GREEDY TECHNIQUE: We can solve imp problems like knapsack problem, travel Sales person problem, PRIM'S KRUSKAL'S problem.

→ In this technique we consider solution of the problem in such a way that we are able to get feasible or optimum solution.

Foll. are the steps in greedy approach:

(1) Feasibility check: in this step we check that weather selected approach satisfies all the problem constraints or not.

(2) Local optimum choice:

Scanned by CamScanner

31/1/17

Fractional eqn: In this technique items are arranged in the ratio of $R_i = \frac{V_i}{w_i}$

We will arrange all the items in descendi order of ~~their~~ above ratio and then we wil solve the knapsack problem as usual.

Q. Consider the foll. information about napsack.

$I = \langle I_1, I_2, \ldots I_5 \rangle$ $W = 60$.

$w = \langle 5, 10, 20, 30, 40 \rangle$

$v = \langle 30, 20, 100, 90, 160 \rangle$

INITIAL

	v	w	R_i
I_1	30	5	6
I_2	20	10	2
I_3	100	20	5
I_4	90	30	3
I_5	160	40	4

Now, by arranging all the elements in descending order of their ratio we get.

	v	w	R_i
I_1	30	5	6
I_3	100	20	5
I_5	160	40	4
I_4	90	30	3
I_2	20	10	2

$W = 60$

1) $S = \langle I_1 \rangle$

$v = 30$

$w = 5$

Now $W = 55$

Scanned by CamScanner

31/1/17

Fractional eqn: In this technique items are arranged in the ratio of $R_i = \frac{v_i}{w_i}$

We will arrange all the items in descending order of ~~their~~ above ratio and then we will solve the knapsack problem as usual.

Q. Consider the foll. information about napsack.

$I = \langle I_1, I_2, \ldots I_5 \rangle$ $W = 60$.

$w = \langle 5, 10, 20, 30, 40 \rangle$

$v = \langle 30, 20, 100, 90, 160 \rangle$

INITIAL

	v	w	R_i
I_1	30	5	6
I_2	20	10	2
I_3	100	20	5
I_4	90	30	3
I_5	160	40	4

Now, by arranging all the elements in descending order of their ratio we get.

	v	w	R_i
I_1	30	5	6
I_3	100	20	5
I_5	160	40	4
I_4	90	30	3
I_2	20	10	2

$W = 60$

(a) $S = \langle I_1 \rangle$

$v = 30$

$w = 5$

Now $W = 55$

(b) Now $20 + 5 = 55$

So, $S = \langle I_1, I_3 \rangle$

$v = 30 + 100 = 130$

$w = 5 + 20 = 25$

$W = 35.$

(c) $40 \rightarrow 160$

$35 \rightarrow \frac{35 \times 160}{40} = 35 \times 4. = 140.$

So, $S = \langle I_1, I_3, I_5^* \rangle$

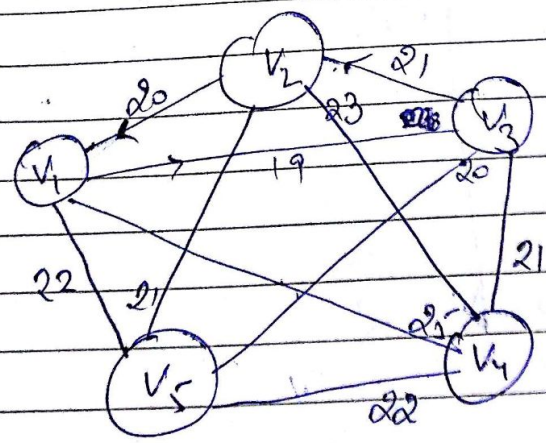
$v = 30 + 100 + 140 = 270$ } Ans.

$w = 5 + 20 + 35 = 60$

TSP problem

Travelling Sales Person problem.

Q. Consider the graph and find the total cost by applying greedy approach



	V_1	V_2	V_3	V_4	V_5
V_1	0	20	19	25	22
V_2	20	0	21	23	21
V_3	19	21	0	21	20
V_4	25	23	21	0	22
V_5	22	21	20	22	0

$$V_1 \xrightarrow{19} V_3 \xrightarrow{20} V_5 \xrightarrow{21} V_2 \xrightarrow{23} V_4 \xrightarrow{25} V_1$$

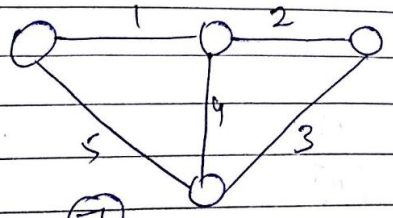
T.C = 108.

Optimum solution = 106.

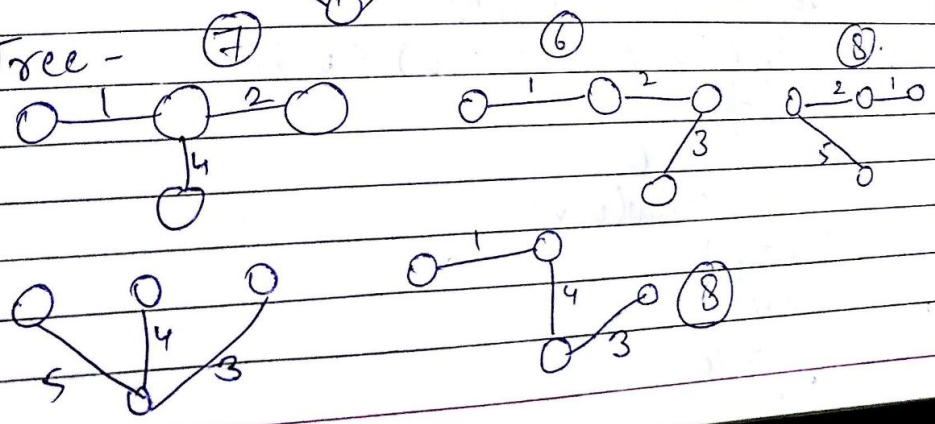
$V_1 \rightarrow V_2 \rightarrow V_3 \rightarrow V_4 \rightarrow V_5 \rightarrow V_1$.

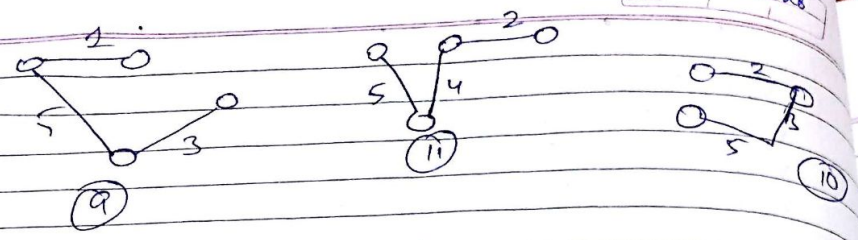
2/2/17

MST- Minimum Spanning Tree



Tree -





Algorithm

* To find MST, we use prim's algorithm and kruskal's algorithm.

7/2/17 PRIM'S ALGORITHM

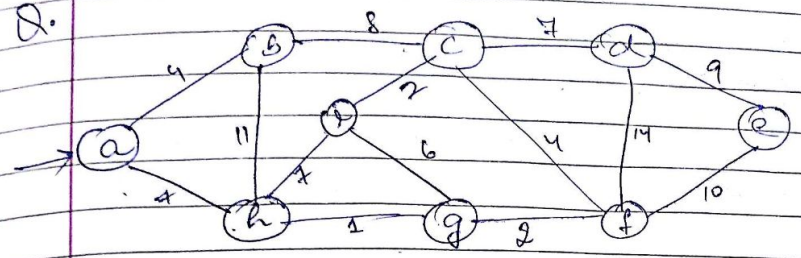
```
PRIM'S ALGO (G, w, r)
//
1. for each edge u ∈ G.V
    u.key = ∞
    u.π = NIL
    r.key = 0
    Q = G.V
//end for

while Q ≠ ϕ
u = EXTRACT_MIN(Q)
for each v ∈ G.Adj[u]
    if v ∈ Q and w(u,v) < v.key
        then
        v.π = u
        v.key = w(u,v)
//end if
//end for
//end while
//END
```

Scanned by CamScanner

→ G is the weighted graph
→ r is the root of spanning tree
→ Q is the minimum priority queue

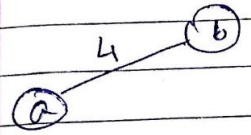
Initially each edge has weight infinite, there is not parent of any edge (π denotes parent) and initially the key of root is 0.

Q.

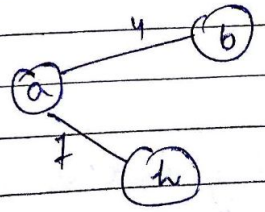


Solⁿ: (i) a

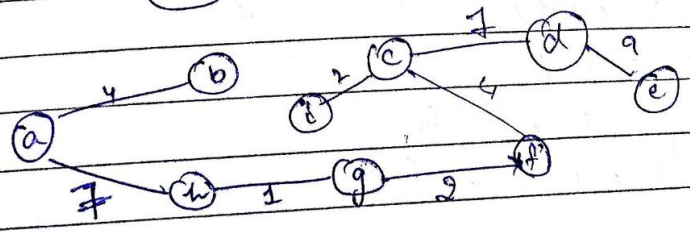
(ii)



(iii)



(iv)

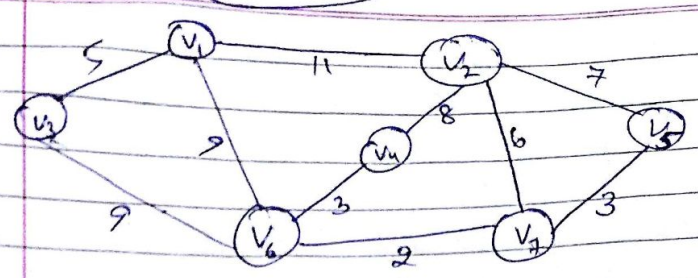


(v) Total cost = 36

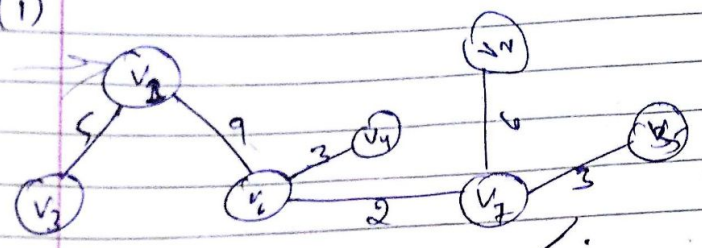
Rough work
a→b & a→h

Ans = 28

Q:



(i)



Ans = 28

9/2/17 MST

KRUSKAL'S (G, w)

graph, weight

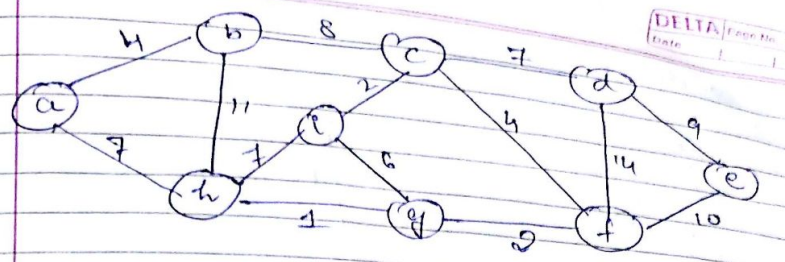
//

1. A = φ // A = MST
2. for each vertex v ∈ G
3. MAKE_SET(v)
4. SORT the edges of G.E in increasing order. (weight (cost))

~~5. If FIND_SET(u) ≠~~

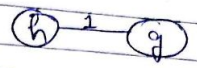
5. for each edge (u, v) ∈ G
6. If FIND_SET(u) ≠ FIND_SET(v)
7. A = A ∪ {(u, v)}
8. UNION (u, v)
9. Return A.

Scanned by CamScanner

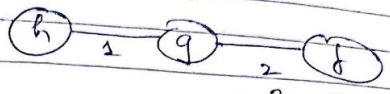


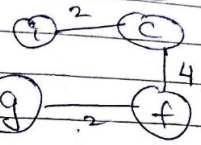
Edge	cost
h - g	1
g - f	2
c - i	2
c - f	4
a - b	4
i - g	6
a - h	7
c - d	7
h - i	7
b - c	8
d - e	9
e - f	10
b - h	11
d - f	14

(i)



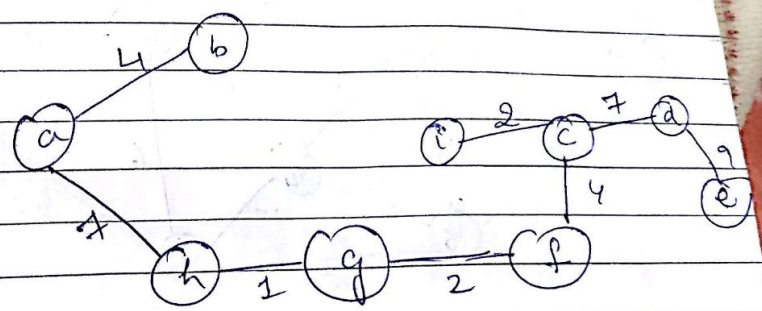
(ii)





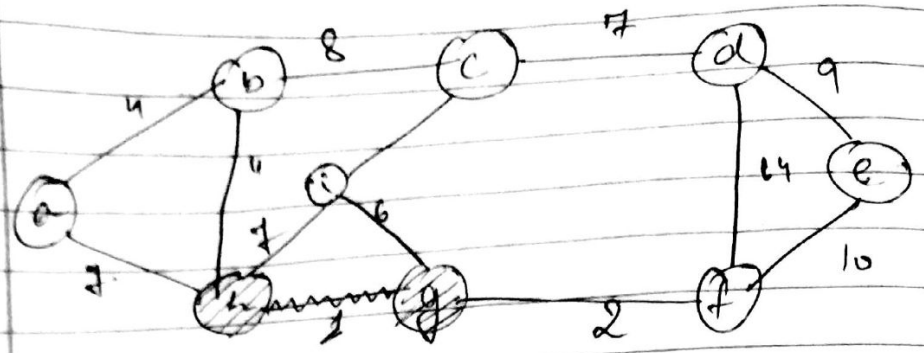
(iii)

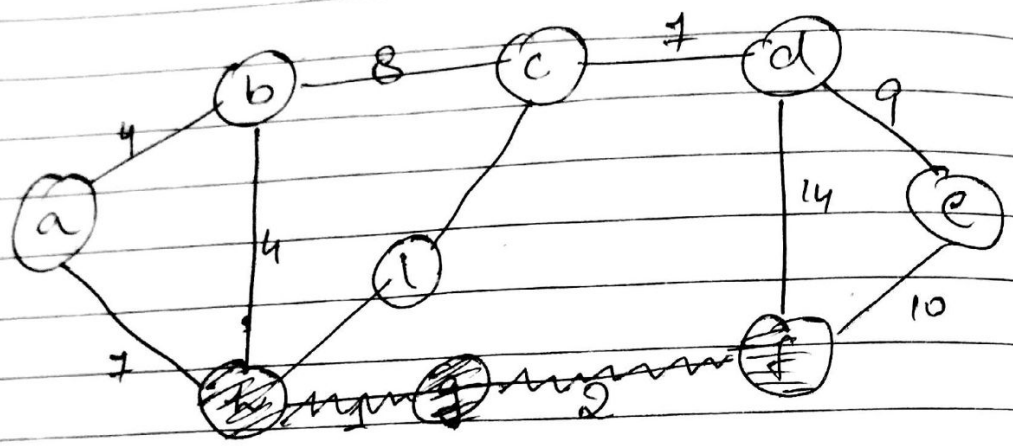
(iv)

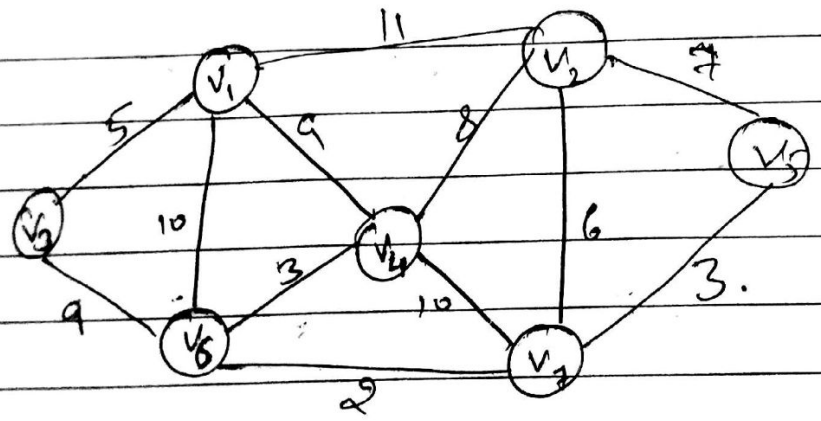


Total cost = 36

How to do in exam

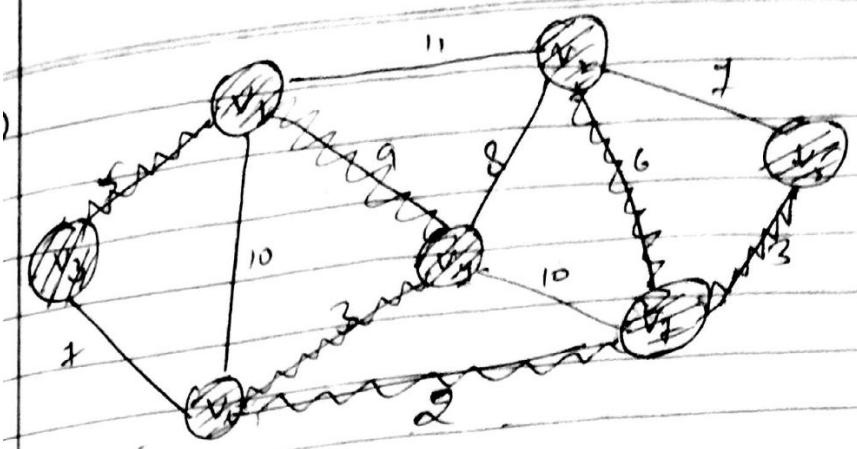






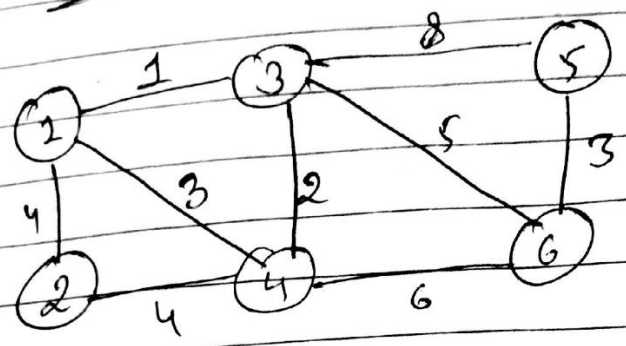
Edge	Cost		
$v_6 - v_7$	2	$v_4 - v_2$	8
$v_4 - v_6$	3	$v_1 - v_4$	9
$v_5 - v_7$	3	$v_3 - v_6$	9
$v_3 - v_1$	5	$v_1 - v_6$	10
$v_2 - v_7$	6	$v_4 - v_7$	10
$v_2 - v_5$	7	$v_1 - v_2$	11

Scanned by CamScanner



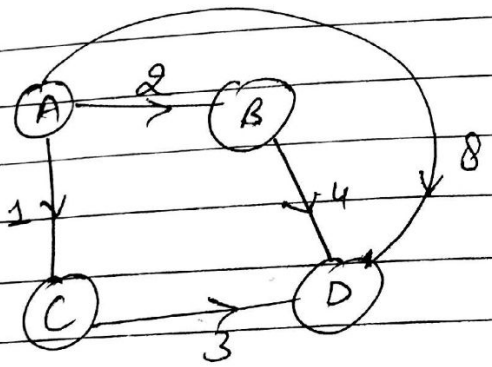
Total Cost = 28

HW



ans = 15

DIJKSTRA'S ALGO. (SINGLE SOURCE SHORTEST PATH PROB).

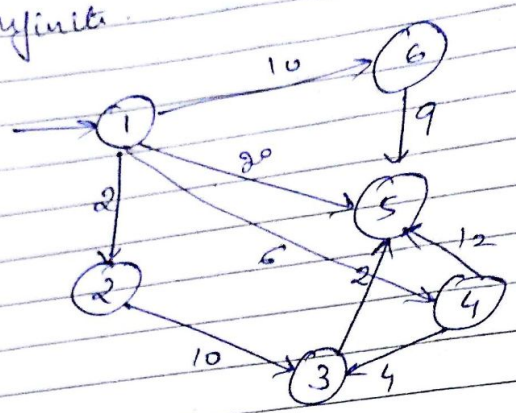


A → D : 8
A → B → D : 6
A → C → D : 4

We find the shortest path from one vertex to all other remaining vertices from the given task one vertex is selected as the source. This algo maintains a set of vertices whose shortest distance from source vertex is given. At each step we add to set the remaining vertex v whose distance from source is shortest.

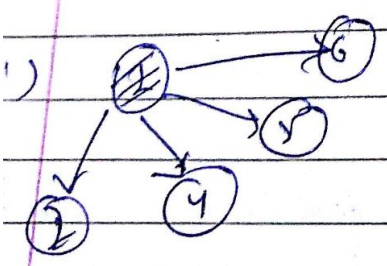
Scanned by CamScanner

Q. If there is no direct path from one city to another city then consider its cost as infinite.

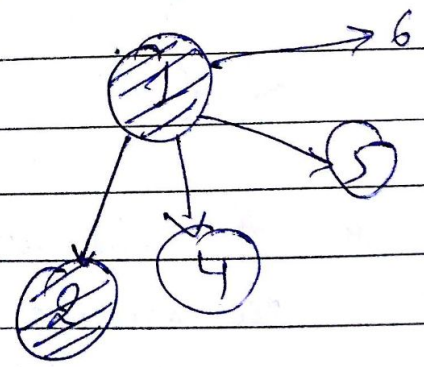


	Set(S)	v	d[2]	d[3]	d[4]	d[5]
INITIALLY	{1}	—	2	∞	6	20
1	{1, 2}	2	2	12	6	20
2	{1, 2, 4}	4	2	10	6	18
3	{1, 2, 4, 3}	3	2	10	6	12
4	{1, 2, 4, 3, 6}					

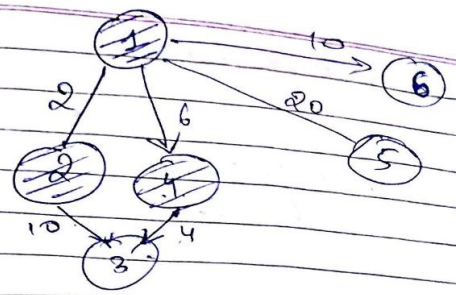
(i)



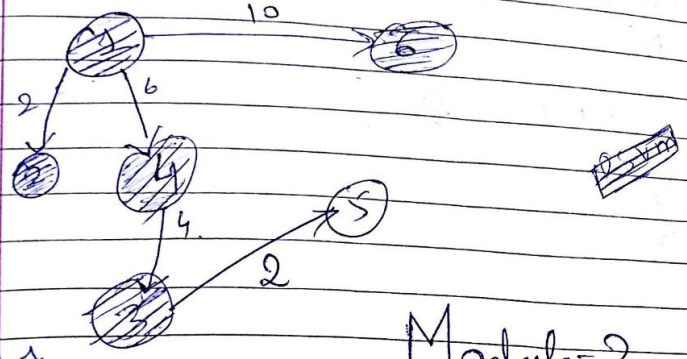
(ii)



(iii)



(iv)



Module-3

28/2/17

DYNAMIC PROG

– RICHARD BELLMAN 1956

This approach is able to solve the real world problem. It is nearly close to divide & conquer approach. Steps in DPR are:

1. Divide problem into subproblem
2. Table storage – We will store subproblems in t table
3. Combine – How ~~from~~ the ~~combination~~ Combination of so of subproblems will provide the solution.

Note: In this approach we always get optimum so

Differentiate Dynamic programming & greedy technique.

Longest Common Subsequence: (LCS)

1) P = <L M N>
Q = <M L N>

<LN> or <MN> Ans

<L>
<M>
<N>

2) <K L O>
<X K L L>
Ans: KL

3) X = K L O K M K N K L O K
Y = X K L L K N K L L K N Y Y

K L K K N = 5
and K L K N K L K = 7. (Ans)

X = <L M N L>
Y = <O L N L>
LCS = LNL

X = <M L N O M>
Y = <M N O M>

LCS = MNOM

Scanned by CamScanner

(6) P = < A B C D B C D C D D >
Q = < B C D C D >
LCS = B C D C D

(7) P = < 1 0 0 1 0 1 1 0 1 1 0 1 >
Q = < 0 1 1 0 >
LCS = < 1 0 >

➡ f_N _ LCS(P, Q)

1. // For i = 0 to m
    LCS[i, 0] = 0 //E.f
  For j = 1 to n
    LCS[0, j] = 0 //E.f

i = row
j = column

2. for (i = 1 to m // row)
    for j = 1 to n // col

If (P[i] = Q[j]) then
    LCS[i, j] = LCS[i-1, j-1] + 1
    b[i, j] = '↖'
else if (LCS[i-1, j] >= LCS[i, j-1]
    LCS[i, j] = LCS[i-1, j]
    b[i, j] = '↑'

else
    LCS[i, j] = LCS[i, j-1]
    b[i, j] = '←'

3. Return LCS[m, n]

Scanned by CamScanner

b[i,j]

		0	1	2	3	4
			M	N	O	M
0		0	0	0	0	0
1	M	0	1 ←	1 ←	1 ←	1 ←
2	L	0	1 ↑	1 ↑	1 ↑	1 ↑
3	N	0	1 ↑	2 ↖	2 ←	2 ←
4	O	0	1 ↑	2 ↑	3 ←	3 ←
5	M	0	1 ←	2 ↑	3 ↑	4 ←

LCS[1,1] = LCS[0,0] + 1 = 1

LCS[3,3]

3/3/17

II PART EXTR_LCS(P, Q, b)

P = <M L N O M>
Q = <M N O M>

//

1. i = m, j = n
2. while (i ≠ 0 and j ≠ 0) AND
   select case b[i, j]
   {
   Case '↖'
   add P[i]// or Q[j] in front of LCS
   i = i-1
   j = j-1
   break
   Case '↑'
   i = i-1
   ~~i = i-1~~
   break;
   Case '←'
   j = j-1
   break
   }
   return LCS
   END

Scanned by CamScanner

		M	N	O	M
	0	0	0	0	0
M	↑0	↖	←	←	←
L	↑0	↑	↑	←	↖
N	↑0	↑	↖	↑	↑
O	↑0	↑	↑	←	←
M	↑0	↖	↑	↑	↖4

LCS = <M N O M>

i = 5
j = 4

i = 4
j = 3

i = 3
j = 2

i = 2
j = 1

i = 1
j = 1

i = 0
j = 0

CHAINED MATRIX MULTIPLICATION

$A_{50\times20}$

$B_{20\times1}$

$C_{1\times10}$

$D_{10\times100}$

$A(B\ C)\ D$

$A(B(CD))$

$(AB)(CD) = 7000$

$((A\cdot B)C)D = 50000$

$A((B\ C)D) = 120{,}200.$

$C\times D = 1\times10\times100 = 1000$

$B_{20\times1}\,(CD)_{1\times100} = 20\times100 = 2000$

$A_{50\times20}\,B\,(CD)_{20\times100} = 50\times20\times100 = 100000$

103000

$A(B(CD))_{50\times100}$

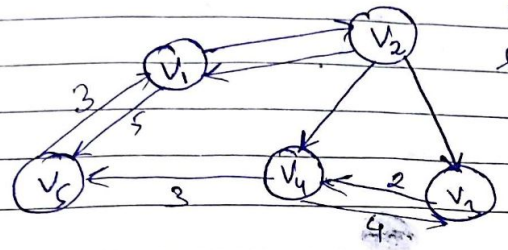
Scanned by CamScanner

Not imp.

3/12 All Pair Shortest Path [D.P]

APSP (w[1...n], n)

1. Initialize the matrix D with weight w.
2. for k = 1 to n //Distance
3. for i = 1 to n //Row
   for j = 1 to n //colymn
   $D^k[i][j] = \min\{D^{k-1}[i][j], D^{k-1}[i][k] + D^{k-1}[k][j]\}$
   End for
   End for
End for



There are 5 vertices hence answer will be D^5.

D^0	1	2	3	4	5
1	0	1	∞	1	5
2	9	0	3	2	∞
3	∞	∞	0	4	∞
4	∞	∞	2	0	3
5	3	∞	∞	∞	0

D^1	1	2	3	4	5
1	0				
2		0			14
3			0		
4				0	
5				4	0

$D^1[2][5] = \min\{D^0[2][5], D^0[2][1] + D^0[1][5]\}$.

$\min\{\infty, 9+5\} = 14$.

$D^1[5][4] = \min[\infty, 3+1] = 4$.

Scanned by CamScanner

Ans →

D	1	2	3	4	5
1	0	1	3	1	4
2	8	0	3	2	5
3	10	11	0	4	7
4	6	7	2	0	3
5	3	4	6	4	0

19/3/17 DFS (Depth First Search)

To solve DFS & BFS we use graph in which the nodes will be in following ~~way~~ 3 states:

1. Ready state (white) status = 1.
2. waiting state (grey)
3. Processed state (black)

→ Ready state means initial state of node

→ In waiting state node has been entered in stack or queue and it is waiting to be processed.

→ Node has been processed i.e. it is terminated from stack or queue. that means that state has been visited.

BFS Queue

DFS Stack

→ traversal of a graph

Date 10, 3, 17

Page No.: 1

DFS (Depth First Search)

→ To solve DFS & BFS we use graph in which the nodes will be in following 3 states:

1. Ready state (Initial) ○ white status = 1
2. Waiting state (grey) status = 2
3. Processed state (black) status = 3

- Ready state means initial state of node
- In waiting state node has been entered in stack or queue & is waiting to be processed.
- Node has been processed that it has been terminated from stack or queue, the state has been visited.

BFS Queue
DFS Stack

Algo_DFS

// Start

1. Initialize all the nodes (status = 1)
2. Put the starting node on the top of the stac & change its state (status = 2)
3. Repeat steps 4 & 5 untill stack is Empty
4. Remove Node (N) from top of the stack, pro N & change its state (status = 3)
5. Push all the adjacent node of N & change t state (status = 2)
6. End Loop // REPEAT
7. // END

Chitra

"Do not say, 'It is morning,' and dismiss it with a name of yesterday. See it for the first time as a newborn child that has no name." —Rabindranath Tagore

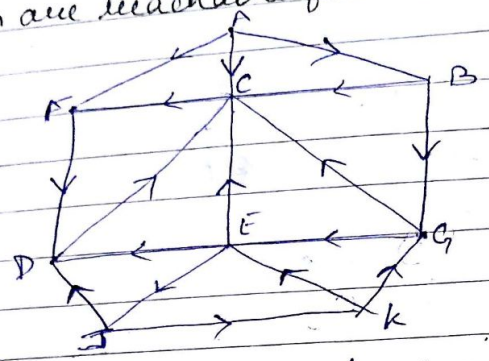
Scanned by CamScanner

Example of DFS

Railway network

In DFS we process children not siblings means we move towards depth.

Q. Consider the following graph & find all the nodes which are reachable from node J.

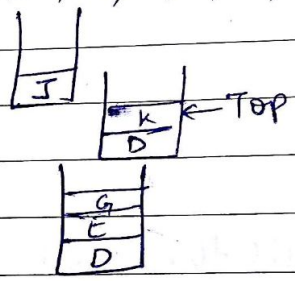


** Do not process any node twice.

PRINT : J, K, G, C, F, E, D → Ans

a) Stack :

b) Stack : J



c) Stack : Dk

d) Stack : DEG

e) Stack : DEC

f) Stack : DEF

h) Stack : DE

(i) Stack : D

(k) Stack :

DFS on a graph with m vertices & n edges take O(n+m) time.

"If we have no peace, it is because we have forgotten that we belong to each other." —Mother Teresa

Scanned by CamScanner

Algo - BFS

1. Initialize all the nodes (Status = 1)
2. Put the starting node in front of queue & change its state (status = 2)
3. Repeat steps 4 & 5 until queue is empty.
4. Remove the node from FRONT of the queue (N) & process N & change its state (status = 3)
5. INSERT all the adjacent nodes of N in rear of the queue & change its state (status = 2)
6. End loop // Repeat
7. End.

The running time for a BFS on a graph with n vertices and m edges is $O(n+m)$.

Scanned by CamScanner

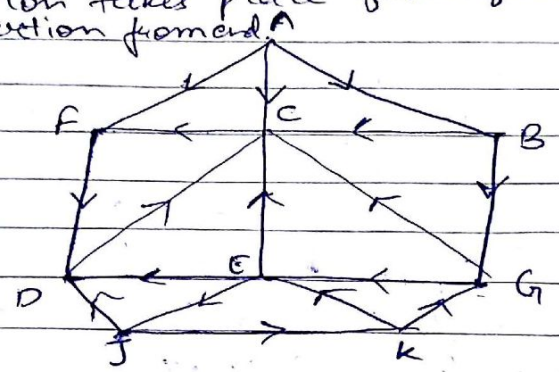
Queue is applied in BFS

→ Deletion takes place from front

→ Insertion from end

Date 14, 3, 17

Page No: 4



A = F C B

B = C, G

→ In this we traverse siblings not children.

→ We have to find a path which tells how many intermediate nodes are there from source to destination.

Q. Consider the above graph & find minimum path from source A to J.

First = 1 Queue : A

Rear = 1 ORIG: ϕ

	1	2	3	4
Parent	ϕ			
	A			

Front, Rear.

Print

A F C B D

ϕ A A A F

G E J K

B G E

3. Front = 2

Rear = 4

	F	C	B

F = 2 R = 4.

Queue = A F C B

ORIG = ϕ A A A

"We have to peace, it is because we have forgotten that we belong to each other." —Mother Teresa

Scanned by CamScanner

→ Front = 3 Queue: A F C B D
Rear = 5 ORIG: ϕ A A A F

A	F	C	B	D

↑ F = 3 ↑ R = 5

→ FRont = 4 Queue: A F C B D G
Rear = 6 ORIG: ϕ A A A F B

A	F	C	B	D	G

↑

→ FRont = 5 Queue: A F C B D G E
Rear = 6 ORIG: ϕ A A A F B G

~~→ FRONT = 6 Queue: A F C B D G E J~~
~~Rear = 8 ORIG: ϕ A A A F B G E~~
~~→ Front = 7 Queue: J k~~
~~Rear = 9 ORIG: E J~~
~~→ FRONT = 8~~
~~Rear =~~

→ Front = 6 Q: A F C B D G
Rear = 6 ORIG: ϕ A A A F B

→ Front = 7 Q: A F C B D G E
Rear = 7 ORIG: ϕ A A A F B G

→ Front = 8 Q: A F C B D G E J
Rear = 8 ORIG: ϕ A A A F B G E.

Scanned by CamScanner

A F C B D G E J
φ A A A C B G I

A → B → G → E → J Ans
This is the minimum path.

~~BACK TRACKING~~

We can solve the same problem using undirected graph also.

BACK-TRACKING

→ This concept was given by D.H. Lehmer, 1950
→ This concept provides partial candidate solution.
→ Backtracking algorithms try each possibility until they find right solution.
→ The key idea behind backtracking algorithm is the binary choice that means either yes or no.
→ Whenever algorithm has choice No that means problem reached at dead end
→ So we will backtrack one step behind & we will follow another path for

"If we have no peace, it is because we have forgotten that we belong to each other." —Mother Teresa

Scanned by CamScanner

choice Yes.

→ Backtracking is a type of DFS.

→ There may be more than one solution.

→ The search is successful if complete solution of the problem is obtained.

16/3/17

N-QUEEN'S PROBLEM

→ In the N-Queen problem ~~appro~~ follows the approach or methodology of backtracking.

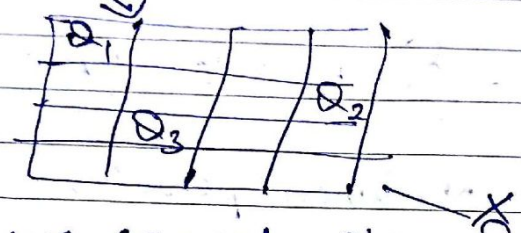
→ This problem states that we place queens in such a way on the chessboard so that no queen attacks each other horizontally, vertically or diagonally.

Note: Chess board may be 4x4, 8x8, 16x16 ... nxn
It should be square only.

Q. Consider the 4x4 chessboard and place the four queens in non attacking position using the concept of backtracking.

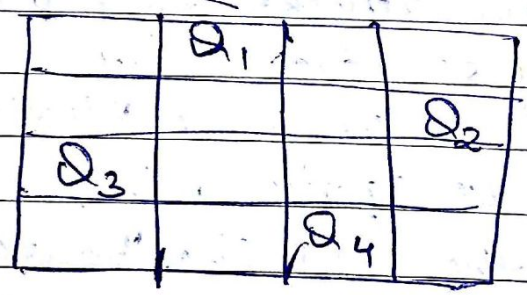
Q_1			

Scanned by CamScanner



Now we cannot place Q_4, so we back track till Q_1

Now,



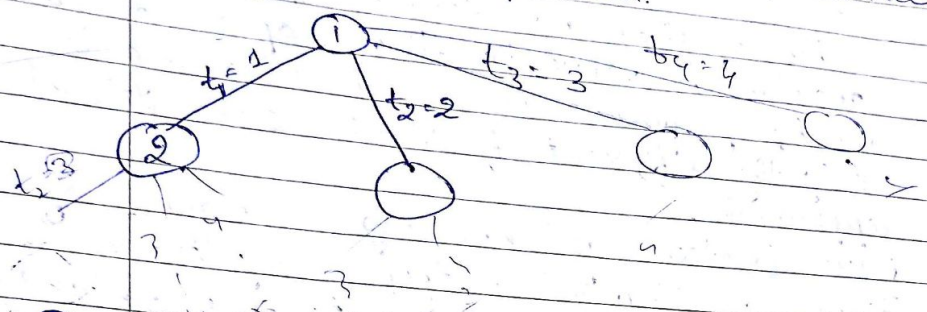
$\langle 2, 4, 1, 3 \rangle$ Ans.

Or $\langle 3, 1, 4, 2 \rangle$

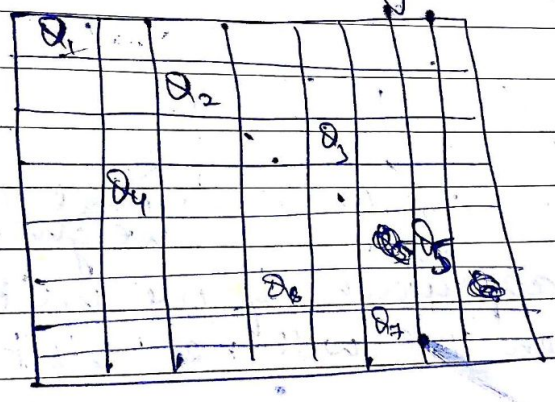
for the same problem we have more than one solution.

Scanned by CamScanner

Q. Draw the state space ~~phase~~ search ~~queen~~ tree for the 4 queen problem.



Q. Consider the 8 queens & place them on the chess board in non attacking position by using backtracking concept.



~~<1, 3, 5, 7, 2, 4, 6, 8>~~

<4, 6, 8, 2, 7, 1, 3, 5>

or <4, 8, 3, 7, 2, 6, 1, 5>.

"Do not say, 'It is morning,' and dismiss it with a name of yesterday.
See it for the first time as a newborn child that has no name." –Rabindranath Tagore

Chitra

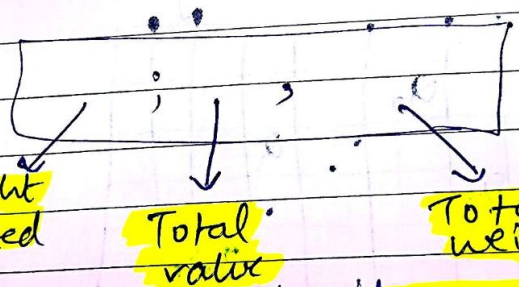
Scanned by CamScanner

Knapsack Problem using backtracking

This method we try to find the maximum value in the knapsack to solve the kp problem using backtracking is slightly different from the conventional approach.

Here, instead of choosing n number of items we choose n types of items $t_1, t_2 \ldots t_n$. W_i represents the weight & V_i represents the value.

In this approach the storage of node is as follows:



In this approach if we visited node (x, y) then we will not visit node ~~(y, x)~~

Q. Consider the following information

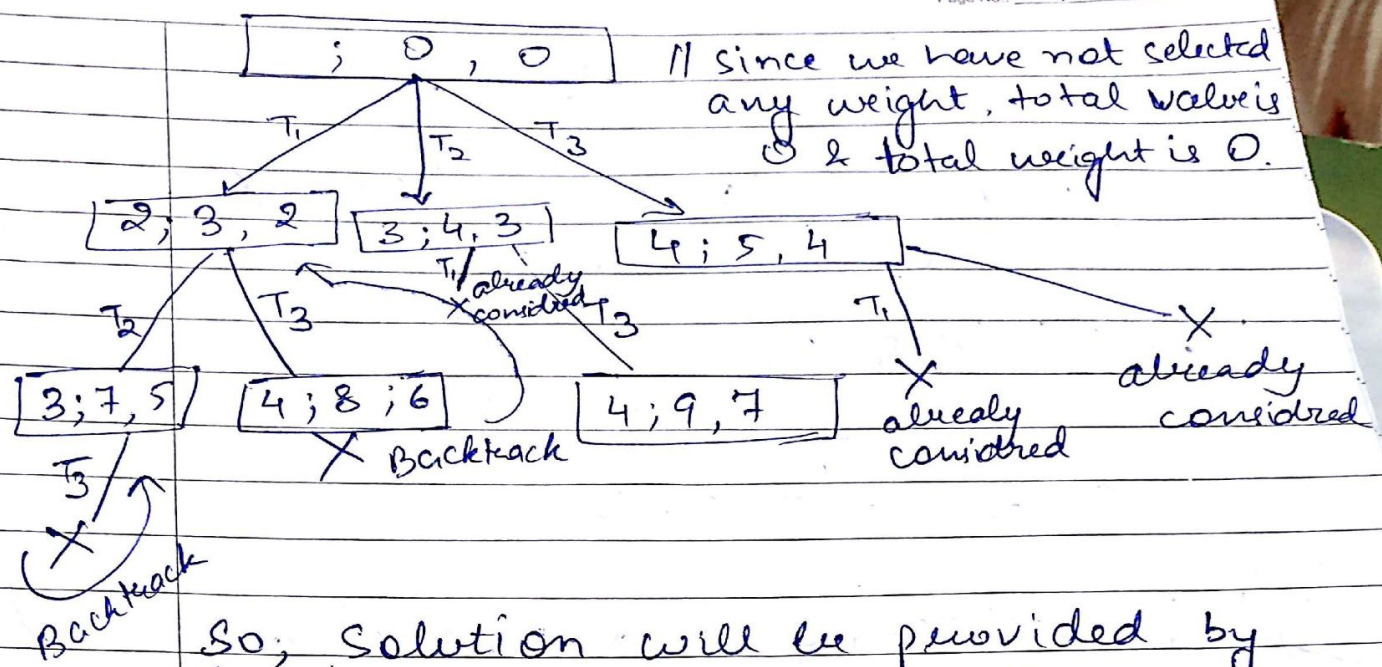
$T = \langle T_1, T_2, T_3 \rangle$

$W = \langle 2, 3, 4 \rangle$

$V = \langle 3, 4, 5 \rangle$

$W = 5$ → Total capacity of kp

Scanned by CamScanner



So, Solution will be provided by $\langle T_1, T_2 \rangle$. Total weight = 5
Total value = 7.

Q. Consider the following information for knapsack & solve it by using backtrack i

$T = \langle T_1, T_2\ T_3 \rangle$

$W = \langle 1, 4, 5 \rangle$

$V = \langle 4, 5, 6 \rangle$

$TW = 5$

"Do not say, 'It is morning,' and dismiss it with a name of yesterday. See it for the first time as a newborn child that has no name." —Rabindranath Tago

Scanned by CamScanner

23/3/17

=> Knapsack Problem using Branch & Bound technique

→ In this approach the nodes are specially structured in three parts

w	v
U_B	

w & v is the value & U_B is the upper bound.

The upper bound can be calculated by using following formula

$$U_B = v + (W - w)\left(\frac{V_{i+1}}{w_{i+1}}\right)$$

v → value of item

W → capacity of kp

w → weight of item

Q: Consider the following information about knapsack

$I = \langle I_1, I_2, I_3 \rangle$

$w = \langle 1, 2, 3 \rangle$

$v = \langle 2, 3, 4 \rangle$

$W = 3.$

Solu: To solve the kp problem using Branch & bound, we have to arrange all

Scanned by CamScanner

the elements in the ratio of $\frac{v_i}{w_i}$ in descending order

	w	v	v_i/w_i
I_1	1	2	2
I_2	2	3	15
I_3	3	4	133

Root node

$w = 0$ | $v = 0$

$U_B = 6$

$U_B = 0 + (3-0) \times 2 = 6$

IN for include

EX for exclude

IN=1 (1) | EX=1 (4)

$w = 1$ | $v = 2$ | $U_B = 5$

$2 + (3-1) \times 5 = 5$

$w = 0$ | $v = 0$ | $U_B = 5$

IN=2 (2) | EX=2 (3) | IN=2 (5) | EX=2 (6)

$w = 3$ | $v = 5$ | $U_B = 5$

X

$w = 4$

$w = 2$ | $v = 3$ | $U_B =$

$w = 3$ | $v = 4$ | $U_B = 4$

IN=3

X

$w = 5$

Since at the leaf node (last level) we found the the highest upper bound is 5 so answer will b as follows:

Chitra

"Do not say, 'It is morning,' and dismiss it with a name of yesterday.
See it for the first time as a newborn child that has no name." —Rabindranath Tagore

Scanned by CamScanner

$I = \langle I_1, I_2 \rangle$
$w = 3$
$v = 5$

Ans

Q. Consider the following information
$I = \langle I_1, I_2, I_3, I_4 \rangle$

	w	v
I_1	9	10
I_2	6	6
I_3	7	5
I_4	3	1

$W = 16.$

Q. Differentiate dead node, live node & e-node

Ans:

Scanned by CamScanner

Branch & Bound

very similar technique to the backtracking technique since both explore (expand) the graph & then it deals with optimum solution of a given problem. In B&B technology, at each step, we calculate the bound for the node & check that this bound is able to give optimum solution or not.
Bound maybe upper or lower; In knapsack, we try to find upper bound, whereas in TSP, we try to find lower bound.
TSP → travelling salesperson problem.
If we find that at any node the obtained solution is suitable but other nodes may give optimum solution, then leave this & try to find solution for an another node.

Properties of B&B

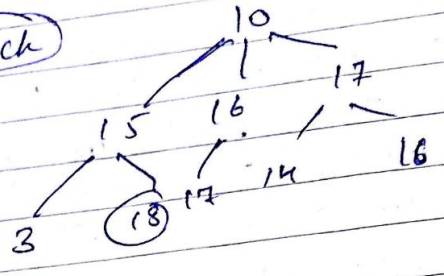
1. used only for optimization problem.
2. It does not stick on any particular way to explore the tree (BFS/DFS/Backtracking)
3. Backtracking always picks 1 single success but branch & bound always has the entire list of successors in queue.

"Do not say, 'It is morning,' and dismiss it with a name of yesterday.
See it for the first time as a newborn child that has no name." —Rabindranath Tagore

Scanned by CamScanner

U_B of parent < U_B of children (T/F) → True >

knapsack



To bound each node, we choose lower bound for minimization of problem & upper bound (U_B) for maximization of problem.

Scanned by CamScanner