I.E.S. College of Engineering

2nd Internal Examination

Date : 20 April 2020

Name : Jovial Joe Jayarson

Roll No. : IES17CS016

Subject : CS302 · Design & Analysis of Algorithms

Marks Awarded:

Q1.) Given Matrices

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 2 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 1 \\ 3 & 2 \end{bmatrix}$$

Acc. to strassen's method the multiplication can be done as follows:

$$M_1 = (A_{11} + A_{22})(B_{11} + B_{22}) = (1+2)(0+2) = 6$$

$$M_2 = (A_{21} + A_{22}) \cdot B_{11} = (0+2) \cdot 0 = 0$$

$$M_3 = (A_{11})(B_{12} - B_{22}) = 1 \cdot (1 - 2) = -1$$

$$M_4 = A_{22}(B_{21} - B_{11}) = 2 \cdot (3 - 0) = 6$$

$$M_5 = (A_{11} + A_{12}) B_{22} = (1+2)(2) = 6$$

$$M_6 = (A_{21} - A_{11}) \cdot (B_{11} + B_{12}) = (0-1)(0+1) = -1$$

$$M_7 = (A_{12} - A_{22}) \cdot (B_{21} + B_{22}) = (2-2)(3+2) = 0$$

∴ The addition will be as follows:

$$C_{11} = M_1 + M_4 - M_5 + M_7 = 6 + 6 - 6 + 0 = 6$$

$$C_{12} = M_3 + M_5 = -1 + 6 = 5$$

$$C_{21} = M_2 + M_4 = 0 + 6 = 6$$

$$C_{22} = M_1 - M_2 + M_3 + M_6 = 6 - 0 + (-1) + (-1) = 4$$

∴ The resultan matric C using strassen's algorithm will be:

$$C = \begin{bmatrix} 6 & 5 \\ 6 & 4 \end{bmatrix}$$

A.10)

To Prove: Hamiltonian Cycle is NP-Complete.

Definition: <u>Hamiltonian Cycle</u>

- It is a path that covers all the vertices and returns to its start point, without repeating any ~~edges~~ vertices.

<u>Proof</u>:

- To show that the hamilonian cycle is NP-complete we have to prove that Ham-Cycle is in NP.
- Second part would be to prove Ham-Cycle is NP-complete would be to prove that every other problem in NP can be reduced to Hamcycle.

Part 1 To prove Ham-Cycle is in NP.

- This is done by taking a certichtifacte.
- This certificate is a set of N vertices making up the Hamiltoniah Cycle.

- To check if this list of vertices is a true solution to the ~~Ham~~-cycle problem, one counts the vertices to make sure that they are all there. (ie all vertices are covered)

- Then it checks that each vertex is connect to the next one by an edge and that the last one is connected to the first.

- This is the so-called "verification algorithm."

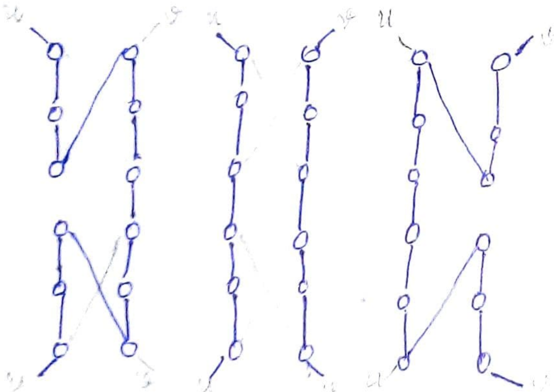
- Now it is easily visible that the time taken by the verification algorithm to verify or prove the certificate is polynomial time.

- This is because there are n-vertices to count and n-edges to check. ⇒ The algorithm approximalty runs in $O(n^2)$ time (maximum time in complete graph).

- Therefore the Hamiltonian cycle is in NP.

Part - 2 : To prove Every other problem in NP can be polynomial time reduced to Hamitonian Cycle.

- Now ~~it is~~ a known NP-complete problem is Vertex cover (which is a set of vertices that touches the all edges in the graph)
- We will reduce this Vertiex Cover in polynomial time to Ham-cycle.
- For that we construct a widget for each edge in the graph.



(i) (ii) (iii)

i.e u, v in the graph creas widgets.
Three ways to travers a widget
(i) Enter and exit through u
(ii) Enter from u, go anywhere in the graph leve the graph throug v.
(iii) Enter and exit through v.

- With such a construction of a graph with vertex cover it can be used to make a graph through Hamiltonial cycle.
- Since this can be done under polynomial time the reduction can be done in polynomial time.
- Therefore Hamiltonian cycle is NP-complete.

A9.)

Steps to show that a give problem is NP-Complete:

1. Prove that the problem is in NP.

ⓐ Analyse the problem and see that if provide a decision making or optimization senario.

ⓑ Proceed if it is decision making which is preferd the most.

ⓒ Obtain a certification - which would be tested.

ⓓ Obtain an algorithm which is in polynomial time to test this certificate

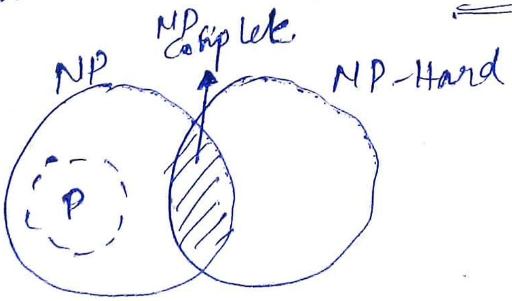
ⓔ If the certificate is valid then the problem is in NP.

2. Once the problem is in NP prove that it is NP-Complete.

ⓐ Take a known NP complete problem.

ⓒ Try to reduce it to the given problem using transformations within polynomal time.

ⓓ If it is possible then the given problem is NP complete.



~ Polynomial time reduction is a method which is used to reduct the ~~un~~known problem to an unknown problem (and vice ~~p~~ versa if possible)

~ This tells us that if "A reduces to B" and if "we can solve B" then "we can solve A", – in polynomial time

~ Cobham's thesis suggest that polynomial-time algorithm capture the notion of efficient algorithm.

~ Most common poly-nomial-time reductions are

(i) Many-One Reduction

~ Karp's reductions or polynomial transforms.

~ Problem A to B - as an transfomation of input to the problems.

(ii) Turing Reduction (cook's reduction)

~ Using polynomial number of subrouting call from A to B

(iii) Truth table reduction

~ It is an algorithm transform from problem A to Problem B as a transformation of outputs

eg:- satisfiability problem to Hamiltonial Cycle Problem.

A 4)

Divide And Conqpure	Dynamic Programming
~ They have Independent Subproblem	~ Have overlapped subproblems.
~ Recursive breakdown of probum into simple sub problems	~ Effiive solution to simpler problems then combining them.
~ Uses top-down approach.	~ Uses bottom-up approach.
~ Relative more time consuming ⇒ lower efficiency	~ Relative less time consuming ⇒ more time consuming.
~ Dupliction is igored.	- Dupliction is coplelly avided.
eg:- ~~Chain Multiplication~~ ~~Bellman~~ Binary Seach Merge Sort	eg:- ~~Binary Search~~ ~~Merge sort~~ Bell Manford algorithm Chain Multipliation

A5.) Given:

Items	A	B	C	D	Total Capacity
Profit	280	100	120	120	W = 60
Weight	40	10	20	24.	

~ Assuming 0-1 knapsack problem

~ The maximumprofits is aimed at but with optimal solution.

~ Following approaches can be used:

(i) Selection of items with largest profit.

A → profit = 280 : wt. = 40 : remain wt. : 20

C → profit = 120 : wt = 60 : remain wt. : 0

⇒ Therefore the net profit = 400

(ii) Selecting items with minimum weight:

B → profit = 100 : wt = 10 : remain wt. = 50

C → profit = 120 : wt = 20 : remaining wt. = 30

D → profit = 120 : wt. = 24. remain wt = 6

⇒ The net profit = 340

(no other complete object can be picked)

∴ The maximum profit is 400 by selecting the item A & C.

S.

A6.) Kruskal's Algorithm

1. Start from minimum weighted edge.

2. Find the next edge with minimum weight.

3. There are 4 case involved.

(i) Both the vertices of the newly selected egdge doesnot belong the the new spaning tree T_1: create a new spanning tree T_2

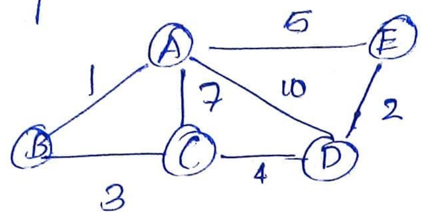
(ii) Both the vertices belong to the same spanning tree T_1 : discard the edge - it forms a cycle.

(iii) One of the vertices is found in spaning tree T_1 : add it to the existinge spaning tree.

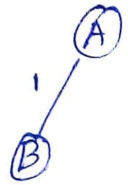
(iv.) One vertex blongs to T_1 and the other belong the T_2 : merge T_1 & T_2 using the currently selected edge, such that it does not ~~pefor~~ create cycle

4. Slow ~~[illegible]~~ increase the minimum weight and keep on collecting edges as in step 3 untill all of them are exhausted.

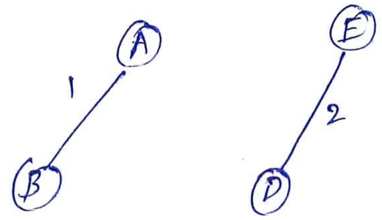
8. Give graph



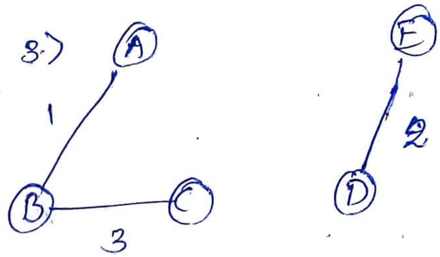
1:)



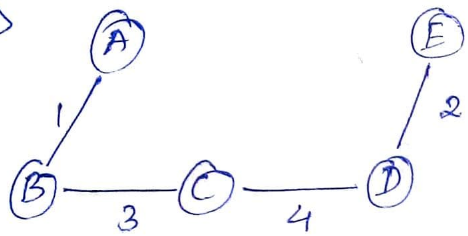
2:)



3:)



4:)



which is the required minimum spaning tree (edges AE, AD, AC cannot be included b'coz they form a cycle).

- Since the algorithm scans for the minimum spaning tree scans for minimum edge one it has to do at a polynomial time O(n).
- Again when it checks for any cycles during tree creation it takes O(1).
- Thus the kruskal algorithm logarithm time O(nlgn).

A3.) Bellman ford algorithm

- This algorithm is used to solved problems that dijkstra algorithm cannot.
- It tries to find the minimum weight even if there are negative weights.

- Bellman ford algorithm will not work if there is any negative weighted cycle in the Graph.

- The algorithm goes like this:

1. Start

2. Select any vertex u; and mark it as 0 weight.

3. Mark all other vertices v_i as ∞.

4. ~~Move~~ Calculate the weights of the adjacent vertices as:

$$\text{if } (d[u] + c(u, v) < d[v])\ \{$$
$$d[v] = d[u] + c(u, v)$$
$$\}$$

where $d[u]$ or $d[v]$ is the current weight upon them & $c(u, v)$ is the cost of that gets added upon when traversing from u to v.

5. Repeat the above steps for $n-1$ times where n is the number of vertices.

6. If weight remain same after any iteration then stop the iteration.

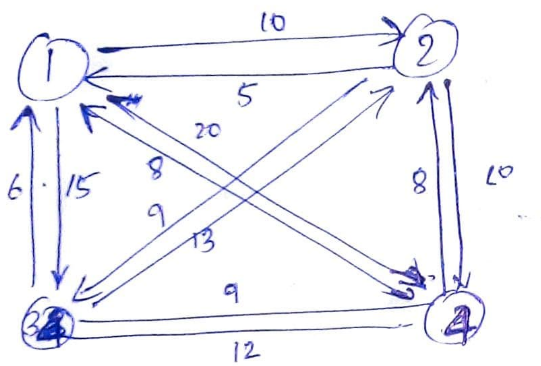
7. Stop.

~ Now if there are m-edge an n-vertices the maximum number of iteration will be $m \times n$.

~ Thus for the complexity of Bellman ford algorithms is $O(m \cdot n)$ - which is polynomial time.

$O(n)$ in general.

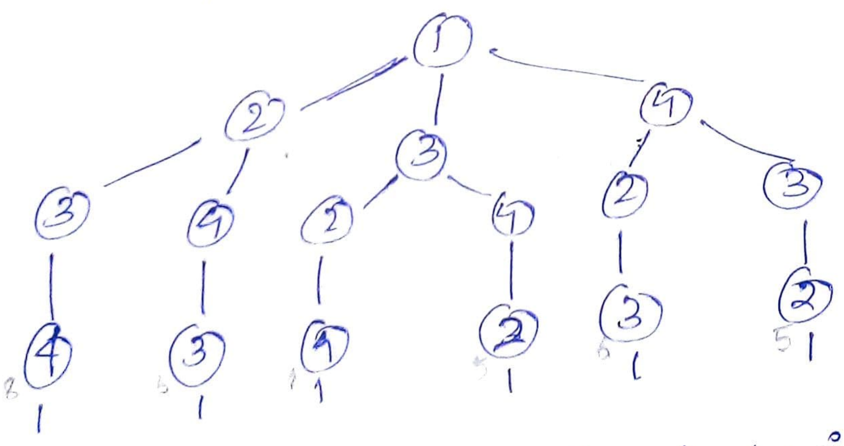
A8.) Given graph



The adjacency Matrix will be :

$$
\begin{array}{c|cccc}
 & 1 & 2 & 3 & 4 \\
\hline
1 & 0 & 10 & 15 & 20 \\
2 & 5 & 0 & 9 & 10 \\
3 & 6 & 13 & 0 & 12 \\
4 & 8 & 8 & 9 & 0
\end{array}
$$

The tree can be drawn as



The dynamic programming problem is solved using:

$$g(i, S) = \min_{k \in S} \{ c_{ik} + g(k, S - \{k\}) \}$$

where, i = starting vertex c_{ik} = cost of (i to k).

S = set of remaining vertices   This is a recursive algorithm.

$\therefore g(2, \phi) = 5$

$g(3, \phi) = 6$

$g(4, \phi) = 8$

$g(2, \{3\}) = c(2,3) + g(3, \phi) = 9 + 6 = 15$

$g(2, \{4\}) = c(2,4) + g(4, \phi) = 10 + 8 = 18$

$g(3, \{2\}) = c(3,2) + g(2, \phi) = 13 + 5 = 18$

$g(3, \{4\}) = c(3,4) + g(4, \phi) = 12 + 8 = 20$

$g(4, \{2\}) = c(4,2) + g(2, \phi) = 8 + 5 = 13$

$g(4, \{3\}) = c(4,3) + g(3, \phi) = 9 + 6 = 15$

$g(2, \{3,4\}) = \min \{ c(2,3) + g(3, \{4\}), \; c(2,4) + g(4, \{3\}) \} = \{9 + 20, 10 + 15\} = \underline{25}$

$g(3, \{2,4\}) = \min \{ c(3,2) + g(2, \{4\}), \; c(3,4) + g(4, \{2\}) \} = \{13 + 18, 12 + 13\} = \underline{25}$

$g(4, \{2,3\}) = \min \{ c(4,2) + g(2, \{3\}), \; c(4,3) + g(3, \{2\}) \} = \{8 + 15, 9 + 18\} = \underline{23}$

$g(1, \{2,3,4\}) = \min \{ c(1,2) + g(2, \{3,4\}), \; c(1,3) + g(3, \{2,4\}), \; c(1,4) + g(4, \{2,3\}) \} = \{10 + 25, 15 + 25, 20 + 23\} = \{35, 40, 43\} = 35$

$\therefore$ The TSP can be solve via 1 → 2 → 3 → 4 with ~~cast~~ ~~cost~~ cost of 35

7.) N-Queen's Problem

- There are N Queen on a chess bored of size N×N.
- Now let N = 4, the condition is that ~~there~~ the ~~mus~~ arragemt must be such that ~~all the~~ ~~ques~~ (i) All the queens should be on the board

(ii) There should not be an attrack on any of the queen either horizontally, verticaly or digonally.

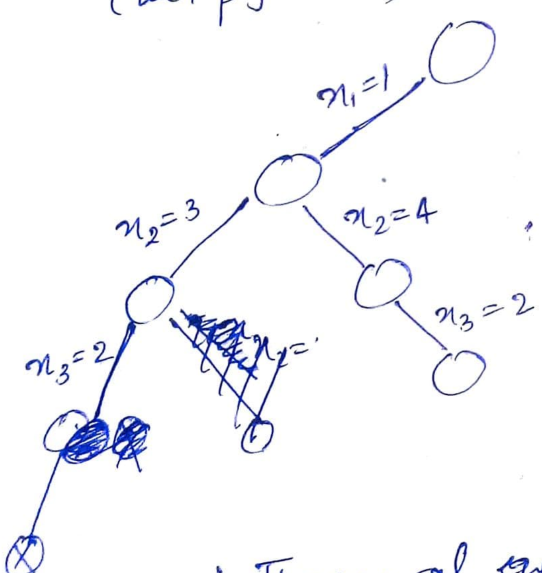
	Q_1 1	Q_2 2	Q_3 3	Q_4 4
1			Q_1	
2	Q_2			
3				Q_3
4		Q_4		

- The queen are places such that they are not in attrack.

x = colum $\quad (1, 2, 3, 4) \Rightarrow$ Queen no. be

y = row

(we keep y constant)



. . . . completing this tree we get $\Rightarrow$

Q_2	Q_4	Q_1	Q_3
1	2	3	4

Q_3	Q_1	Q_4	Q_2

∴ The general equation will be: $1 + \sum_{i=0}^{N-1} \left[\prod_{j=0}^{i} (N-j) \right]$

when N is the number of queens.

A2.) The given matrix and dimentions

Matrix	Dimension
A1	30 * 35
A2	35 * 15
A3	15 × 15
A4	5 × 10
A5	10 × 20
A6	20 × 25

M	1	2	3	4	5	6	7
1	0	15750					
2		0	2625				
3			0				
4				0			
5					0		
6						0	
7							0

The possible combinations are $\frac{^{2\times5}C_5}{6} = 42.$

The equation for min number of multiplication is.

$$m[i,j] = \begin{cases} 0 & , \text{ if } i=j \\ \min\limits_{i\le k<j}\{m[i,k] + m[k+1,j] + P_{i-1}\cdot P_k \cdot P_j\} & , \text{ if } i<j \end{cases}$$

$$\Rightarrow m[2,5] = \min_{2\le k\le 5}\{m[2,k] + m[k+1,5] + P_1 \cdot P_k \cdot P_5\}$$

The dimension are

$P_0 = 30$, $P_1 = 35$, $P_2 = 15$, $P_3 = 5$, $P_4 = 10$, $P_5 = 20$, $P_6 = 25$

k	1	2	3	4	5	6	7
1	0	1					
2		0	2				
3			0				
4				0			
5					0		
6						0	
7							0

With chainlength = 1 $(i==j)$

$m[1,1] = m[2,2] = m[3,3] = m[4,4]$

$m[5,5] = m[6,6] = 0$

With chain length = 2 $(|i-j|=1)$

$m[1,2] = m[1,1] + m[2,2] + P_0 \cdot P_1 \cdot P_2$

$k=1$ $= 0 + 0 + 30 \times 35 \times 15 = 15750$

$m[2,3] = m[2,2] + m[3,3] + P_1 \cdot P_2 \cdot P_3$

$k=2$ $= 0 + 0 + 35 \times 15 \times 5 = 2625$