→2

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 4 & \infty & 0 \\ 0 & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & 0 & \infty & \infty \end{bmatrix}$$

(6)

$C(4,2) + 0$
$3 + 25 = 28$

from 4→3

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & \infty & 0 \\ \infty & 3 & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & 0 & \infty & \infty & \infty \end{bmatrix} \quad 2 \quad \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & \infty & \infty & 0 \\ \infty & 1 & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & 0 & \infty & \infty & \infty \end{bmatrix}$$

$+ \quad 4 = 13$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & \infty & 0 \\ \infty & 1 & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \end{bmatrix}$$

(7)

$= C(4,3) + C(4) + r$
$= 12 + 25 + 13 = 50$

from

from 4→5

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & 0 & \infty & \infty \\ 0 & 3 & \infty & \infty & \infty \\ 1 & \infty & \infty & \infty & \infty \\ \infty & 0 & 0 & \infty & \infty \end{bmatrix} \quad r=4$$

(8)

$= C(4,5) + C(4) + r$
$= 0 + 25 + 11 = 36.$

from all roots 4→2 is the cheap.

from 2 → 3

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 2 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \end{bmatrix} \quad r = 2+11 = 13$$

(9)

$= 11+28+13$

$= 52$

from 2 → 5

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \end{bmatrix} \quad r = 0.$$

(10)

$C = (2,5) + C(6) + r$

$= 0 + 28 + 0 = 28$

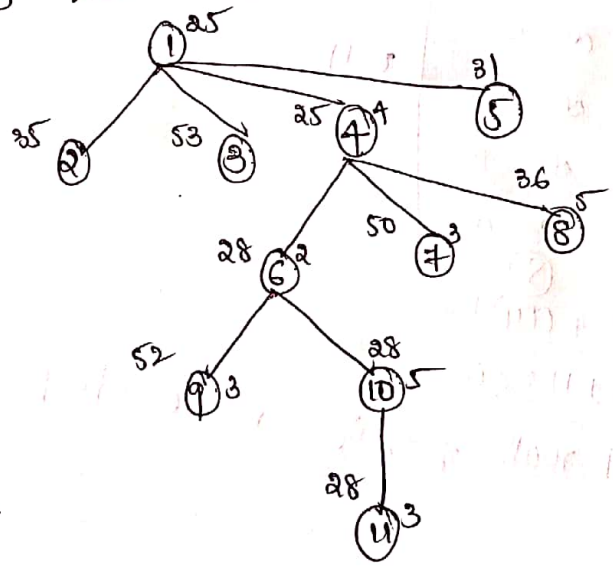
Among all roads 2 → 5 is cheap.

from 5 → 3

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

Cost = 0 + 28 + 0 = 28.

∴ now root map for traversal is



output:-

1 → 4 → 2 → 5 → 3 →)