DAY:_____ DATE:_____

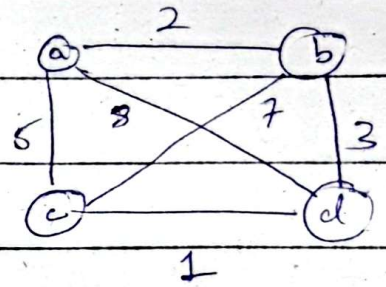
DAA -10 (20/03/25)

Optimal Solution:

Exhaustive Search.

$x+y = 100$ ⇒ 100 combination of x & y values.

$\frac{1}{2}(n-1)!$



⇒ Prioritize any parameter
↳ Time vs cost

Knapsack Problem:

Limitation : Capacity of bag.

Goal : Max cost (Maximization Problem)

Item	w	v
1	7	42
2	3	12
3	4	40
4	5	25

1) 0-1 Knapsack ⇒ Total pick or not ⇒ 2^n (No of subsets)
2) fractional knapsack

position not matter.

DAY: _____ DATE: _____

Assignment Problem: (Opt - prob)

↳ cost minimization.

↳ Person (with 4 skills) ⇒ Hiring for 4 positions.

1	P1 → J2	
2	P2 → J3	J1
1	P3 → J3	J3 (min)
100	P4 → J4	

↳ In last, persons may be assigned max values.

↳ 1) Exhaustive Way:

P1, shuffle Remaining cost:

P2, shuffle Remaining

2) Hungarian Method: (Non-exhaustive).

$$\begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix} \xrightarrow[\text{Reduction}]{\text{Row}} \begin{bmatrix} 7 & 0 & 5 & 6 \\ 3 & 1 & 0 & 4 \\ 4 & 7 & 0 & 4 \\ 3 & 2 & 5 & 0 \end{bmatrix}$$

(Row Reduction) 1) Min of Row & Sub from that Row.

(Col Reduction) 2) Min of Col & Sub from that Col.

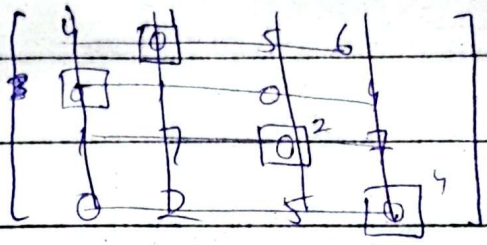
DAY:_____ DATE:_____

Col Red →

$$\begin{bmatrix} 4 & 0 & 5 & 6 \\ 0 & 1 & 0 & 4 \\ 1 & 7 & 0 & 7 \\ 6 & 2 & 5 & 0 \end{bmatrix}$$

3) Draw ^{min} lines to cover all zeros.

↳ lines = Order (Optimal).



Example 1:

	X	Y	Z
A	250	400	350
B	400	600	350
C	200	400	250

R.R

$$\begin{bmatrix} 0 & 150 & 100 \\ 50 & 250 & 0 \\ 0 & 200 & 50 \end{bmatrix}$$

C.R

$$\begin{bmatrix} 0 & 0 & 100 \\ 50 & 100 & 0 \\ 0 & 50 & 50 \end{bmatrix}$$

(B|P2) → J3 350

C|P3) → J1 200

A(P1) → J2 400

950 cost

DAY: DATE:

Example 2:

$$\begin{bmatrix} 90 & 75 & 75 & 80 \\ 35 & 85 & 55 & 65 \\ 125 & 95 & 90 & 105 \\ 45 & 110 & 95 & 115 \end{bmatrix}$$

R.R

$$\begin{bmatrix} 15 & 0 & 0 & 5 \\ 0 & 50 & 20 & 30 \\ 35 & 5 & 0 & 15 \\ 0 & 65 & 50 & 70 \end{bmatrix}$$

C.R

$$\begin{bmatrix} 15 & 0 & 0 & 0 \\ 0 & 50 & 20 & 25 \\ 35 & (5) & 0 & 10 \\ 0 & 65 & 50 & 65 \end{bmatrix}$$

Not Optimal.

① Sub Min → subtract from uncovered row.

↳ add to covered (col)

② Intersection values add.

Uncovered values sub

$$\begin{bmatrix} 15 & 0 & 0 & 0 \\ -5 & 45 & 15 & 20 \\ 30 & 0 & -5 & 5 \\ -5 & 60 & 45 & 60 \end{bmatrix}$$

Again

$$\begin{bmatrix} 20 & 0 & 5 & 0 \\ 0 & 45 & 20 & 20 \\ 35 & 0 & 0 & 5 \\ 0 & 60 & 50 & 60 \end{bmatrix}$$

DAY:_____ DATE:_____

DAA -11 (16/04/25)

Hamalgonian Circuit:

↳ Not visit any point twice

optimization => cost reduction / Maximization.

Optimization Problem

NP problem.

↳ non polynomial runtime.

x^n => exponent.

Assignment, Knapsack, => exp run time.

Hungarian method => convert to polynomial runtime.

Backtracking:

↳ DFS ↳ for non-optimization problems.

N Queen Problem:

n => no of queens nxn = board size.

n = 2 => no sol exist.

n = 3

↳ state space Tree

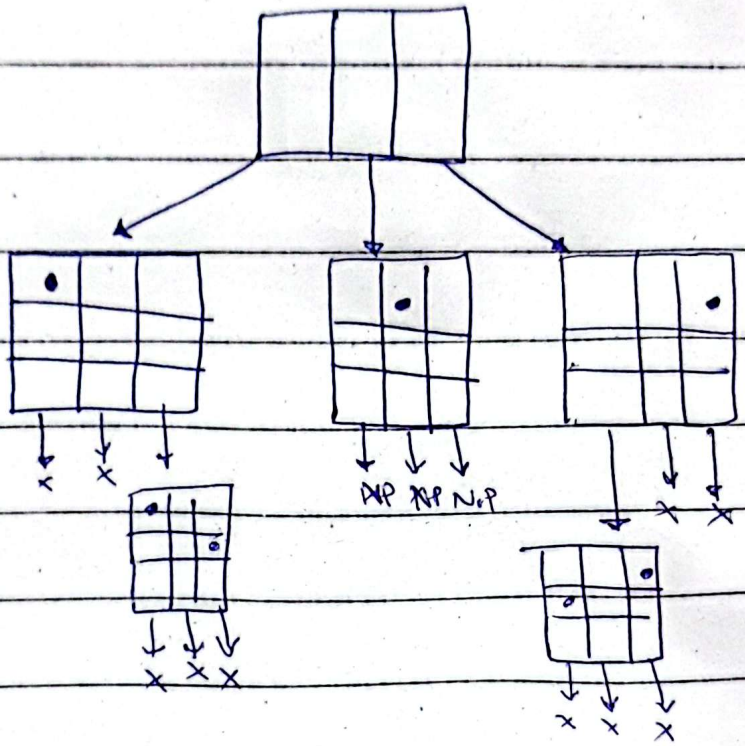
↳ Promising state

↳ Non-promising state.

DAY: _____ DATE: _____

n=3:

1) first empty: 2) Add row by row.



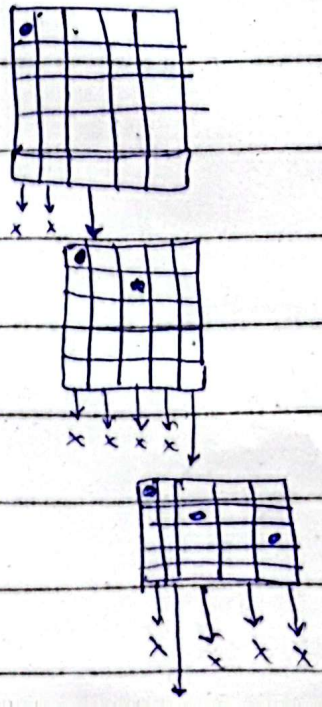
↳ 2nd Half state space tree is mirror of 1st Half

↳ for true/false, we have to solve till half state space tree. (At max)

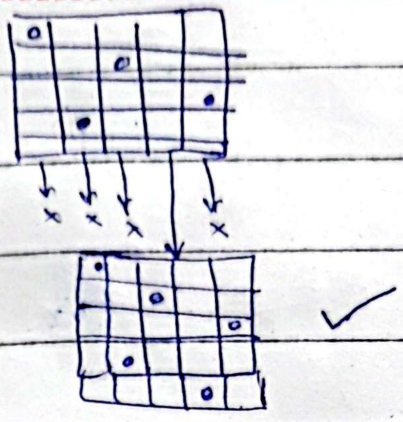
↳ for all combinations ⇒ complete solve

n=5:

1st cell:

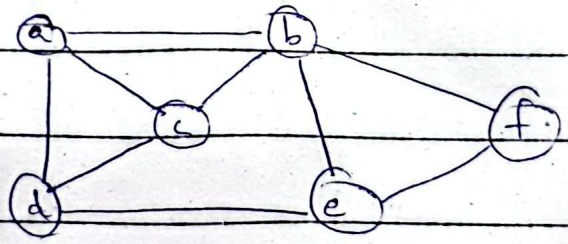


DAY:_____ DATE:_____



DAA -12 (18/04/25)

Backtracking:



Optimization vs Non Optimization?

Objective func?

Q: Can assignment problem be solved by backtracking?

↳ backtracking ⇒ for non-optimization prob.

↳ backtracking is exhaustive technique

Hamiltonian Circuit:

(non-optimal) ⇒ (no min cost etc)

↳ don't visit again.

↳ travelling sales person problem.

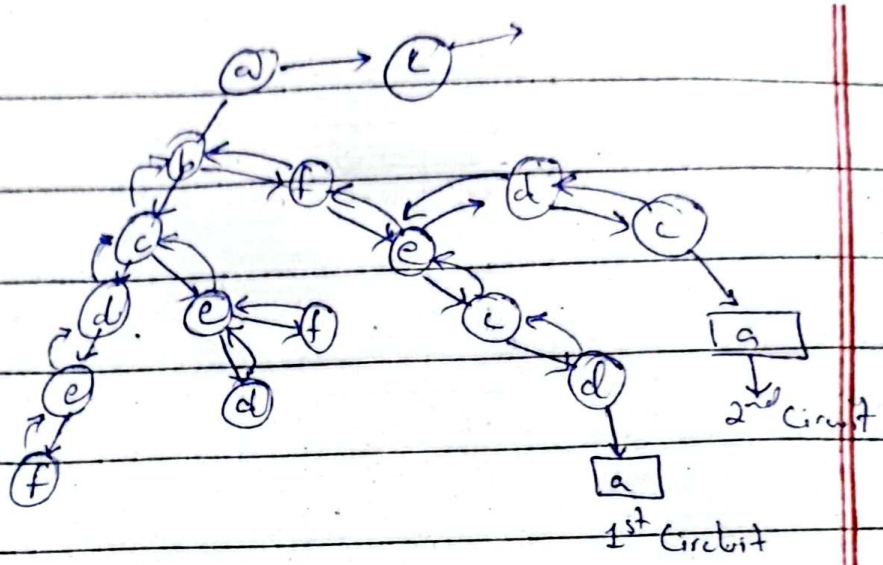
↳ starting vertex ↳ direct/indirect edge.

↳ use alphabetical order

↳ (for circuit) ⇒ if you have visited all vertex, you must have direct edge to start

DAY: _____ DATE: _____

a : b, c, d
b : a, c, f
c : a, b, d, e
d : a, c, e
e : c, d, f
f : e, b.



Subset Sum Problem:

Set = {3, 5, 6, 7}

d = 15

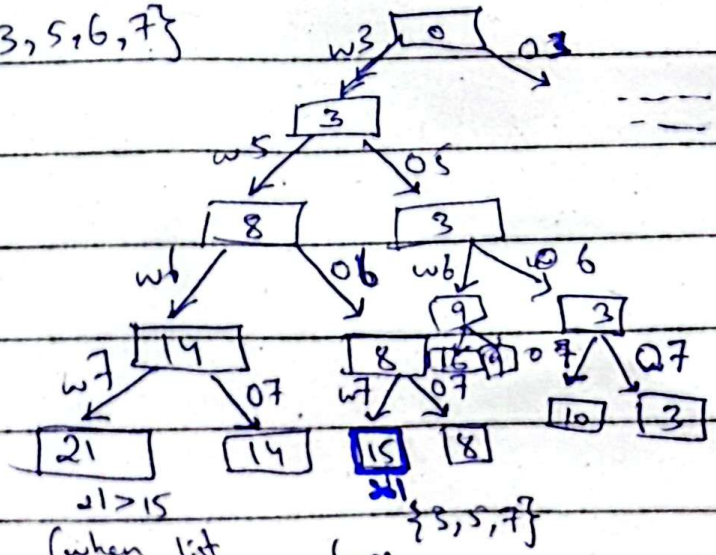
↳ All Possible Subset (emptyset also) Knapsack

↳ check: sum of all value is less than d return {}

↳ if sum == d ?

↳ if sum > d. ?

{3, 5, 6, 7}

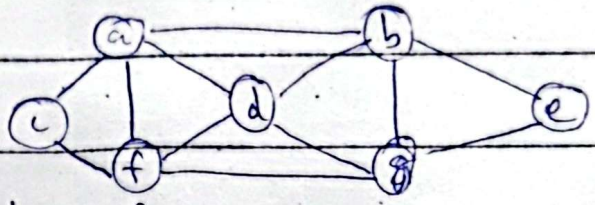


(when list is sorted → no solⁿ → next stem)

here

DAY:_____ DATE:_____

Hamultonian Circuit:



a : b, c, d, f

b : a, d, e, g

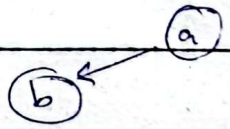
c : a, f

d : a, b, f, g

e : b, g

f : a, c, d, g

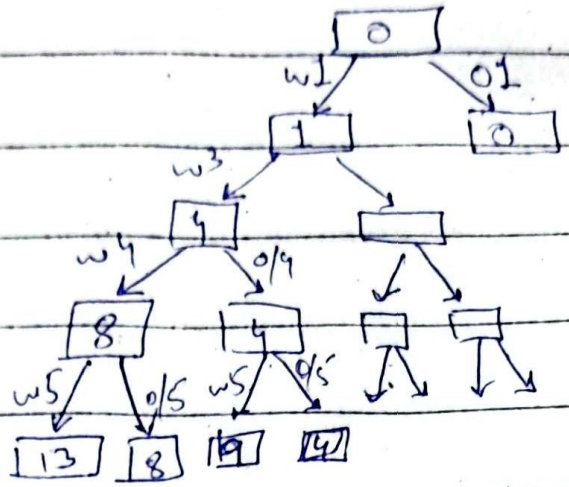
g : b, d, e, f



DAY:_____ DATE:_____

Q No 8:

$d = 11$ $\{1, 3, 4, 5\}$



⇒ don't leave when sum < d.
sum values could be removed to produce d.

Q No 7:

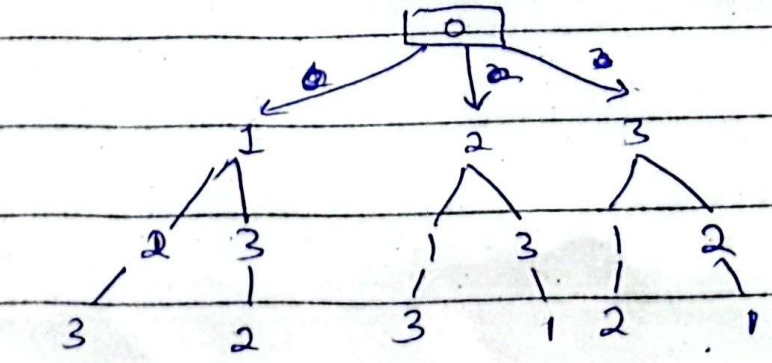
Backtracking to produce permutations of $\{1, 2, 3, 4\}$

permutation ⇒ order matter.

$\underline{1}, 2, 3 \Rightarrow$
fix

$1, \underline{2}, 3 \Rightarrow$
fix

$1, 2, \underline{3} \Rightarrow$
fix



(1,2,3) (1,3,2) .....

DAY:_____ DATE:_____

DAA-13 (30/04/25)

Branch & Bound

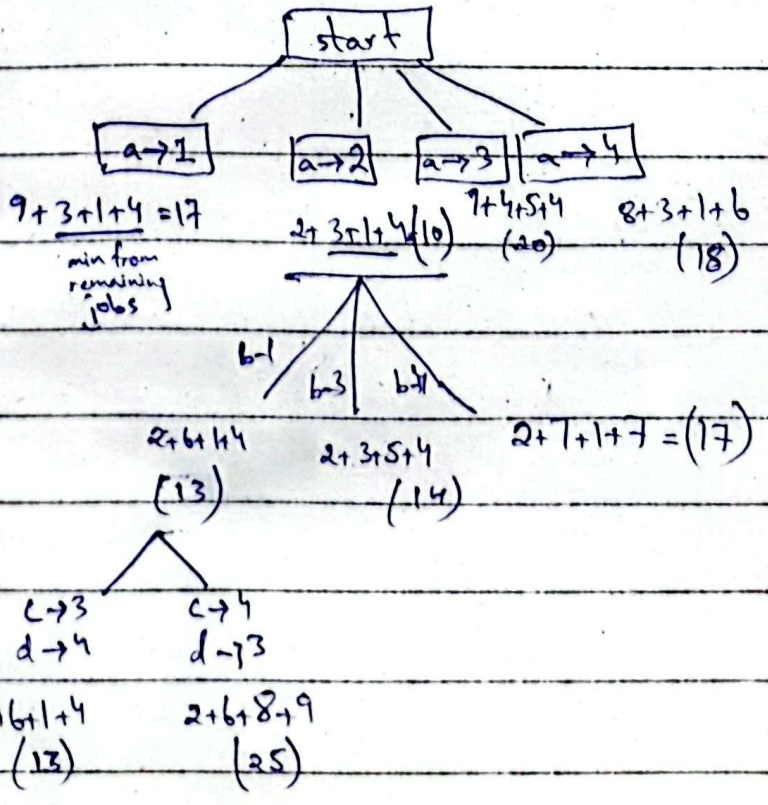
↳ similar to backtracking
↳ initial state
↳ state space tree ↳ best fit check(close to lower/upper bound)
↳ live node.
↳ node cannot be cancelled until you reach optimal solution.
↳ bounding function ⇒ upper bound / lower bound
↓ maximization ↓ minimization

1) Assignment Problem.

objective ⇒ cost minimization ⇒ lower bound.

$$\begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix}$$

2+3+1+4 = 10 ⇒ LB.



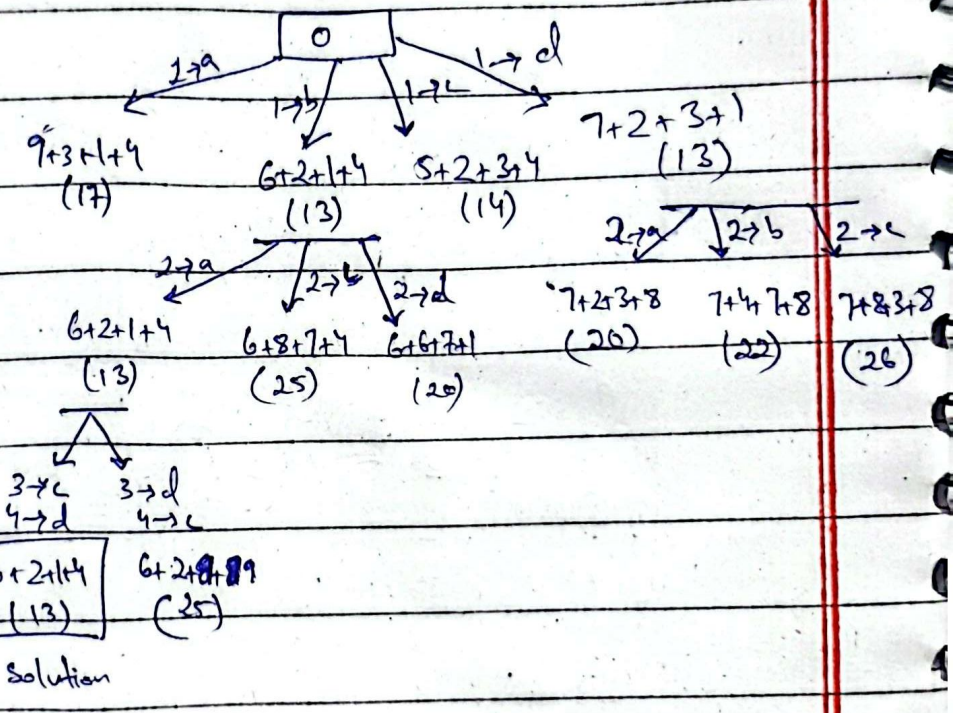
DAY:_____ DATE:_____

↳ live nodes are crossed out at the end.

Example:

⇒ on matrix col rather than rows

$$\begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix}$$



2) Knapsack Problem:

↳ Optimization → maximization.

Item	Weight	Value	Value/Weight
1	4	40	10
2	7	42	6
3	5	25	5
4	3	12	4

DAY:_____ DATE:_____

0-1 Knapsack → value.

fractional Knapsack → value/weight (greedy).

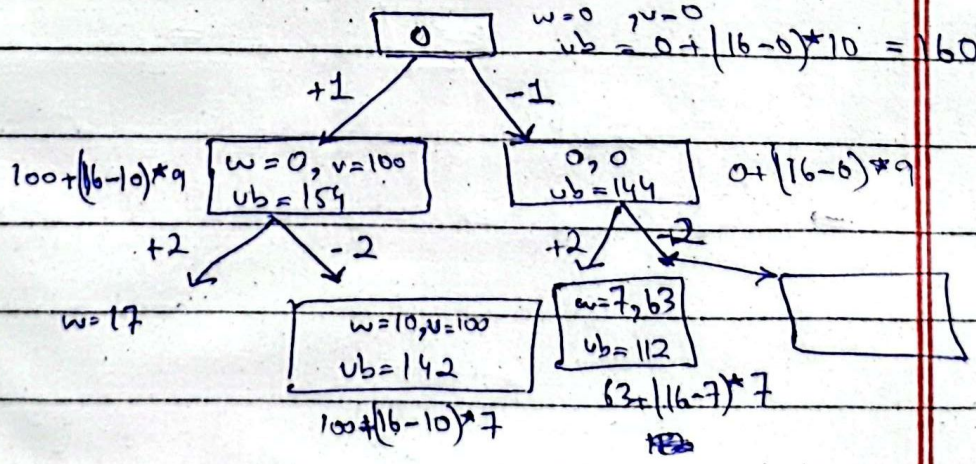
Upper Bound:

$ub = v + (W - w)(v_{i+1} / w_{i+1})$

∵ W → Sack's weight capacity.

= value of items + (Sack Capacity - weight of items) * ratio of next

Item	W	V	
1	10	100	10
2	9	63	9
3	8	56	7
4	4	12	3



DAY: DATE:

DAA -14 (02/05/25)

Branch & Bound

Travelling Salesperson Problem.

↳ Hamiltonian Circuit ⇒ Non Opt Prob.

↳ Opt ⇒ Obj ⇒ cost minimization.

boundary func ⇒ lower bound.

a: 1, 3, 5, 8
b: 3, 6, 7, 9
c: 1, 2, 4, 6
d: 3, 4, 7, 5
e: 2, 3, 8, 9

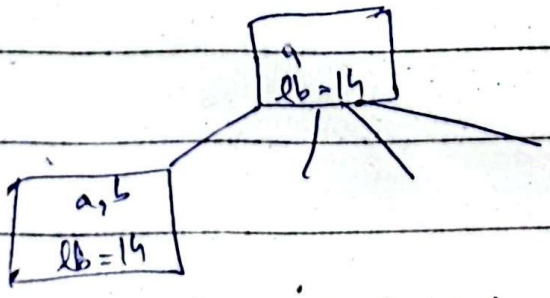
⇒ must-include edge (a,d)

(2 edges ⇒ divided by 2)

$lb = [(1+3) + (3+6) + (1+2) + (3+5) + (2+3)]/2 = 16$

⇒ for (a,d) ⇒ at a include (a,d) + smallest

at d include (a,d) + smallest



(3+1) + (3+6) + (1+2) + (3+4) + (2+3)/2.

a	b	c	d	e
(4, 9, 3 + 7 + 5)/2 = 28/2 = 14				

fixed in tree below.

DAY:_____ DATE:_____

a, b, c, d, (e, a) ⇒ (3+8) + (3+6) + (6+4) + (4+3) + (3+8)

a, b, c, e, (d, a) ⇒ (3+5) + (3+6) + (6+2) + (3+5) + (2+3)

Priority ⇒ Branch & Bound.

⇒ **Greedy Algorithm:**

↳ most of the time optimal but sometime may not lead to optimal.

(Ex) ↳ solve 0-1 using greedy.

Greedy Choice

1) Huffman 2) Kruskal 3) Prims.

Encoding vs encryption vs compression

↳ Dynamic programming solves the subproblems before making first choice.

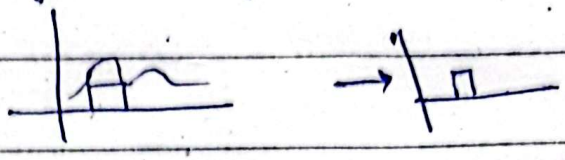
↳ Greedy algorithm makes its first choice before solving any sub problem.

↳ fixed length coding ⇒ less compression due to most frequent words.

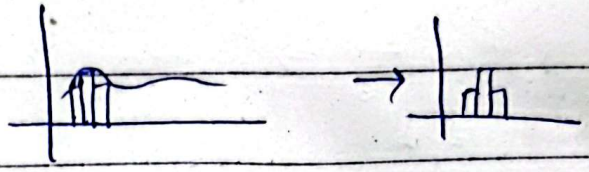
↳ good encoding schemes ⇒ compression also

DAY:_____ DATE:_____

Analog → Digital → Analog:



① Lossy compression



② lossless compression

Text: a l i a

a	2	0
l	1	1
i	1	01

↳ while decrypting ⇒ wrong decryptions also

↳ **prefix code:** (should not repeat)

01 → i/al ✗

↳ very hectic manually.

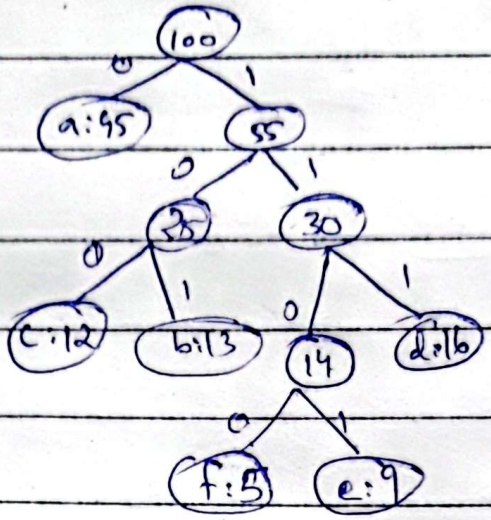
↳ huffman.

↳ **Huffman Codes:**

1) sort values in asc order.
2) Join 2 min nodes (parent has sum)
3) Repeat.

a:45 , b:13 , c:12 , d:16 , e:9 , f:5

DAY:_____ DATE:_____



fade → 1100 0 1111 1101
af a d e

↳ CS ⇒ heap (linear + nlgn)

Spanning Tree:

↳ All vertex connected to edges.

↳ $|E| = |V| - 1$

↳ Opposite to forest (disconnected tree)

⇒ Greedy.

↳ Kruskal

↳ Prism

↳ Knapsack

DAY: DATE:

DAA - 15 (14/05/25)

Dynamic Programming.

x^n → exp

n^2 → polynomial.

DP solves optimization problem.

↳ optimal results.

↳ convert to polynomial runtime.

→ Row Coins Problems

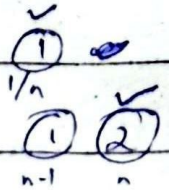
O O O O O

↳ Cannot pick 2 adjacent coins.

factorial (Recursive → top down)

factorial (Iterative ⇒ bottom up)

↳ DP ⇒ Bottom Up Approach.



min no of coin pick = 0 f(n) ⇒ n = 0

f(0) ← 0 f[1] ← C[1]

for(i = 2 to n):

$f(n) = \max\{c_n + f[n-2], f(n-1)\}$

DAY:_____ DATE:_____

n	0	1	2	3	4	5	6
C		5	1	2	10	6	2
Left → F	0	5	5	7	15	15	17

only

$f(n) = 0 \quad , n = 0$

$f(n) = c \quad , n = 1$

$f(n=2) = \{ 1 \; 0 \; , 5 \} \Rightarrow 5$

$\max \{ c_2 + f[0] , f[1] \}$

↳ which part gives max benefits ⇒ coins.

backtracking for the coin indices.

Change-Making Problem:

return change with minimum coins.

↳ minimization.

$n = 6 \quad d_i : 1, 3, 4.$

↳ 1, 1, 1, 1, 1, 1 (6)

↳ 3, 3 (2)

↳ 4, 1, 1 (3)

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

$f(0) = 0$

$f(n) = \min_{j : n \geq d_j} \{ f[n - d_j) \} + 1 \quad n > 0$

DAY: _____ DATE: _____

$$f[1] = \min_{n=1 \geq 1} \{ f[1-1] \} + 1 = \min\{f[0]\} + 1 =$$

$$= 0 + 1 = 1$$

$$f[2] = \min_{n=2 \geq 1} \{ f[2-1] \} + 1 = 1 + 1 = 2$$

$$f[3] = \min_{n=3 \geq 3} \{ f[3-1], f[3-3] \}$$

↗ dj

$$= \min_{n=3 \geq 3} \{ f[2], f[0] \} + 1 = 0 + 1$$

2 , 0

↓ min

$$= 1$$

BackTrack:

min value ⇒ dj