LCS 1. What is LCS ?

2. LCS using recursion;

3. LCS " dynam.

Should not intersect each other

string1 : abcdefghij

string2 : cdgi

sequence1: cdgi longest common subseq.

dgi

gi

abcdefghij

ecdgi

egi

abcdefghij

ecdgi → cdgi

Str1 abdace

babce → bace

abdace

babce → abce

abdace

babce → bce

Starting index of A and B

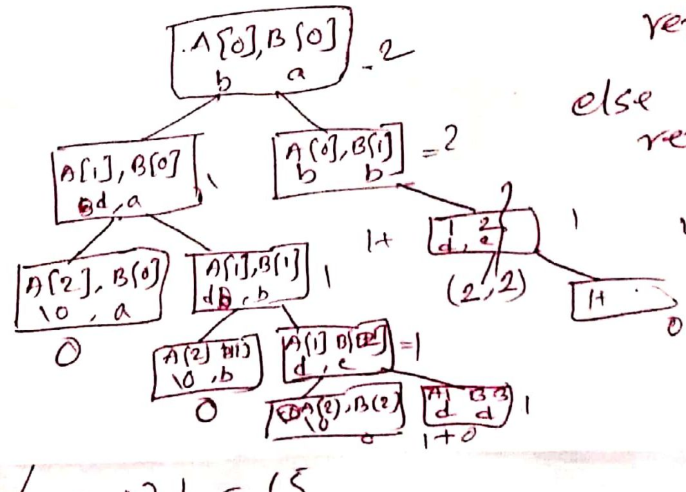
```
int LCS(i,j)
{ if (A[i]=='\0' || B[j]=='\0')
     return 0;
  else if (A[i]==B[j])
     return 1+LCS(i+1,j+1);
  else
     return max(LCS(i+1,j),
                LCS(i,j+1));
}
```

Recursion

A | b | d | \0 |

B | a | b | e | d | \0 |

0 1 2 3 4



Page. 2

Weight = 15

a(j) < a(i)

0	4	12	2	10	6	9	13	3	11	7	15
0	1	2	3	4	5	6	7	8	9	10	11

2+1 = 3 max(2,3)

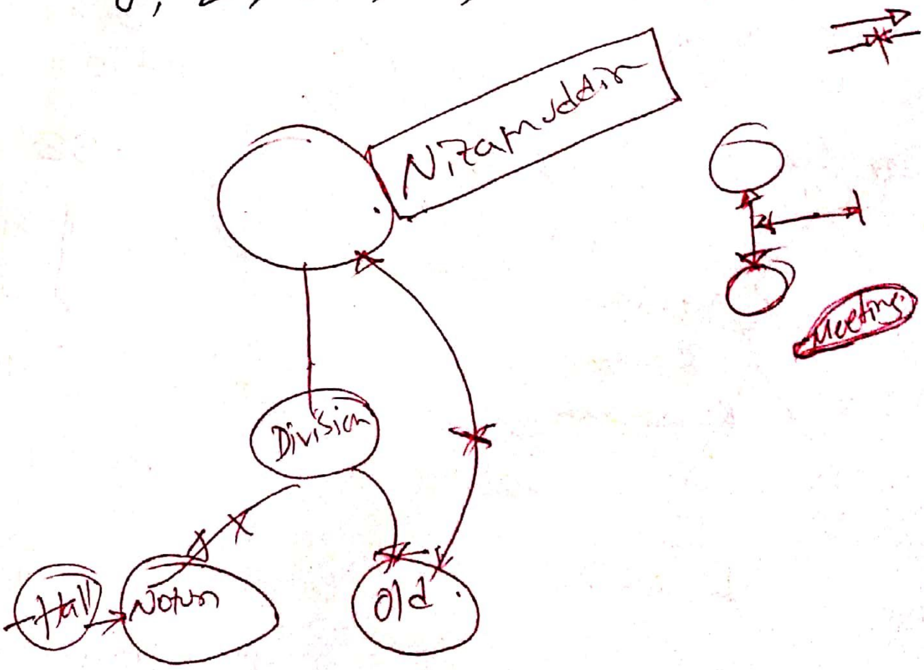
	0	1	2	3	4	5	6	7	8	9	10	11
length	1	1	1	1	1	1	1	1	1	1	1	1
len	1	2	~~2~~ 3	2	3	3	4	4 5	3	5	4	6
Subsequence	[illegible]	0	1	0	3	3	[illegible]	6	[illegible]	6	8	9

length 6 $\boxed{0, 2, 6, 9, 11, 15}$

Ans

0 3 5 6 9 11 →

0, 2, 6, 11, 15



Scanned by CamScanner

(M)

0/1 KP FIFO-BB

Upper bound. $U := -\cancel{32}\ -38$

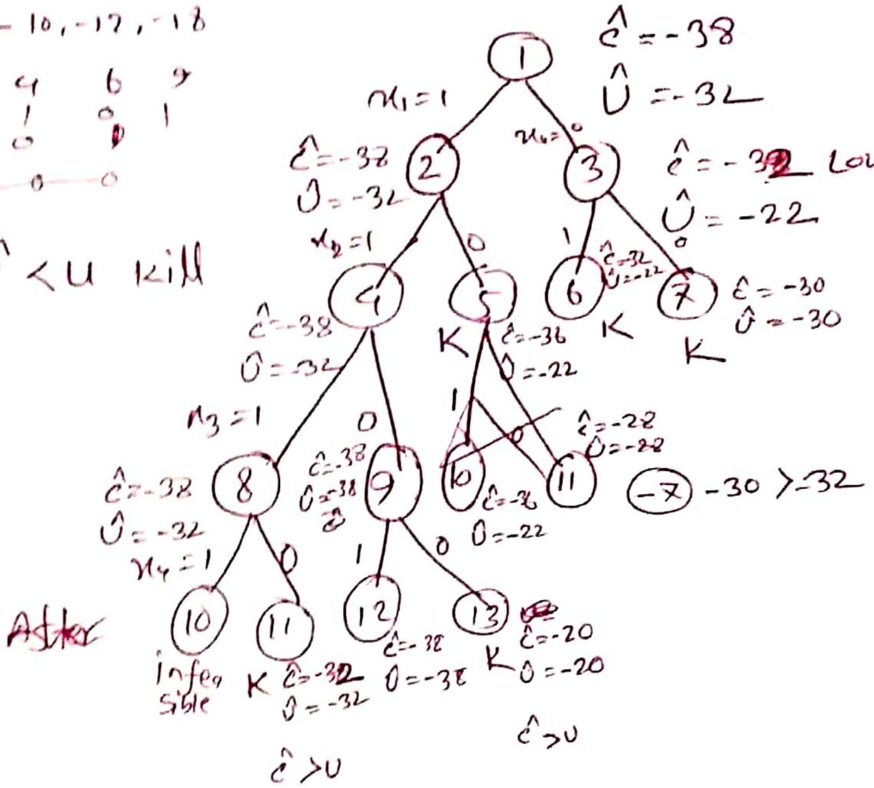
-10, -10, -12, -18

2 4 6 9

$\hat{c} < U$ kill

Upper - fraction ×

Lower - fraction ✓



if $\hat{U} < U$ replace

$U = \hat{U}$

if lower bound > global upper b

Kill the node

Knapsack is $x_1 = 1$ $x_2 = 1$ $x_3 = 0$ $x_4 = 1$

FIFO-BB

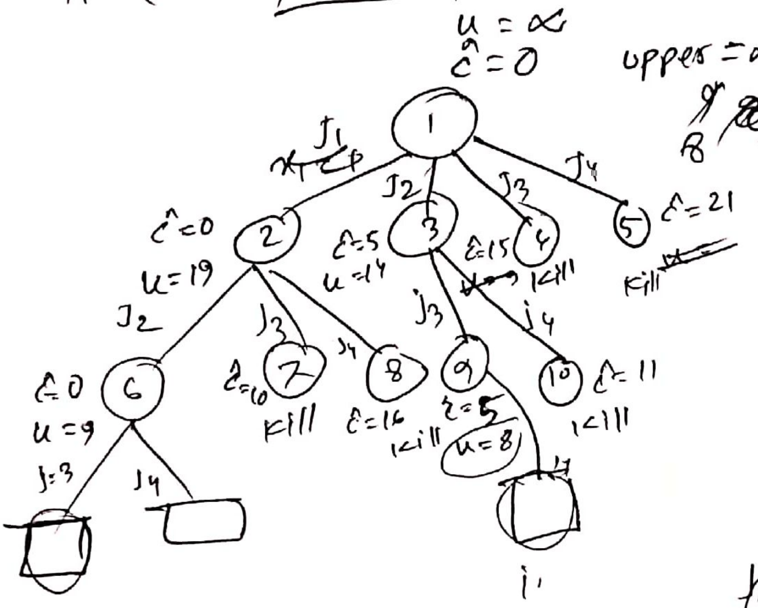
	1	2	3	4
Penalty	5	10	6	3
deadline	1	3	2	1
time	1	2	1	1

u = Sum of all ~~to~~ penalties excet that included in som

$\hat{c}$ = sum of all penalties till the last job conside.

$U = \sum_{i \notin S} P_i$

$C = \sum_{i \in S_K} P_i$



J_1	J_2	J_3	J_4
1	3	2	
1	2	1	1

3

↳ cannot be ~~ti~~ done.

$\hat{c} = 5$
$u = 8$

J_2 J_3
10 6 = 16

J_1 J_4 =
5 3 = 8

	J_1	J_2	J_3	J_4	J_5
prof	20	15	10	5	1
ded	2	2	1	3	3

Scanned by CamScanner

	a (0)	b (1)	c (2)	d (3)	10 (4)
b 0	2	2			
d 1	1	1	1	1	
10 2	0	0	0		0

2^n $(m \times n)$

A [b | d | 10]   m

B [a, b, c, d, 10]   = n

$O(m \times n)$

By Dynamic programming

		a	b	c	d
0	0	0	0	0	0
b 1	0	0	1	1	1
d 2	0	0	1	1	2

diagonal plus +1

a b

A [b | d]
1 | 2

B [a | b | c | d]
0 1 2 3

if (A[i] = B[j])
LCS(i, j) = 1 + LCS(i-1, j-1);
else
LCS(i, j) = max(LCS(i-1, j),
LCS(i, j-1));

b d

Str1 : s t o n e

Str2 : l o n g e s t

	l (0)	o (1)	n (2)	g (3)	e (4)	s (5)	t (6)	
	0	0	0	0	0	0	0	0
s	0	0	0	0	0	0	1	1
t	0	0	0	0	0	0	1	2
o	0	0	1	1	1	1	1	2
n	0	0	1	2	2	2	2	2
e	0	0	1	2	2	3	3	3

o n e

Ans := one

Sol :- / weight = 15

Scanned by CamScanner

2^{n} Solution

2^{n} Solution

$O(2^{n})$

$m = 8$ $\quad P = \{1, 2, 5, 6\}$ ← Profit

$n = 4$ $\quad \omega = \{2, 3, 4, 5\}$ ← weights

Table V

P_i	ω_i	C	0	1	2	3	4	5	6	7	8
		0	0	0	0	0	0	0	0	0	0
1	2	1	0	0	1	1	1	1	1	1	1
2	3	2	0	0	1	2	2	3	3	3	3
5	4	3	0	0	1	2	5	5	6	7	7
6	5	4	0	0	1	2	5	6	6	7	8

Previous row

$V[i, \omega] = \max\{V[i-1, \omega], V(i-1, \omega - \omega[i]) + P[i]\}$

Row, col

$V(4,1) = \max\{V(3,1), V(3, 1-5)+6\}$
$(3, -4)$ undefined

$= \max(0$

$V(4,5) = \max(V(3,5), V(3, 5-5)+6\}$
$0+6$

$= \max(5, 0+6)$

$V(4,6) = \max(V(3,6), V(3, 6-5)+6\}$
$V(3,1)+6\}$

$= \max(6, 0+6)$

$V(4,7) = \max(V(3,7), V(3, 7-5)+6)$

$= \max(7, 1+6)$

$V(4,8) = \max(V(3,8), V(3, 8-5)+6\}$

$= \max(7, 2+6) = (7, 8\}$

Soln

x_1	x_2	x_3	x_4
0	1	0	1

$8 - 6 = 2$

$2 - 2 = 0$

Sol - / weight = 15

Scanned by CamScanner

0-1 Knapsack BB

Profit 10 10 12 18

weight 2 4 6 9

$m = 15$, $n = 4$

: minimization problem: by putting negative sign.

LC-BB

$U = \sum_{i=1}^{n} p_i x_i$ $\le m$ without fraction

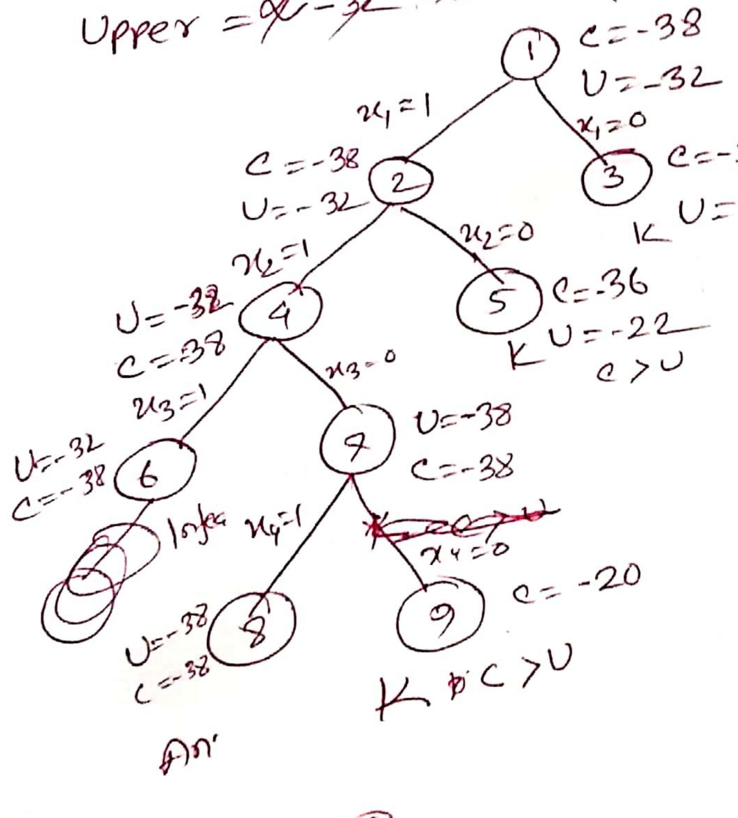
$C = \sum_{i=1}^{n} p_i x_i$ with fraction.

Subset of these objectives $S = \{x_1, x_4\}$ - Variable

$S = \{1,0,0,1\}$ - fixed size Solution.

② if c > upper Kill the node.

Upper = ~~∞~~ ~~-32~~ -38



$C = 10 + 10 + 12 + \frac{18}{9} \times 3$

2 + 4 + 6

15 - 12 = 3

= -38

$C = 10 + 10 + 12 + \frac{18}{9} \times 5$

2 + 4 + 6

15 - 10 = 5

$C = 10 \quad 10 + 12 + \frac{18}{9} \times 7$

2 4 6

15 - 8 = 7

$C = 10 + 10 + 12 + 18$

2 4 6 + 9

15 - 6 = 9

Ans

Sol^n: $x\{1,1,0,1\}$

10 + 10 + 18 = 38

weight = 15

Dynamic programming 0-1 Knapsack problem

Solving optimization problems

Principle of optimality: Every stage we take decisions.

$$fib(n) = \begin{cases} 0 & \text{if } n=0 \\ 1 & n=1 \\ fib(n-1)+f(n-2) & \text{if } n>1 \end{cases}$$

5

0, 1, 1, 2, 3, 5, 8, 13
1 2 3 4 5 6 7 8 9 10

```
int fib( int n)
{  if (n <= 1)
       return n;
   return fib( n-2) + fib(n-1) ;
}
```

Recurrence relation

$T(n) = 2T(n-1)+1$ adding

Upper bound $O(2^n)$ Decreasing function

Master theorem.

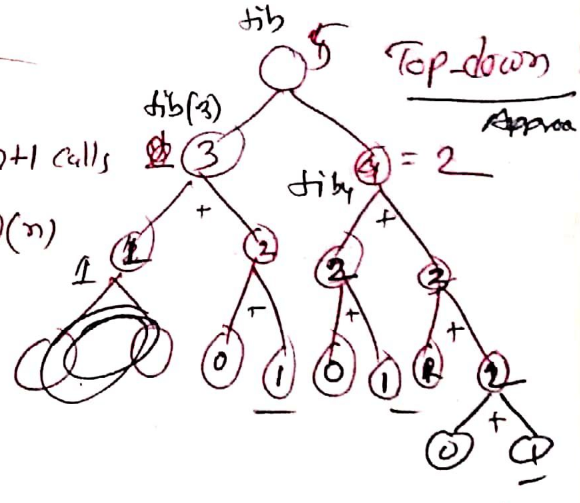
if there any way to reduce the time

Array

F

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

$fib(n) = n+1$ calls $= O(n)$



Top-down Approach

memorization - followes top down approach

Tabulation method (iterative function)

Bottom up approach

```
int fib(int n)
{
  if (n<=1)
     return n;
  F(0) =0; F(1)= 1;
  for (int i=2; i<=n; i++)
  {  F[i] = F(i-2)+F(i-1);
  }
  return F(n) }
```

$fib(n)$ F

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

Scanned by CamScanner

Travelling Salesperson Problem

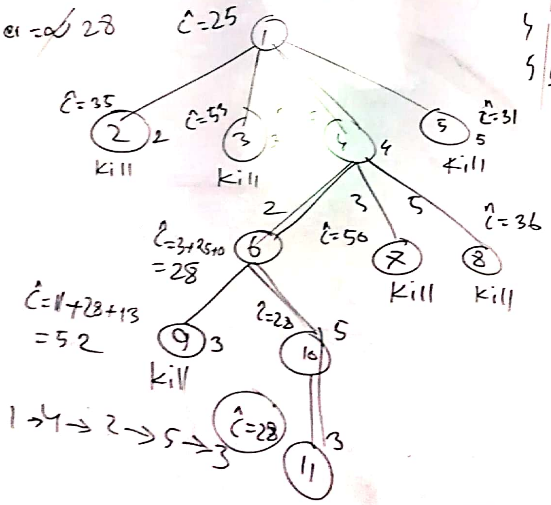
6

∞	∞	∞	∞	∞	
12	∞	11	∞	0	
0	∞	∞	∞	2	
∞	∞	∞	∞	∞	
11	∞	0	∞	∞	=0

7

∞	∞	∞	∞	∞
∞	∞	∞	∞	∞
0	∞	∞	∞	∞
∞	∞	∞	∞	∞
∞	∞	∞	∞	∞

Upper = ~~∞~~ 28



	1	2	3	4	5	
1	∞	10	17	0	1	10
2	12	∞	11	2	0	2
3	0	3	∞	0	2	2
4	15	3	12	∞	0	3
5	11	0	0	12	∞	4
	1	0	3	0	0	+21 = 25

4

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	12	∞	11	∞	0
3	0	3	∞	∞	2
4	∞	3	12	∞	0
5	11	0	0	∞	∞

2

	1	2	3	4	5	
1	∞	∞		∞	∞	0
2	∞	∞		2	0	0
3	0	∞		0	2	0
4	15	∞	2	∞	0	0
5	11	∞	0	12	∞	0

3

1	∞	∞	∞	∞	∞	
2	1	∞	∞	2	0	0
3	∞	3	∞	0	2	0
4	4	3	∞	∞	0	0
5	0	0	∞	12	∞	0
	11	0	0	0	0	=11

Scanned by CamScanner