Neel Katrodiya 1002254987 DAA

ch-15

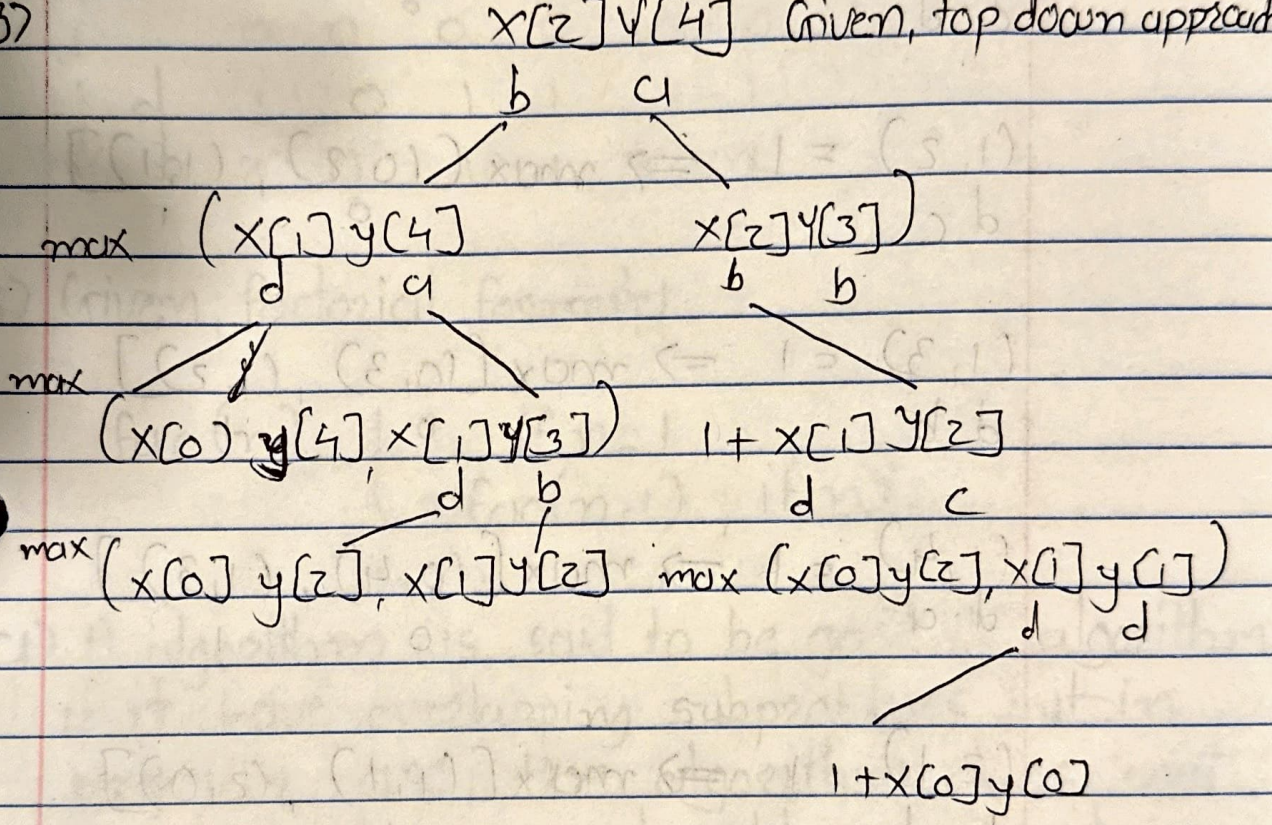
① Divide and Conquer is a strategy for separating a problem into smaller sectors and addressing each one separately, whereas dynamic programming is a strategy for solving large issues by breaking them into smaller pieces or can say has overlapping subproblems.

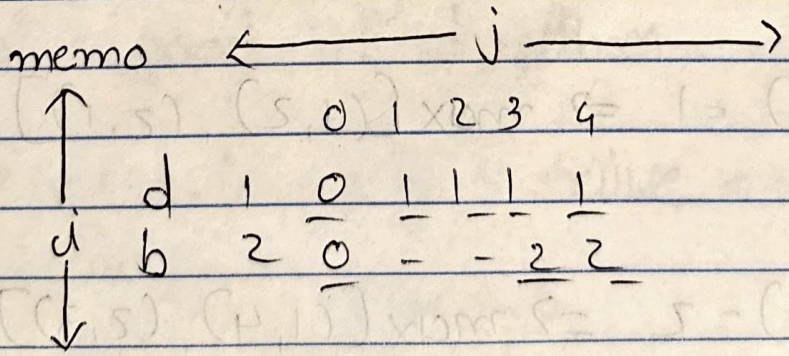
② Given, it is a naive recursive Algo, we need to fill the code which is needed using top-down approach.

a) sheet = { } [∵ sheet-name]
DP(0 0 0
if n,w in sheet return sheet
[n,w]
if n==0 or w==0
sheet[n,w] = 0
return 0
if wt[n-1] ?0
sheet[n,w] = DP(w,wt,val,n-1)
return sheet[n,w]
else
sheet[n,w] = max(0,0,0)
return sheet[n,w]

b) Other approaches that can be used are bottom-down approach or iterative approach.

3)





4) Using the bottom up code for LCS,
let the notation be (i, j) = value

(1,1) = 1 $\Longrightarrow$ 1 + (0,0)
d d

(1,2) = 1 $\Longrightarrow$ max [(0,2), (1,1)]
d c

(1,3) = 1 $\Longrightarrow$ max [(0,3), (1,2)]
d b

(1,4) = 1 $\Longrightarrow$ max [(0,4), (1,3)]
d a

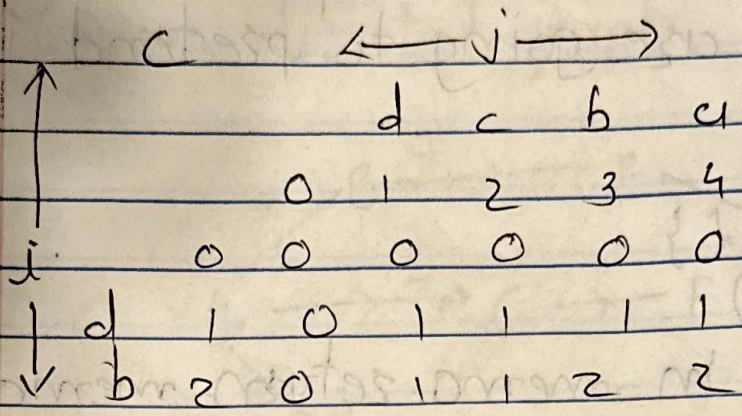
(2,1) = 1 $\Longrightarrow$ max [(1,1), (2,0)]
b d

(2,2) = 1 $\Longrightarrow$ max [(1,2), (2,1)]
b c

(2,4) = 2 $\Longrightarrow$ max [(1,4), (2,3)]

G

Given



5) Given, factorial formula

$$fac(n) = \begin{cases} 1, & \text{if } n=1 \\ n\,fac(n-1), & \text{if } n>1 \end{cases}$$

a) A algorithm is said to be good DP algorithm if it have overlapping subproblems but in ~~pair~~ fair case, the algorithm does not have overlapping subproblems, so it is not a good DP algorithm.

b) we need to write a naive recursive algorithm,

```
fac(n)
    if n==1
        return 1
    else
        return n * fac(n-1)
```

c) We need to write a top down algo and we are going to pretend its a DP algo.

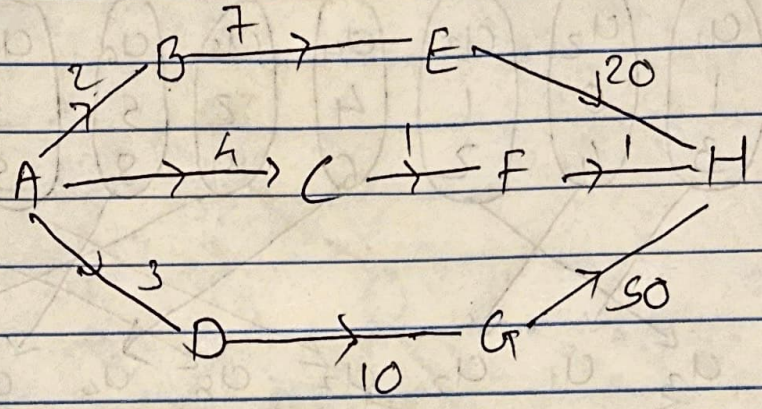
```
memo = { }
fac (n)
  if n in memo return memo[n]
   if n == 1
       memo[n] = 1
       return 1
   else
     memo[n] = n * fac (n-1)
     return memo[n]
```

d) We need to write a bottom-up algo and we are going to pretend it is a DP algorithm.

```
A = { }
A[1] = 1
fac(n)
  for i = 1 to n
    A[i+1] = [i+1] * A[i-1]
    return A[n]
```

chapter - 16

1) Given, directed graph



we need to find the shortest path between A → H
start at A

A has 3 nodes A-B, A-C, A-D
A → B is the smallest with distance 2

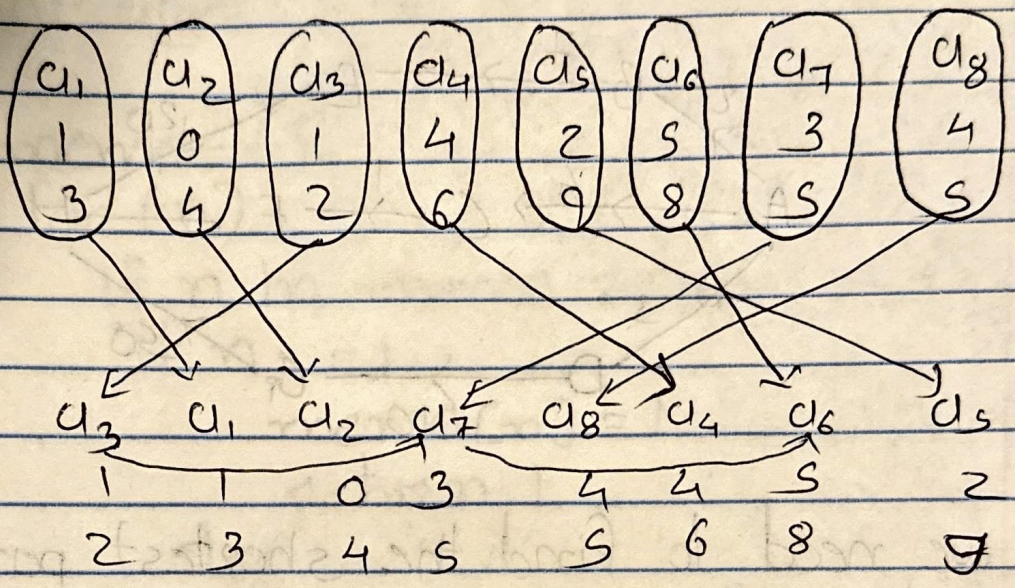
B has 2, nodes B-E, B-F
B-F is the smallest with distance 4

F has, 1 node F-H
F → H, with distance 1

The required shortest path is
A → B → F → H

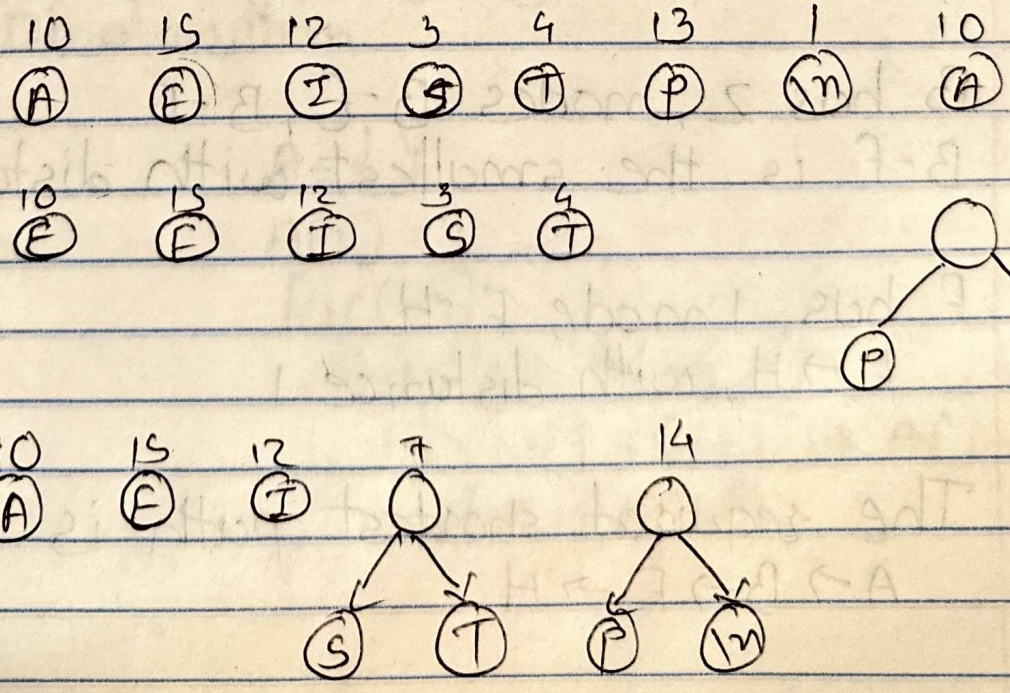
2) We need to solve the following activity problem.

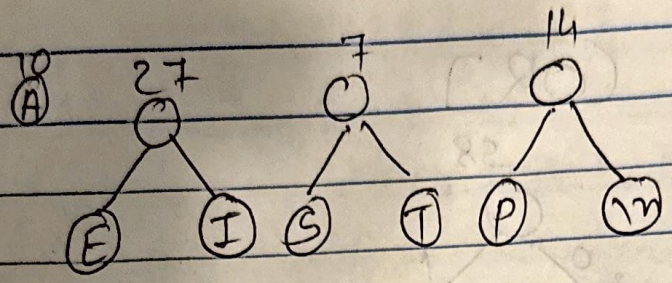
activity	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
start	1	0	1	4	2	5	3	4
finish	3	4	2	6	9	8	5	5

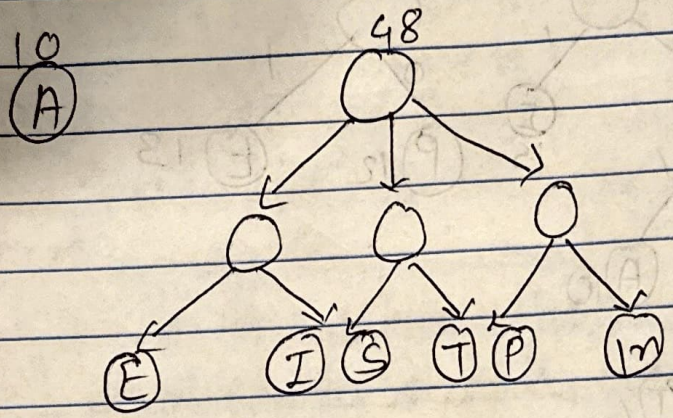


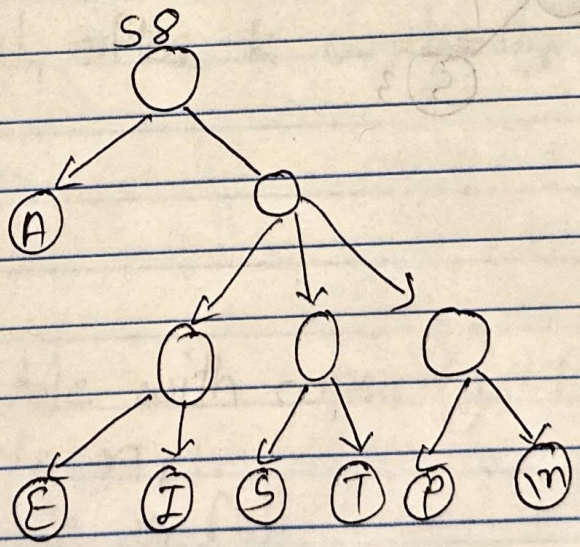
$\therefore a_3 \rightarrow a_7 \rightarrow a_6$

3) Given,




10
A
27
7
14
E
I
S
T
P
m


10
A
48
E
I
S
T
P
m


58
A
E
I
S
T
P
m

[OR]