20/4/17

MODULE - 4

DYNAMIC PROGRAMMING (Planning)

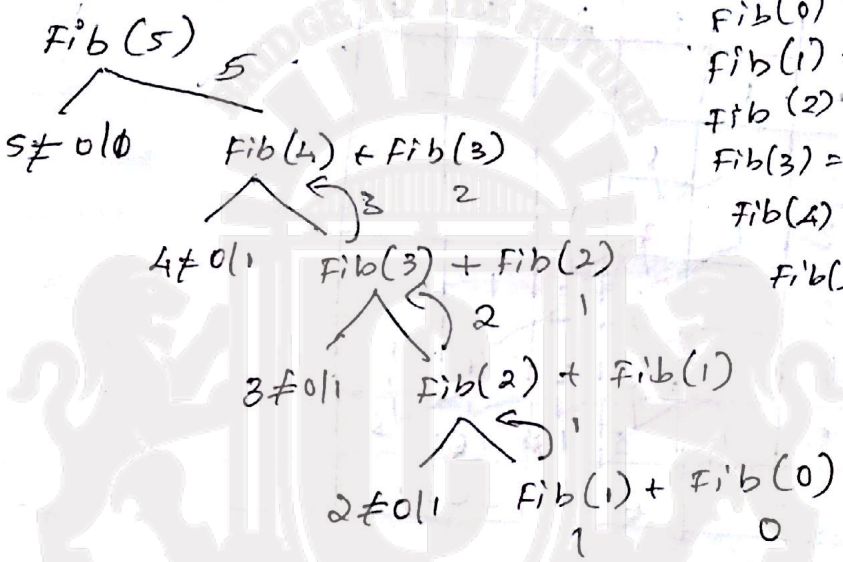
Richard Bellman

It is a general method for optimizing multistage decision process.

Dynamic programming

It is a technique for solving problem through overlapping subproblems. Each subproblem is generated through recurrence relation solving them only once and recording the result for future use.

Ex:- Fib(5)



Fib(0) = 0
Fib(1) = 1
Fib(2) = 1
Fib(3) = 2
Fib(4) = 3
Fib(5) = 5

$^1C_1$ $a+b = a+b$ (1, 1)

$^2C_0 + ^2C_1 + ^2C_2$ $(a+b)^2 = a^2 + 2ab + b^2$ (1, 2, 1)

$^3C_0 + ^3C_1 + ^3C_2 + ^3C_3$ $(a+b)^3 = a^3 + 3a^2b + 3ab^2 + b^3$ (1, 3, 3, 1)

$^4C_0 + ^4C_1 + ^4C_2 + ^4C_3 + ^4C_4$ $(a+b)^4 = a^4 + 4a^3b + 6a^2b^2 + 4ab^3 + b^4$ (1, 4, 6, 4, 1)

$$C(n,k) = \begin{cases} 1 & k=0 \text{ or } n=k \\ C(n-1,k-1) + C(n-1,k) & \text{otherwise} \end{cases}$$

C(5,3).

	0	1	2	3
0	1			
1	1	1		
2	1	2	1	
3	1	3	3	1
4	1	4	6	4
5	1	5	10	10

C(6,4)

	0	1	2	3	4
0	1				
1	1	1			
2	1	2	1		
3	1	3	3	1	
4	1	4	6	4	1
5	1	5	10	10	5
6	1	6	15	20	15

Algorithm :- Binomial-Coefficient (n,k)

// Input :- Positive integers n,k where n>=k>=0.

// output :- C(n,k) coefficient.

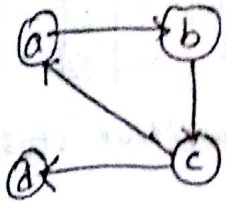
```
for i ← 0 to n do.
    for j ← 0 to min (i,k) do.
        if (j = 0 or i = j) then
            c[i,j] ← 1
        else
            c[i,j] ← c[i-1, j-1] + c[i-1, j]
        end if
    End for
End for
return (c(n,k))
```

22/4/13

Warshall's Algorithm

AND operation

Transitive closure of adjacency matrix



$R^{(0)}$ =

	a	b	c	d
a	0	1	0	0
b	0	0	1	0
c	1	0	0	1
d	0	0	0	0

	a	b	c	d
a	1	1	1	1
b	1	1	1	1
c	1	1	1	1
d	0	0	0	0

Transitive closure

→ Warshall Algorithm estimates transitive closure for the given adjacency matrix

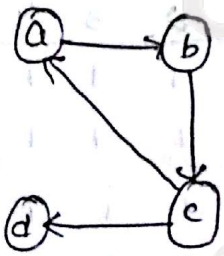
→ For any two vertices u and v of a given graph, if there is a direct path or via intermediate vertices then $R(u,v) = 1$ else zero

→ Initially $R^{(0)}$ is same as adjacency matrix

→ Estimate $R^{(1)}$ (vertex 1 as intermediate node), then $R^{(2)} \ldots R^{(n)}$

→ Declare $R^{(n)}$ as transitive closure

ex



$R^{(0)}$ =

	a	b	c	d
a	0	1	0	0
b	0	0	1	0
c	1	0	0	1
d	0	0	0	0

$R^{(a)}$ =

	a	b	c	d
a	0	1	0	0
b	0	0	1	0
c	1	1	0	1
d	0	0	0	0

$R^{(b)}$ =

	a	b	c	d
a	0	1	1	0
b	0	0	1	1
c	1	1	1	1
d	0	0	0	0

$R^{(c)}$

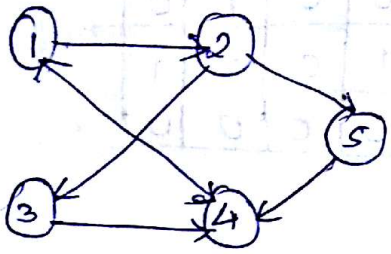
	a	b	c	d
a	1	1	1	1
b	1	1	1	1
c	1	1	1	1
d	0	0	0	0

$R^{(d)}$ =

	a	b	c	d
a	1	1	1	1
b	1	1	1	1
c	1	1	1	1
d	0	0	0	0

Transitive closure.

(2)



$R^{(0)}$ =

	1	2	3	4	5
1	0	1	0	0	0
2	0	0	1	0	1
3	0	0	0	1	0
4	1	0	0	0	0
5	0	0	0	1	0

$R^{(1)}$ =

	1	2	3	4	5
1	0	1	0	0	0
2	0	0	1	0	1
3	0	0	0	1	0
4	1	1	0	0	0
5	0	0	0	1	0

$R^{(2)}$ =

	1	2	3	4	5
1	0	1	1	0	1
2	0	0	1	0	1
3	0	0	0	1	0
4	1	1	1	0	1
5	0	0	0	1	0

$R^{(3)}$ =

	1	2	3	4	5
1	0	1	1	1	1
2	0	0	1	1	1
3	0	0	0	1	0
4	1	1	1	1	1
5	0	0	0	1	0

$R^{(4)}$ =

	1	2	3	4	5
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
5	1	1	1	1	1

$R^{(5)}$ =

	1	2	3	4	5
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
5	1	1	1	1	1

Algorithm Warshall (A[1...n, 1...n])

//Input : Adjacency matrix A[1...n, 1...n] of graph

//output : Transitive closure $R^{(n)}$

$R^{(0)} \leftarrow A$

for $k \leftarrow 1$ to n do

for $i \leftarrow 1$ to n do

for $j \leftarrow 1$ to n do

$R^{(k)}[i, j] \leftarrow R^{(k-1)}[i][j]$ OR

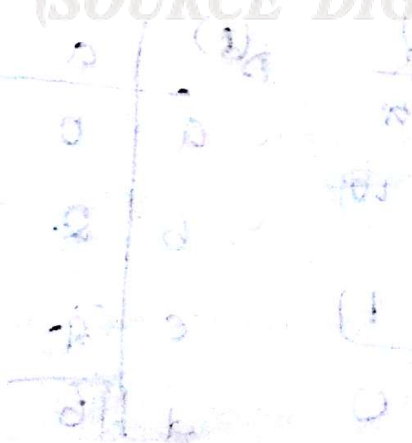
$(R^{(k-1)}[i, k]$ AND $R^{(k-1)}[k, j])$

end for

end for

end for

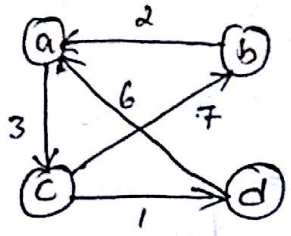
Return $R^{(n)}$



27/4/17

Floyd's Algorithm [All pair shortest Path]

(1)



We've to find the shortest path

$D^{(0)}$ =

	a	b	c	d
a	0	∞	3	∞
b	2	0	∞	∞
c	∞	7	0	1
d	6	∞	∞	0

shortest distance.

⇒ I/P

OR operation

$D^{(1)}$ =

	a	b	c	d
a	0	∞	3	∞
b	2	0	5	∞
c	∞	7	0	1
d	6	∞	9	0

→ fix dicagonal elemnt

→ add row and column and replace ∞ with smaller no of addition

$D^{(2)}$ =

	a	b	c	d
a	0	∞	3	∞
b	2	0	5	∞
c	9	7	0	1
d	6	∞	9	0

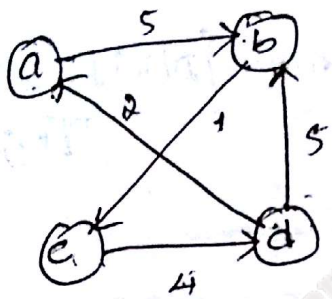
$D^{(3)}$

	a	b	c	d
a	0	10	3	4
b	2	0	5	6
c	9	7	0	1
d	6	16	9	0

$D^{(4)}$ =

	a	b	c	d
a	0	10	3	4
b	2	0	5	6
c	7	17	0	1
d	6	16	9	0

(2)



$= D^0 =$

	a	b	c	d
a	0	5	∞	∞
b	∞	0	1	∞
c	∞	∞	0	4
d	2	5	∞	0

$D^{(1)}$ =

	a	b	c	d
a	0	5	∞	∞
b	∞	0	1	∞
c	∞	∞	0	4
d	2	5	∞	0

$D^{(2)}$ =

	a	b	c	d
a	0	5	6	∞
b	∞	0	1	∞
c	∞	∞	0	4
d	2	5	6	0

$D^{(3)}$ =

	a	b	c	d
a	0	5	6	10
b	∞	0	1	5
c	∞	∞	0	4
d	2	5	6	0

$D^{(4)}$ =

	a	b	c	d
a	0	5	6	10
b	7	0	1	5
c	6	9	0	4
d	2	5	6	0

P. T: 0

Algorithm · Floyd (w[1....n]).

// Input : weight matrix w[1....n]

// output : D - shortest path.

```
D ← w.
for k ← 1 to n do //vertex
    for i ← 1 to n do //row
        for j ← 1 to n do //column
            D[i,j] ← min{D[i,j], D[i,k] + D[k,j]}
        end for
    endfor
endfor
return D
```

Time efficiency

$$T(n) = \sum_{k=1}^{n} \sum_{i=1}^{n} \sum_{j=1}^{n} 1$$

$$T(n) \in \Theta(n^3)$$

0/1 Knapsack using Dynamic Programming

n = no of items, m = capacity of Knapsack.

P[P1, P2, P3 .... Pn] w[w1, w2 .... wn]

Value table.

V	0	1	2	----	j	----	m
0	0	0	0	0	- - -	-	0
1	0						
2	0						
3	0						
:	:						
i	:						
:	:						
n	:						

i

i=0 & j=0

$$v[i,j] = \begin{cases} 0 & i=0 \text{ \& } j=0 \\ v[i-1,j] \text{ (previous)} & j < w_i \\ \max\{v[i-1,j],\ p[i]+v[i-1,j-w[i]]\} & j \geq w_i \end{cases}$$

doubt

28/4/17

Ex

$n=4 \quad m=5$

$p[12, 10, 20, 15] \quad w[2, 1, 3, 2]$

j → capacity

i ↓	V	0	1	2	3	4	5
	0	0	0	0	0	0	0
w=2, p=12	1	0	0				
w=1, p=10	2	0					
w=3, p=20	3	0					
w=2, p=15	4	0					

j →

i ↓	V	0	1	2	3	4	5
	0	0	0	0	0	0	0
w=2, p=12	1	0	0	max(0, 12+0) = 12	max(0, 12+0) = 12	max(0, 12+0) = 12	max(0, 12+0) = 12
w=1, p=10	2	0	max(0, 10+0) = 10	max(12, 10+0) = 12	max(12, 10+12) = 22	max(12, 10+12) = 22	max(12, 10+12) = 22
w=3, p=20	3	0	10	12	max(22, 20+0) = 22	max(22, 20+10) = 30	max(12, 20+12) = 32
w=2, p=15	4	0	10	max(12, 15+0) = 15	max(22, 15+10) = 25	max(30, 15+12) = 30	max(32, 15+12) = 37

Optimal solution is 37.

[∅ ∅ 0 ∅] = 1, 2, 4.
  1 1   1 = 37.

Example 2

$n=5$, $m=7$

$P[25, 20, 15, 40, 50]$, $w[3, 2, 1, 4, 3]$.

i \ j →	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
$w_1=3$ $p_1=25$	0	0	0	max(0, 25+0) 25	max(0, 25+0) 25	max(0, 25+0) 25	max(0, 25+0) 25	max(0, 25+0) 25
$w_2=2$ $p_2=20$	0	0	(0, 20+0) 20	(25, 20+0) 25	(25, 20+0) 25	(25, 20+25) 45	(25, 20+25) 45	(25, 20+25) 45
$w_3=1$ $p_3=15$	0	(0, 15+0) 15	(20, 15+0) 20	(25, 15+20) 35	(25, 15+25) 40	(45, 15+25) 45	(45, 15+45) 60	60
$w_4=4$ $p_4=40$	0	15	20	35	(40, 40+0) 40	(45, 40+15) 55	(60, 40+20) 60	(60, 40+35) 75
$w_5=3$ $p_5=50$	0	15	20	(35, 50+0) 50	(40, 50+15) 65	45, 50+20 70	60, 50+35 85	90

i ↓

Optimal sol^{n} = 90

[0 0 0 0 0]

02/5/17

Algorithm DP_Knapsack (n, m, P[ ], w[ ]).

//input : no. of items – n , knapsack_capacity = m

Prices of all items – P[1...n], weight of all items – w[1...n]

//output : optimal feasible solution v[n, m]

```
for i ← 0 to n do
    for j ← 0 to m do
        if (i=0 or j=0) then
            v[i,j] ← 0
        else if (j < w[i]) then
            v[i,j] ← v[i-1,j]
        else
```

$V[i,j] \leftarrow \max(V[i-1,j], P[i] + V[i-1, j-w[i]])$

end if

end for

end for.

return $V[n,m]$

Time efficiency

$T(n) \in \Theta(n,m)$

Memory function Knapsack

$n=4, \quad m=5$.

$P = \{12, 10, 20, 15\}$ $\quad w = \{2, 1, 3, 2\}$.

It is an optimised DP- knapsack problem. It solves only those set of sub-problems which yields optimal solution $V[n]$. And rest of the other subproblems are not been solved.

Algorithm MF_Knapsack (i, j)

//input : 2 non-negative integer i, j

where i denotes item no and j denotes capacity of knapsack

//output : optimal feasible solution ie $V[n,m]$

//note : initialize V[][] with -1 for its cells except 0^{th} row and 0^{th} column

0	0	0	0	0
0	-1	-1	-1	-1
0	-1	-1	-1	-1
0	-1	-1	-1	-1

if $(V[i,j] < 0)$ then

if $(j < w_i)$ then

$V[i,j] \leftarrow$ MF_Knapsack $(i-1, j)$

else

~~return (max(MF_knapsack(~~

return (max(MF_knapsack(i-1, j), P_i + MF_knapsack(i-1, j - w_i))

end if

end if

~~En~~

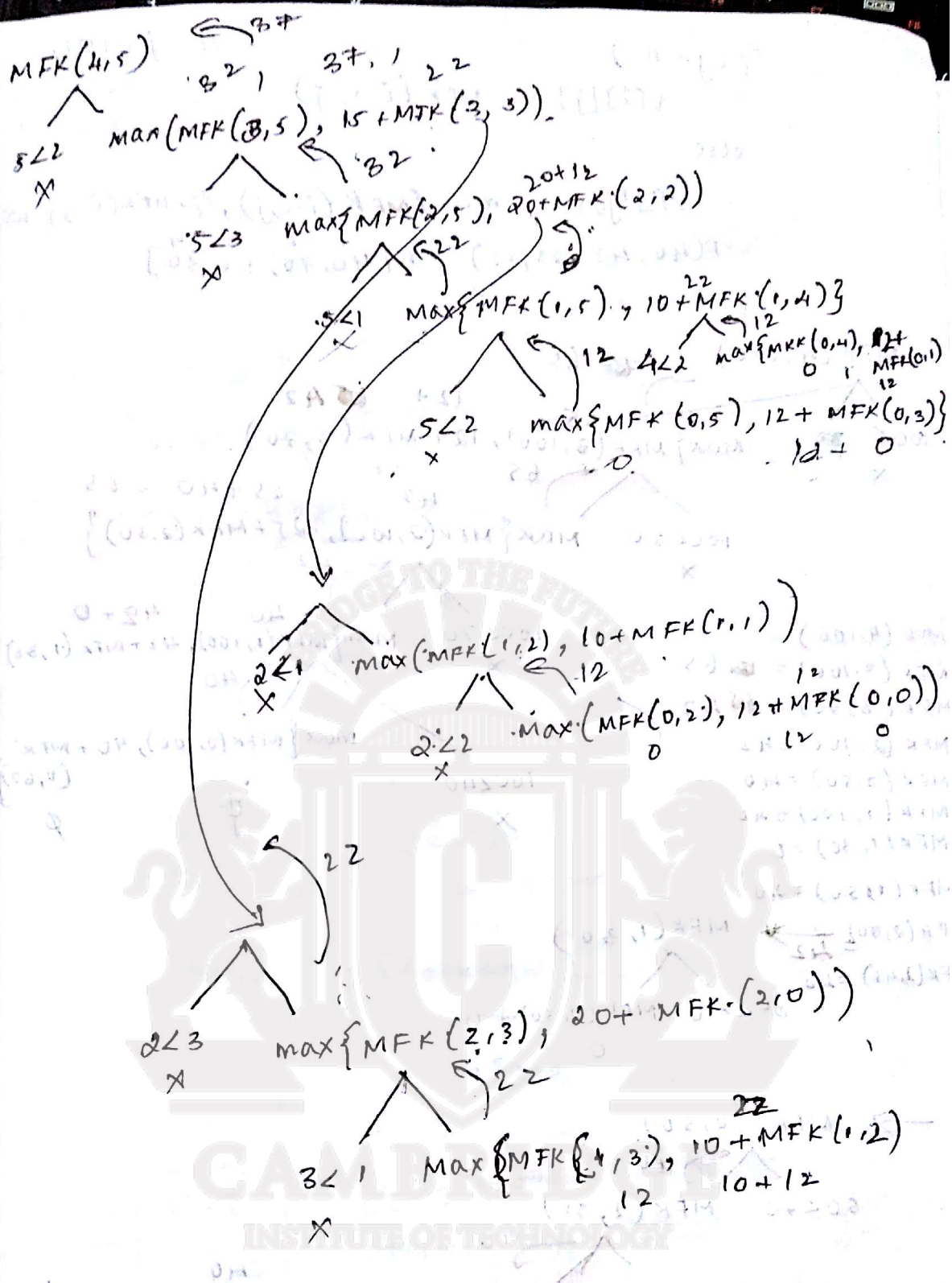
Example n = 4, m = 5

P = {12, 10, 20, 15} w = {2, 1, 3, 2}

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	-1 ~~12~~	-1 12	-1 12	-1 12	-1 12
2	0	~~0~~ -1	-1 12	-1	-1	-1 22
3	0	-1	-1	-1 22	-1	-1 32
4	0	-1	-1	-1	-1	-1 37

MFK (4, 5)

~~P = 0.~~



3/5/17

Solve the given knapsack problem using Dynamic programming technique where

$n = 4$, $m = 100$ = capacity

$P[40, 42, 25, 12]$ $w[40, 70, 50, 30]$

i=4, j=100

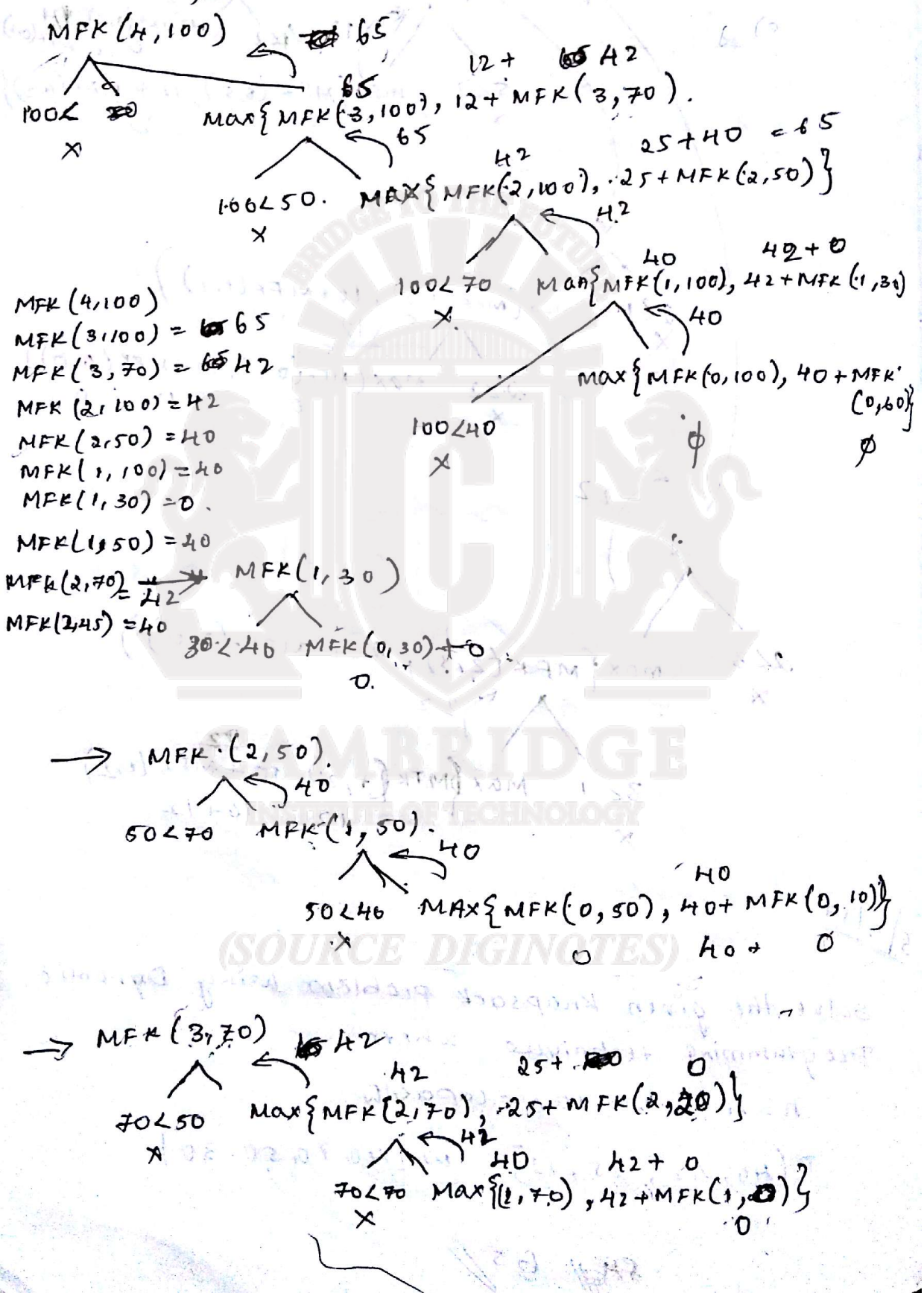
if (j < w[i])

v[i][j] ← MFK(i-1, j)

else

v[i][j] ← max(MFK(i-1, j), P[i] + MFK(i-1, j-w[i]))

P(40, 42, 25, 12) w[40, 70, 50, 30]



MFK(1,70) = 40

MFK(1,0) = 0

→ MFK(1,70) ← 40

70 < 40 ✗ MAX{MFK(0,70), 40 + MFK(0,30)}

40 + 0

→ ~~MFK(1,28)~~ ← 0

~~28 < 40 MFK(0,28)~~

0

→ ~~MFK(1,45)~~ ← 40

~~45 < 40 MFK(1,45)~~ ← 40

~~45 < 40 MAX{MFK(0,45) → 40 + MFK(0,5)}~~

0 40 + 0

40 0

→ MFK(1,20)

20 < 70 MFK(1,20) ← 0

20 < 40 MFK(0,20)

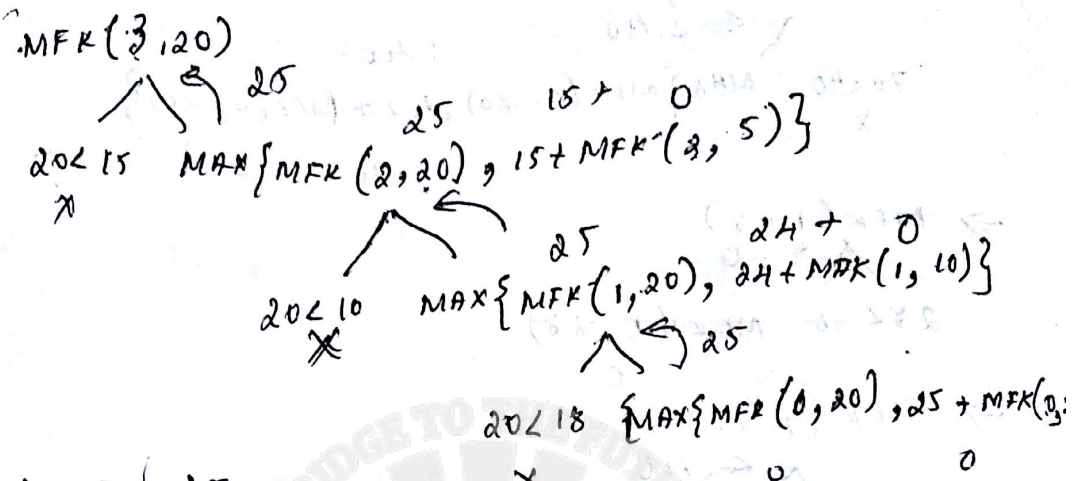
0

V	0	20	30	50	70	100
0	0	0	0	0	0	0
1	0	∞	∞	40	40	40
2	0	0	-1	40	42	42
3	0	-1	-1	-1	42	65
4	0	-1	-1	-1	-1	65

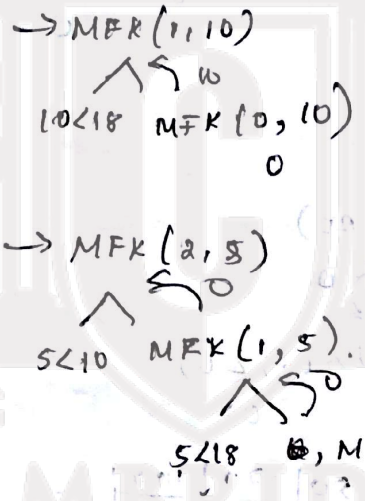
ex 3

n = 3 , m = 20 ;

P[25, 24, 15] w[18, 10, 15]



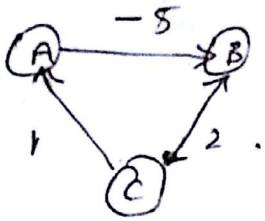
MFK(3,20) = 25
MFK(1,20) = 25
MFK(1,10) = 0.
MFK(2,20) = 25
MFK(2,5) = 0
MFK(1,5) = 0



	0	2	5	10	20
0	0	0	0	0	0
1	0	-1	0	0	25
2	0	-1	0	-1	25
3	0	-1	-1	-1	25

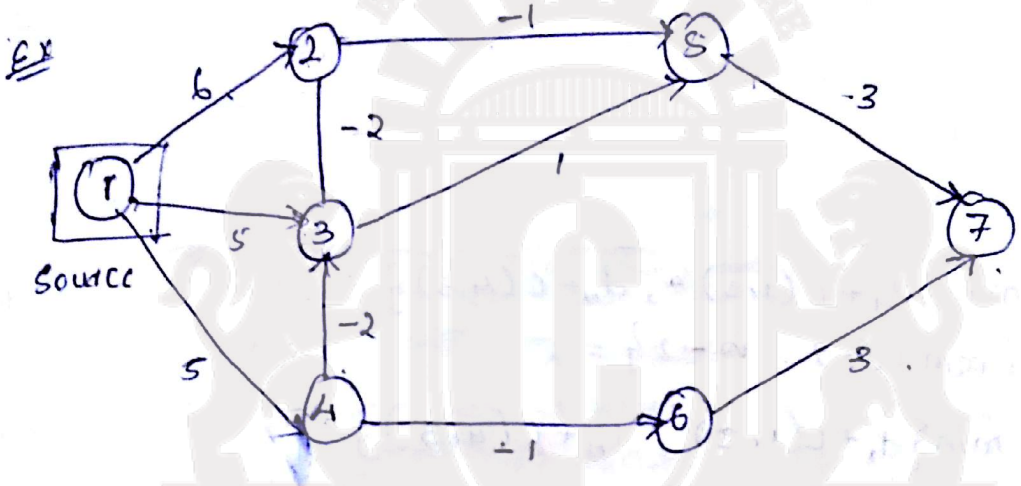
Bellman Ford Algorithm

→ single source shortest path.
→ used in routing protocol - Distance vector
→ works on the negative weights.
[cannot work on -ve cycle].



Estimate shortest path based on neighbour shortest path from source.

Ex



Iterations	d_1	d_2	d_3	d_4	d_5	d_6	d_7	
Initial.	0	∞	∞	∞	∞	∞	∞	
1	0	6	5	5	∞	∞	∞	
2	0	3	3	5	5	4	∞	
3	0	1	3	5	0	4	7	
4	0	1	3	5	0	4	5	
5	0	1	3	5	0	4	3	
6	0	1	3	5	0	4	3	
negative cycle check	0	1	3	5	0	4	3	Same value no negative sign

(1) $d_2 = \min\{d_1 + c(1,2),\ d_3 + c(3,2)\}$

$= \min\{0+6,\ \infty - 2\} = 6$.

(2) $d_2 = \min\{d_1 + c(1,2),\ d_3 + c(3,2)\}$.

$\min\{0+6,\ 5 + (-2)\} = 3$

(3) $d_2 = \min\{0+6,\ 3-2\} = 1$

(4) $d_2 = \min\{0+6,\ 3-2\} = 1$

(5) $d_2 = \min\{0+6,\ = 1$

(6) $d_2 = 1$

---

(1) $d_3 = \min\{d_1 + c(1,3),\ d_4 + c(4,3)\}$

$= \min\{0+5,\ \infty - 2\} = 5$

(2) $d_3 = \min\{d_1 + c(1,3),\ d_4 + c(4,3)\}$

$\min\{0+5,\ 5-2\} = 3$

(3) $d_3 = \min\{0+5,\ 5-2\} = 3$

(4) $d_3 = \min\{0+5,\ 5-2\} = 3$

(5) $d_3 = 3$

(6) $d_3 = 3$

$d_4 = \min\{d_1 + c(1,4)\}$

$= 0 + 5 = 5$

$d_4 = d_1 + c(1,4)$

$0 + 5 = 5$

$d_4 = 0 + 5 = 5$

$d_4 = 0 + 5 = 5$

$d_4 = 0 + 5 = 5$

$d_4 = 0 + 5 = 5$.

$d_5 = \min\{d_2 + c(2,5), d_3 + c(3,5)\}$

$\infty + \infty + \epsilon), \infty + 1 = \infty$.

$d_5 = \min\{d_2 + c(2,5), d_3 + c(3,5)\}$

$\min\{6 - 1, 5 + 1\} = 5$.

$d_5 = \min\{3 - 1, 3 + 4\} = 2$.

$d_5 = \min\{2 - 1, 3 + 4\} = 0$.

$d_5 = 0$

$d_5 = 0$

$d_6 = d_4 + c(4,6) =$

$\infty + \epsilon) = \infty$

$d_6 = d_4 + c(4,6)$

$= 5 + 1 = 4$

$d_6 \quad 5 - 1 = 4$

$d_7 = \min\{d_5 + c(5,7), d_6 + c(6,7)\}$.

$\{\infty + 3, \infty + 3\} = \infty$

$d_7 = \min\{d_5 + c(5,7), d_6 + c(6,7)\}$

$\min\{\infty + 3, \infty + 3\} = \infty$

$d_7 = \min\{5 + 3, 4 + 3\} = 7$

$d_7 = \min\{2 + 3, 4 + 3\} = 5$

$d_7 = \min\{0 + 3, 4 + 3\} = 3$

$d_7 = 3$

Shortest path

(1) $1 \to 1 = 0$

(2) $1 \to 4 \to 3 \to 2 = -1$

(3) $1 \to 4 \to 3 = 3$

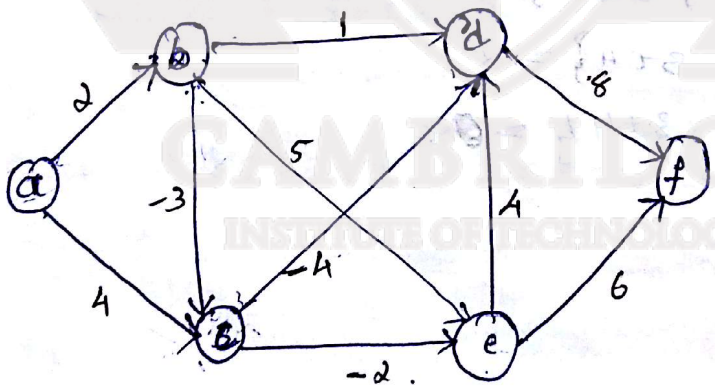
(4) $1 \to 4 = 5$

(5) $1 \to 4 \to 3 \to 2 \to 5 = 0$

(6) $1 \to 4 \to 6 = 4$

(7) $1 \to 4 \to 3 \to 2 \to 5 \to 7 = 3$

2.



Shortest path

(a) $a \to a = 0$

(b) $a \to b = 2$

(c) $a \to b \to c = -1$

(d) $a \to b \to c \to d = -5$

(e) $a \to b \to c \to e = -3$

(f) $a \to b \to c \to e \to f = 3$

Iterations	d_a	d_b	d_c	d_d	d_e	d_f	d
Initial	0	∞	∞	∞	∞	∞	
1	0	2.	4.	∞	∞	∞	
2	0	2	−1	0	2	∞	
3	0	2	−1	−5	−3	8	
4	0	2	−1	−5	−3	3	
5	0	2.	−1	−5	−3	3	
Neg cycle check	0	2	−1	−5	−3	3	

$d_B = \min d_A + c(a,b)$

$= 0 + 2 = 2.$ } It is set.

$d_c = \min\{d_a + c(a,c), d_b + c(b,c)\}$.

$\{0 + 4, \infty + (-3)\} = 4.$ It is set

$d_c = \min\{0+4, 2-3\} - 1$

$d_d = \min\{d_B + c(b,d), d_c + c(c,d), d_e + c(e,d)\}$

$\{\infty, \infty, \infty\} = \infty.$

$d_d = \min\{2+1, 4-4, \infty\} = 0$

$d_d = \min\{3, -1-4, 2+4\} = -5$ { set

$d_e = \min\{d_b + c(b,e), d_c + c(c,e)\}$.

$\{\infty, \infty\} = \infty$

$d_e = \min\{d_b + c(b,e), d_c + c(c,e)\}$

$\{2+5, 4-2\} = 2.$

$d_e = \min\{2+5, -1-2\} = -3$

set

$d_f = \min\{ d_d + c(d,f) , d_e + c(e,f) \}$

$\infty \quad , \quad \infty \quad = \infty$

$d_f = \min\{ \infty , \infty \} = \infty$

$d_f = \min\{ 0+6 , 2+6 \} = 8$

$d_f = \min\{ -5+8 ; -3+6 \} = 3$. get

Algorithm : Bellman_Ford (v, cost[i, j], dist[ ], n)

//input : source vector -v, Graph in cost matrix ~~cost[1...n, 1...n]~~.

-cost[1...n, 1...n], shortest distance dist[],

no of vertices -'n'

//output : shortest distance from source vertices -v ie dist[]

for i ← 1 to n do // first iteration

dist[i] ← cost[v, i].

end for

for k ← 1 to n-2 do

for each vertex u such that $u \neq v$ and u has atleast one incoming edge do

~~if~~

~~for each~~

for each {i, u} in graph do

if dist[u] > dist[i] + cost[i,u] then

dist[u] ← dist[i] + cost[i,u]

end if

end for

end for

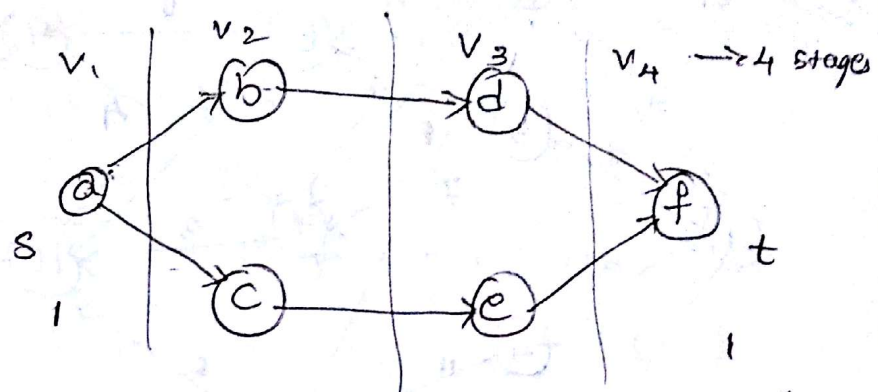
end for

return dist[]

{i, u} → is an edge coming to an vertex.

Multistage graph

A graph :



→ A multi-stage graph G is a directed graph in which the vertices are partitioned into 'K' disjoint sets. as.

$G = \{V, E\}$ as V_i where $1 \le i \le x$.

& $K \ge 2$.

→ And if $\{u, v\}$ is an edge in 'E' then, $u \in V_i$ and $v \in V_{i+1}$, $|V_1| = |V_K| = 1$

→ Let $s \in V_1$ & $t \in V_K$.

s → source vertex

t → sink vertex

→ The cost of path from s to t is the sum of the cost of the edges on the path. The MSG problem is to find minimum cost path from s to t.

Every V_i denotes a stage

V_1 — stage 1

V_2 — stage 2

⋮

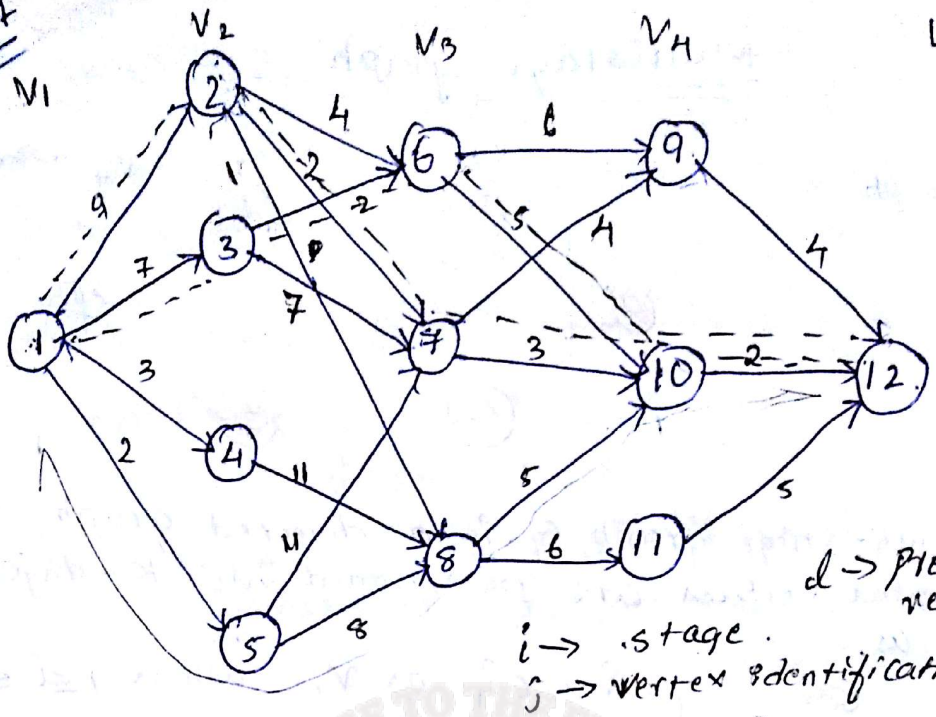
V_K — stage K.

∴ It is called as multistage graph.

Cost → ~~cost~~ shortest path

C → connectivity cost.

11/5/17



d → Preceeding vert

i → stage.

j → vertex identification.

$cost(i, j) = \min \{ c(j, l) + cost(i+1, l) \}$

$l \in V_{i+1}$

$(j, l) \in E$

Stage 4

$cost(4, 9) = c(9, 12) = 4$ $\quad d[9] = 12$

$cost(4, 10) = c(10, 12) = 2$ $\quad d[10] = 12$

$cost(4, 11) = c(11, 12) = 5$ $\quad d[11] = 12.$

Stage 3

$cost(3, 6) = \min \{ c(6, 9) + cost(4, 9),$

$c(6, 10) + cost(4, 10) \}$

$= \min \{ 6+4, 5+2 \}$

$= 7.$ $\quad d[6] = 10$

$cost(3, 7) = \min \{ c(7, 9) + cost(4, 9),$

$c(7, 10) + cost(4, 10) \}$ $\quad d[7] = 10$

$\min \{ 4+4, 3+2 \}$

$= 5$

$cost(3, 8) = \min \{ c(8, 10) + cost(4, 10),$

$c(8, 11) + cost(4, 11) \}$

$\min \{ 5+2, 6+5 \} = 7$ $\quad d[8] = 10.$

Stage 2

$cost(2,2) = \{ c(2,6) + cost(3,6),\ c(2,7) + cost(3,7),$
$c(2,8) + cost(3,8) \}$

$4+7,\ 2+5,\ 1+7\}$

$= 7. \Rightarrow d[2] = 7.$

$cost(2,3) = \min\{ c(3,6) + cost(3,6),$
$c(3,7) + cost(3,7)\}$

$\min\{2+7,\ 7+5\}$

$= 9. \quad d[3] = 6.$

$cost(2,4) = c(4,8) + cost(3,8).$

$11 + 7 = 18 \quad d[4] = 8.$

$cost[2,5] = \min\{c(5,7) + cost(3,7),\ c(5,8) + cost(3,8)\}$

$\min\{11+5,\ 8+7\}$

$= 15 \quad d[5] = 8$

Stage 1

$cost(1,1) = \min\{c(1,2) + cost(2,2),\ c(1,3) + cost(2,3),$
$c(1,4) + cost(2,4),\ c(1,5) + cost(2,5)\}$

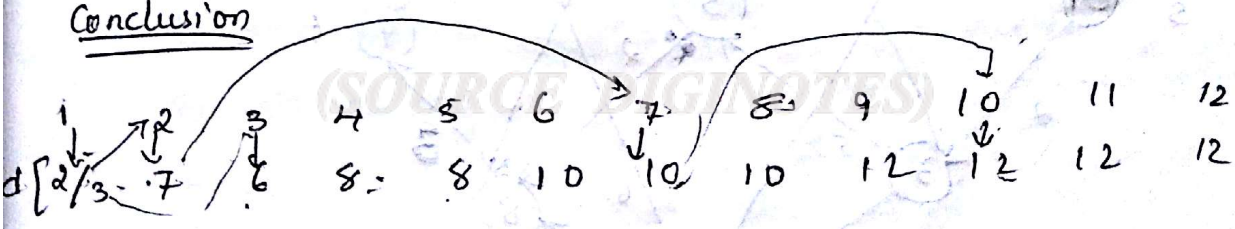
$d[1] = 2/3$

$= \min\{9+7,\ 7+9,\ 3+18,\ 2+15\}$

$16 \quad 16$

$= 16.$

Conclusion



Path ① $1 \rightarrow 2 \rightarrow 7 \rightarrow 10 \rightarrow 12$.

Path ② $1 \rightarrow 3 \rightarrow 6 \rightarrow 10 \rightarrow 12$.

Algorithm FGraph (G, k, n, P[1....k])

// Input : Graph 'G = {V, E}, k → no of stages,
n - no of vertices, ~~min-cost~~
min-cost path - P[1....k]

output : minimum-cost path P[1....k].

cost[n] = 0.0;

for j ← n-1 downto 1 do.

minimum // Let r be a vertex such that <j, r> is edge of G and c[j, r] + cost[r] is minimum, then.

cost[j] ← c[j, k] + cost[r]

d[j] ← r

end for

P[1] ← 1, P[k] ← n.

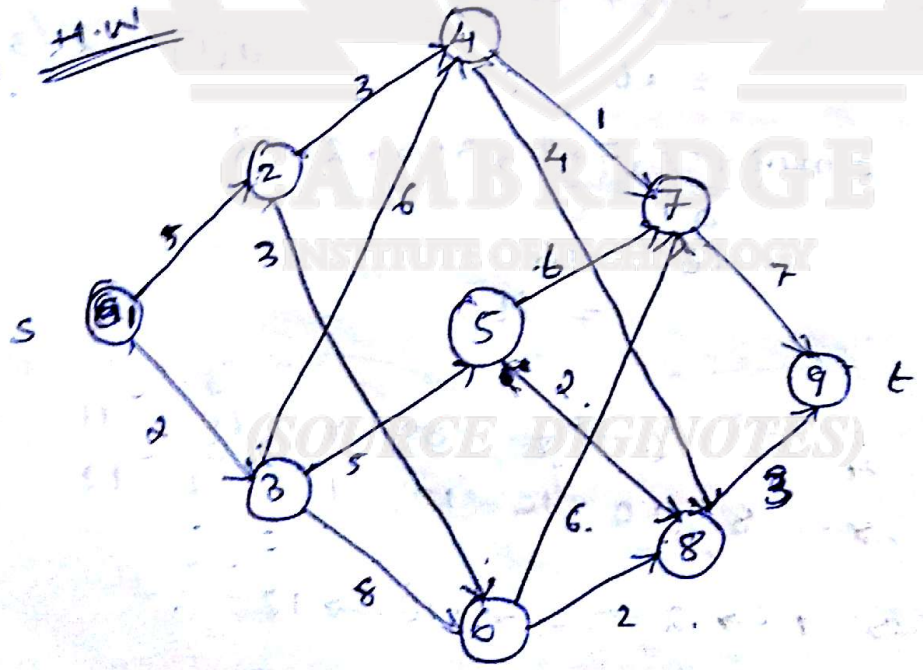
for j ← 2 to k-1 do.

P[j] ← d[p[j-1]]

end for

return P.

H.W

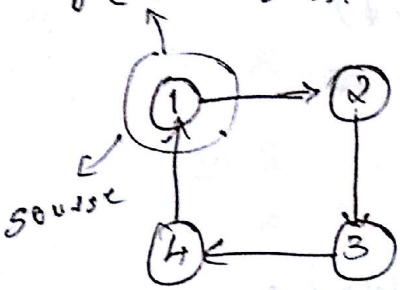


12/05/19

Travelling Salesperson problem.

$G = \{V, E\}$

$$g(1, V-\{1\}) = \min_{2 \le k \le n} \{c_{1k} + g(k, V-\{1,k\})\}$$



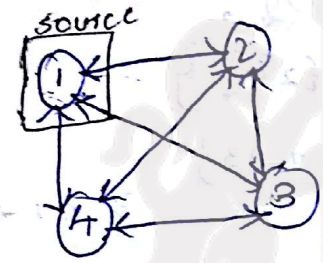
$i \rightarrow$ source

generalized eq^{-n}

$$g(i, s) = \min_{j \in s} \{c_{ij} + g(j, s-\{j\})\}$$

$$g(i, \phi) = c_{i1}$$

Ex



Cost matrix

$$\begin{bmatrix} 0 & 10 & 15 & 20 \\ 5 & 0 & 9 & 10 \\ 6 & 13 & 0 & 12 \\ 8 & 8 & 9 & 0 \end{bmatrix}$$

Zero intermediate Vertices to Source

$$c(2, \phi) = c_{21} = 5$$

$$c(3, \phi) = c_{31} = 6.$$

$$g(4, \phi) = c_{41} = 8.$$

1 Intermediate Vertex to source

$$g(2, \{3\}) = c_{23} + g(3, \phi)$$
$$= 9 + 6 = 15$$

$$g(2, \{4\}) = c_{24} + g(4, \phi)$$
$$= 10 + 8 = 18$$

$$g(3, \{2\}) = c_{32} + g(2, \phi)$$
$$= 13 + 5 = 18$$

$g(3,\{4\}) = c_{34} + g(4,\phi)$

$= 12 + 8 = 20$

$g(4,\{2\}) = c_{42} + g(2,\phi)$

$= 8 + 5 = 13$

$g(4,\{3\}) = c_{43} + g(3,\phi)$

$9 + 6 = 15$

2 - intermediate vertex

$g(2,\{3,4\}) = \min\{c_{23} + g(3,\{4\}),$

$c_{24} + g(4,\{3\})\}$

$= \min\{9+20, 10+15\}$

$= 25$ $\quad 2 \to 4 \to 1$

$g(3,\{2,4\}) = \min\{c_{32} + g\{2,\{4\}\}\},$

$c_{34} + g\{4,\{2\}\}\}$

$\min\{13+18, 12+13\}$ $\quad 3 \to 4 \to 1$

$= 25$

$g(4,\{2,3\}) = \min\{c_{42} + g(2,\{3\}), c_{43} + g(3,2)\}$

$= \min\{8+15, 9+18\}$

23 $\quad 4 \to 2$

3-intermediate vertex

$g(1,\{2,3,4\}) = \min\{c_{12} + g\{2,\{3,4\}\}, c_{13} + g(3,\{2,4\})$

$c_{14} + g\{4,\{2,3\}\}\}$

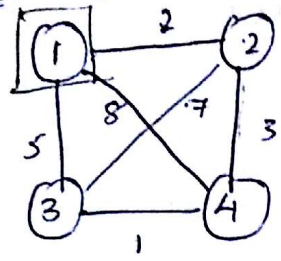
$= \min\{10+25, 15+25, 20+23\}$

$= \min\{35, 40, 43\}$

$1 \to 2 \to 4 \to 3 \to 1$

$10 + 10 + 9 + 6 = 35$

Ex 2 source -



It is a symmetric matrix

0 - intermediate vetex.

$c(2, \phi) = 2$.

$c(3, \phi) = 5$

$c(4, \phi) = 8$.

1 - intermediate vertex.

$g[2, \{4\}] = c_{24} + g(4, \phi)$

$= 3 + 8 = 11$.

$g(2, \{3\}) = c_{23} + g(3, \phi)$

$= 7 + 5 = 12$.

$g(3, \{2\}) = c_{32} + g(2, \phi)$.

$= 7 + 2 = 9$.

$g(3, \{4\}) = c_{34} + g(4, \phi)$

$= 1 + 8 = 9$.

$g\{4, \{2\}) = c_{42} + g(2, \phi)$

$= 3 + 2 = 5$

15/8/17

Optimal Binary Search Tree

Dictionary - Unique elements (key)

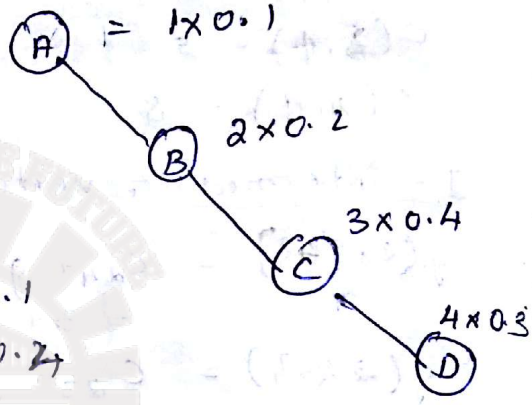
$a_1, a_2 \ldots\ldots a_n$ (Ascending order).

$P_1, P_2 \ldots P_n$ (Probability search)

To build BST - minimum no of Average comparision.

$P_i =$

A	B	C	D
0.1	0.2	0.4	0.3



A = $1 \times 0.1 = 0.1$

B = $2 \times 0.2 = 0.4$

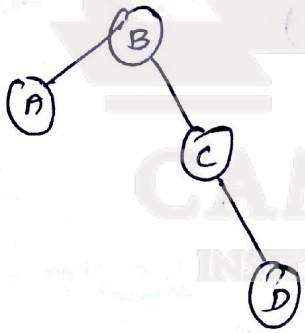
C = $3 \times 0.4 = 1.2$

D = $4 \times 0.3 = 1.2$

2.9

minimum no of Av.

(2)



A = $2 \times 0.1 = 0.2$

B = $1 \times 0.2 = 0.2$

C = $2 \times 0.4 = 0.8$

D = $3 \times 0.3 = 0.9$

2.1

~~n+1 →~~

Average value table

n+1 → rows [1 ... n+1]

n+1 → column [0 ... n]

C	0	1	2	3	4
1	0	0.1	0.4	1.1	1.7
2		0	0.2	0.8	1.4
3			0	0.4	1.0
4				0	0.3
5					0

Root table

R	0	1	2	3	4
1		1	2	3	3
2			2	3	3
3				3	3
4					4
5					

$R[i, i] \leftarrow i$

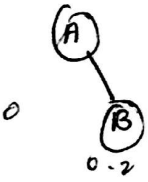
Initially $c[i, i-1] \leftarrow 0$.

$$c[i,j] = \min_{i<k\leq j} \{ \underbrace{c(i,k-1)}_{\text{Left sub tree}} + \underbrace{c(k+1,j)}_{\text{Right sub tree}} \} + \sum_{s=i}^{j} P_i$$

$$c[1,2] = \min_{k=1} \{ c[1,0] + c[2,2], \underset{k=2}{c[1,1] + c[3,2]} \} + P_1 + P_2$$

$$= \min \{ 0 + 0.2, 0.1 + 0 \} + 0.1 + 0.2$$

$$= 0.4.$$



$$c[2,3] = \min_{k=2} \{ c[2,1] + c[3,3], \underset{k=3}{c[2,2] + c[4,3]} \} + P_2 + P_3$$

$$= \min \{ 0 + 0.4, 0.2 + 0 \} + 0.2 + 0.4$$

$$= 0.8$$

$$c[3,4] = \min_{k=3} \{ c[3,2] + c[4,4], \underset{k=4}{c[3,3] + c[5,4]} \} + P_3 + P_4$$

$$\min \{ 0 + 0.3, 0.4 + 0 \} + 0.4 + 0.3$$

$$\Rightarrow 1.0$$

Next Diagonal.

$$c[1,3] = \min_{k=1} \{ c[1,0] + c[2,3], \underset{k=2}{c[1,1] + c[3,3]}, c[1,2] + c[4,3] \} + P_1 + P_2 + P_3$$

$$= \min \{ 0 + 0.8, 0.1 + 0.4, \underset{k=3}{0.4 + 0} \} + 0.1 + 0.2 + 0.4$$

$$= 1.1$$

$k=2$ $k=3$

$$c[2,4] = \min\{c[2,1]+c[3,4],\ c[2,2]+c[4,4],\ c[2,3]+c[5,4]\} + P_2+P_3+P_4$$

$k=4$

$$= \min\{0+1.0,\ 0.2+0.3,\ 0.8+0\} + 0.2+0.4+0.3$$

$$= 1.4$$

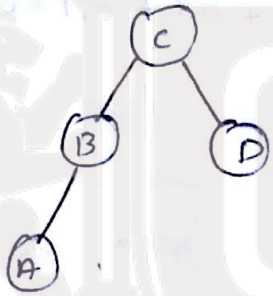
$k=1$ $k=2$ $k=3$

$$c[1,4] = \min\{c[1,0]+c[2,4];\ c[1,1]+c[3,4],\ c[1,2]+c[4,4],\ c[1,3]+c[5,4]\} + P_1+P_2+P_3+P_4.$$

$k=4$.

$$= \min\{0+1.4,\ 0.1+1.0,\ 0.4+0.3,\ 1.1+0\} + 1.0$$

1.7.



26/5/17

2.

	A	B	C	D	E
P	0.1	0.3	0.2	0.1	0.3

Average value	0	1	2	3	4	5
1	0	0.1	0.5	0.9	1.2	2.0
2		0	0.3	0.7	1.0	1.7
3			0	0.2	0.4	1.0
4				0	0.1	0.5
5					0	0.3
6						0

	0	1	2	3	4	5
1		1	2	2	2	3
2			2	2	2/3	3
3				3	3	5
4					4	5
5						5
6						

$K=1$

$K=2$

$c[1,2] = \min\{c[1,0]+c[2,1],\ c[1,1]+c[3,2]\} + P_1 + P_2$

$= \min\{0+0.3,\ 0.1+0\} + 0.1 + 0.3 = 0.5$

$K=2$ $K=3$

$c[2,3] = \min\{c[2,1]+c[3,3],\ c[2,2]+c[4,3]\} + P_2 + P_3$

$= \min\{0+0.2,\ 0.3+0\} + 0.3 + 0.2$

$= 0.7$

$K=3$ $K=4$

$c[3,4] = \min\{c[3,2]+c[4,4],\ c[3,3]+c[5,4]\} + P_3 + P_4$

$= \min\{0+0.1,\ 0.2+0\} + 0.2 + 0.1$

$= 0.4$

$K=4$ $K=5$

$c[4,5] = \min\{c[4,3]+c[5,5],\ c[4,4]+c[6,5]\} + P_4 + P_5$

$= \min\{0+0.5,\ 0.1+0\} + 0.1 + 0.3$

$K=1$ $K=2$ $K=3$

$c[1,3] = \min\{c[1,0]+c[2,3],\ c[1,1]+c[3,3],\ c[1,2]+c[4,3]\} + P_1 + P_2 + P_3$

$= \min\{0+0.7,\ 0.1+0.2,\ 0.5+0\} + 0.1 + 0.3 + 0.2$

$= 0.9$

Ally $c[2,4] = 1.0$.

$\therefore\ c[3,5] = 1.0$.

$$C[1,4] = \min\{ c[1,0] + c[2,4], c[1,1] + c[3,4], c[1,2] + c[4,4], c[1,3] + c[5,4] \} + P_1 + P_2 + P_3 + P_4$$

$$= \min\{0 + 1.0, 0.1 + 0.4, 0.5 + 0.1, 0.9 + 0\} + 0.7$$

$$= 1.2$$

$$C[2,5] = \min\{ c[2,1] + c[3,5], c[2,2] + c[4,5], c[2,3] + c[5,5], c[2,4] + c[5,5] \}$$

$$= \min\{0 + 1.0, 0.3 + 0.5, 0.7 + 0.3, 1.0 + 0\} + 0.9$$

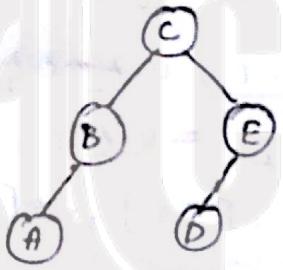
$$= 1.7$$

$$C[1,5] = \min\{ c[1,0] + c[2,5], c[1,1] + c[3,5], c[1,2] + c[4,5], c[1,3] + c[5,5], c[1,4] + c[6,5] \}$$

$$+ P_1 + P_2 + P_3 + P_4 + P_5$$

$$= \{0 + 1.7, 0.1 + 1.0, 0.5 + 0.5, 0.9 + 0.3, 1.2 + 0\} + 1$$

$$= 2$$



Algorithm : Optimal_BST(P[1...n])

//Input : An array P[1...n] of search probability for sorted ~~using~~ a list of n keys.

//Output : An ~~array~~ average no of comparision in succesful search in the optimal BST and Table 'R' of subtree roots in optimal BST.

```
for i ← 1 to n do
    c[i, i-1] = 0   // diagonal
    c[i, i] ← Pi
    R[i, i] ← i
end for
c[n+1, n] ← 0
```

```
for d ← 1 to n-1 do.
    for i ← 1 to n-d do
        j ← i + d                    // starting with 2.
        minval ← ∞
        for k ← i to j do.
            if ((c[i, k-1] + c[k+1, j] < minval)
                                                Then
                minval ← c[i, k-1] + c[k+1, j]
                kmin ← k .
            end if
        end for
        R[i, j] ← kmin
        sum ← P[i]
        for s ← i+1 to j do.
            sum ← sum + P[i]
        end for.
        c[i, j] ← minval + sum.
    end
end for
end for
return (R, c[1, n])
```