Dynamic Programming

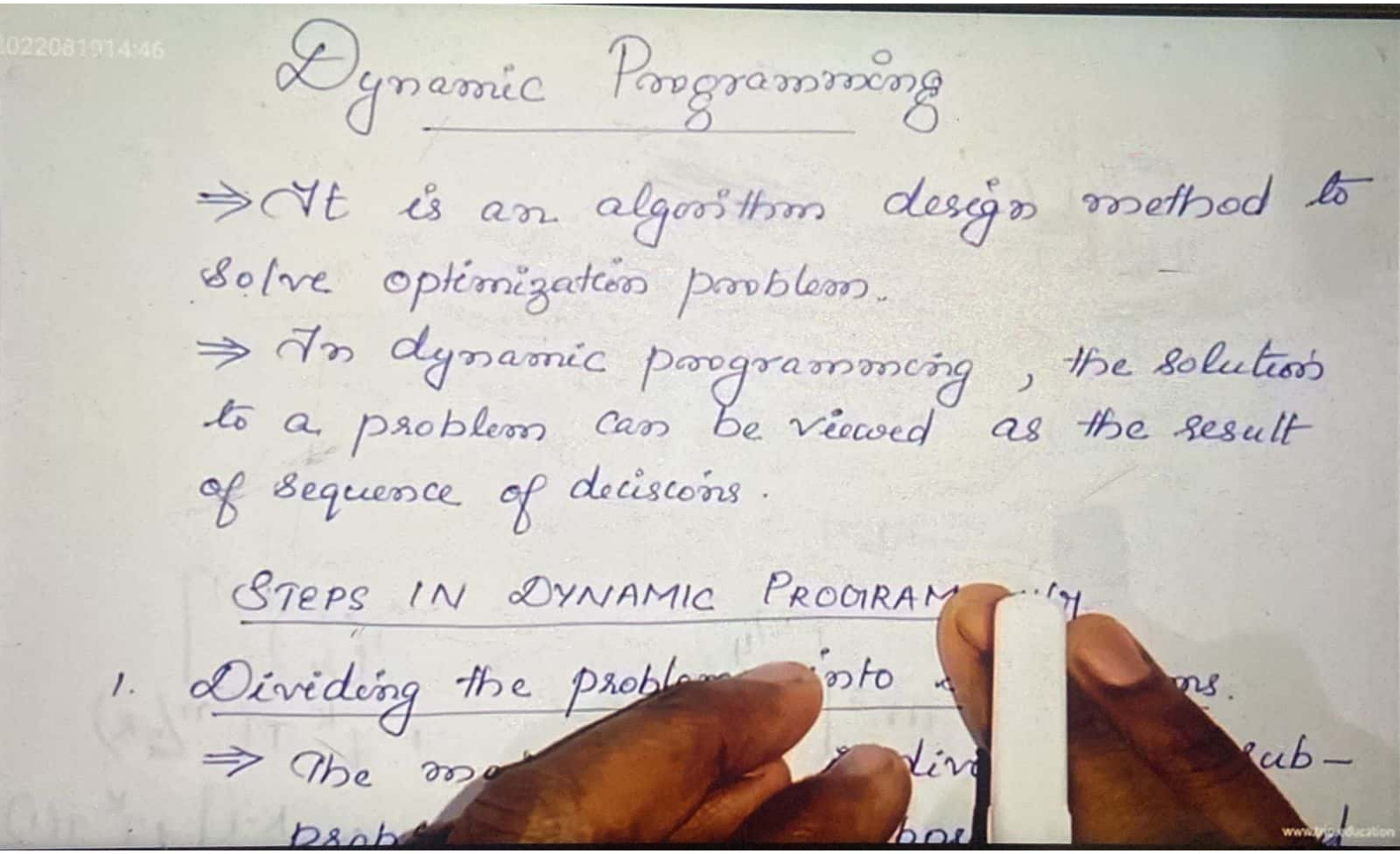
⇒ It is an algorithm design method to solve optimization problem.

⇒ In dynamic programming, the solution to a problem can be viewed as the result of sequence of decisions.

STEPS IN DYNAMIC PROGRAMMING

1. Dividing the problem into ... ns.

⇒ The ... div... sub-



STEPS IN DYNAMIC PROGRAMMING

1. Dividing the problem into subproblems.

⇒ The main problem is divided into sub-problem and each subproblem is solved only once.

⇒ The dynamic programming technique starts with the smaller problem instance. So, dynamic programming is bottom up approach

2. Storing the subsolution in a table

⇒ The solution for each subproblem is

only once.

⇒ The dynamic programming technique starts with the smaller problem instance. So, dynamic programming is bottom up approach

2. Storing the subsolution in a table

⇒ The solution for each subproblem is stored in a table so that it can be refered many time whenever require. It is also called memoization

will examine the previously solved subprob-
lems and will combine their solutions to
give the best solution for the given problem.

imp Principle Of Optimality - It states that
An optimal sequence of decisions has
the property that whatever the initial state
and decisions are; the remaining decisions
must, constitute an optimal decision sequence
with regard to the state resulting from the
first decision.

Consider the test path problem.

Principles of Optimality

- **Definition**: The principle of optimality states that an optimal sequence of decisions has the property that whatever the initial state and decision are, the remaining decisions must constitute an optimal decision sequence with regard to the state resulting from the first decision.

- A problem is said to satisfy the Principle of Optimality if the subsolutions of an optimal solution of the problem are themesleves optimal solutions for their subproblems.
- Examples: The shortest path problem satisfies the Principle of Optimality.
  - This is because if $a,x_1,x_2,\ldots,x_n,b$ is a shortest path from node a to node b in a graph, then the portion of x_i to x_j on that path is a shortest path from x_i to x_j.

Characteristics of Dynamic Programming

1. **Overlapping Subproblems**

- In dynamic programming the solution of sub-problems are needed repeatedly. The computed solutions are stored in a table, so that these don't have to be re-computed. Then combines the solutions of the sub-problems.

2. **Optimal Substructure**

- A given problem has Optimal Substructure Property, if the optimal solution of the given problem can be obtained using optimal solutions of its sub-problems.
- Example: Shortest Path problem
  - If a node **x** lies in the shortest path from a source node **u** to destination node **v**, then the shortest path from **u** to **v** is the combination of the shortest path from **u** to **x**, and the shortest path from **x** to **v**.

Steps in Dynamic Programming

1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution, typically in a bottom-up fashion.
4. Construct an optimal solution from computed information.

Optimal Matrix Multiplication

- Suppose we wish to compute the product of 4 matrices $A_1 \times A_2 \times A_3 \times A_4$
- The different parenthesizations are

$(A_1(A_2(A_3 A_4)))$

$(A_1((A_2 A_3) A_4))$

$((A_1 A_2)(A_3 A_4))$

$((A_1(A_2 A_3))A_4)$

$(((A_1 A_2) A_3)A_4)$

Optimal Matrix Multiplication

- Example: Consider 3 matrices A_1, A_2 and A_3. Its dimensions are 10X100, 100X5, 5X50 respectively. Compute the product($A_1A_2A_3$).
- Two ways
  - $((A_1A_2)A_3)$
    - Number of scalar multiplications

      $=10x100x5 + 10x5x50 = 7500$
  - $(A_1(A_2A_3))$
    - Number of scalar multiplications

      $=100x5x50 + 10x100x50 = 75000$
- Thus, computing the product according to the first parenthesization is 10 times faster.

Optimal Matrix Multiplication

- A_1 → 10X100 A_2 → 100X5 A_3 → 5X50

- $((A_1A_2)A_3)$
  - Number of scalar multiplications =**10x100x5**
  - The resultant matrix dimensions will be 10x5

  - Number of scalar multiplications =**10x5x50**

  - Total number of scalar multiplications
    = 10x100x5 + 10x5x50
    = **7500**

Optimal Matrix Multiplication

- A_1 → 10X100 A_2 → 100X5 A_3 → 5X50

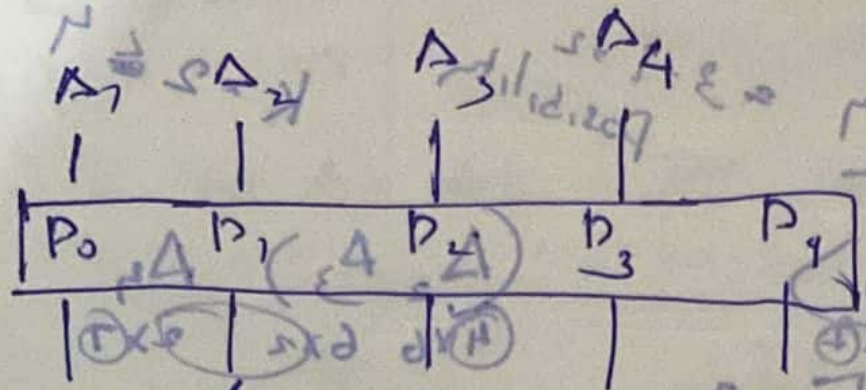
- $(A_1 (A_2 A_3))$
  - Number of scalar multiplications =**100x5x50**
  - The resultant matrix dimensions will be 100x50

  - Number of scalar multiplications =**10x100x50**

  - Total number of scalar multiplications
    = 100x5x50 + 10x100x50
    = **75000**

Given a chain of 4 matrices $<A_1,A_2,A_3,A_4>$ with dimensions<5X4>,<4X6>,<6X2>,<2X7> respectively. Using Dynamic programming find the minimum number of scalar multiplications needed and also write the optimal multiplication order

$A_1\ A_2\ A_3\ A_4$ 5×4 4×6 6×2 2×7



m table

i/j	1	2	3	4
1	0	120	88	158
2	x	0	48	104
3	x	x	0	84
4	x	x	x	0

i/j	1	2	3	3
1	0	1	88	3
2	x	0	2	3
1	x	x	0	3
4	x	x	x	0

$m[1,2] = 120$

$m[2,3] = 48$

$m[3,4] = 84$

$m[1,3]$

$A_1\ A_2\ A_3$ — 2 possibilities

$A_1\ (A_2\ A_3)$

$5\times4 \quad 4\times6 \quad 6\times2$

$m[1,1] + m[2,3] + 5\times4\times2$

$= 88$ — min

$m[1,3] = d\times3$

$A_1\ A_2 + A_3$

$A_1\ (A_2\ A_3)$

$(A_1\ A_2)\ A_3$

$m[1,1] + m[2,3] +$

$(A_1\ A_2)\ A_3$

$5\times4 \quad 4\times6 \quad 6\times2$

$m[1,2] + m[3,3] + 5\times6\times2$

$= 180$

→ m[2,4]

$A_2\ A_3\ A_4$ = 3 matrices

pos 1, 2

k = 2 ≤ 4

$A_2 (A_3 A_4)$

4×6 6×2 2×7

M[2,2] + M[3,4] + 4×6×7

= 0 + 84 + 168

= 252

$(A_2 A_3) A_4$

4×6 6×2 2×7

M[2,3] + M[4,4] + 4×2×7

= 48 + 0 + 56

= 104 min

→ m[1,4]

$A_1\ A_2\ A_3\ A_4$

$A_1 (A_2 A_3 A_4)$

5×4 4×6 6×2 2×7

M[1,1] + M[2,4] + 5×4×7

= 0 + 244

$(A_1 A_2 A_3) A_4$

5×4 4×6 6×2 2×7

M[1,3] + M[4,4] + 5×2×7

= 158 min

$(A_1 A_2) (A_3 A_4)$

5×4 4×6 6×2 2×7

M[1,2] + M[3,4] + 5×6×7

= 414

k = 1 ≤ 4

Scanned with CamScanner

A_1	A_2	A_3	A_4
5x4	4x6	6x2	2x7

m

	1	2	3	4
1	0	120	88	158
2		0	48	104
3			0	84
4				0

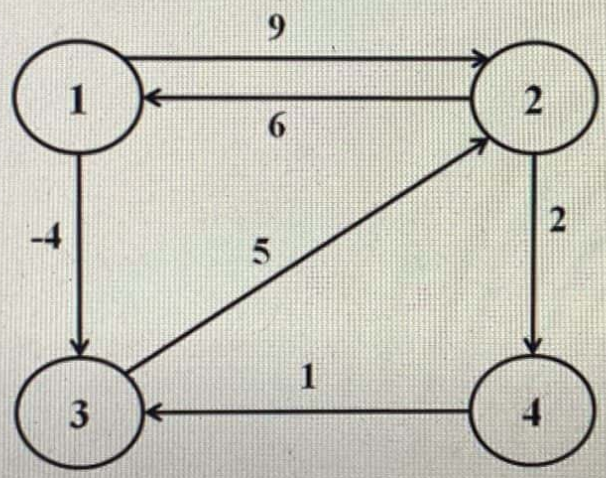
s

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

$((A_1 \; (A_2 \; A_3)) \; (A_4))$

All Pairs Shortest Path Problem

- Find the shortest distances between every pair of vertices in a given weighted directed Graph.



All Pairs Shortest Path Problem

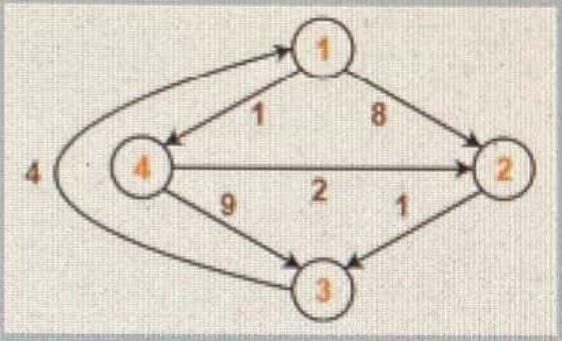
- The **Floyd Warshall Algorithm** is for solving the All Pairs Shortest Path problem.
- Negative edge weights are also allowed.
- As a result of this algorithm, it will generate a matrix, which will represent the minimum distance from any node to all other nodes in the graph

Floyd-Warshall algorithm

```
FLOYD-WARSHALL(W)
1  n = W.rows
2  D^(0) = W
3  for k = 1 to n
4      let D^(k) = (d_ij^(k)) be a new n × n matrix
5      for i = 1 to n
6          for j = 1 to n
7              d_ij^(k) = min(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
8  return D^(n)
```

- Running time is $O(n^3)$.

- Consider the below graph. Apply Floyd Warshall's algorithm to find shortest path between all pair of nodes.



Floyd-Warshall algorithm

- When k = 0

$D_0 =$

	1	2	3	4
1	0	8	∞	1
2	∞	0	1	∞
3	4	∞	0	∞
4	∞	2	9	0

- When k =1

$D_1 =$

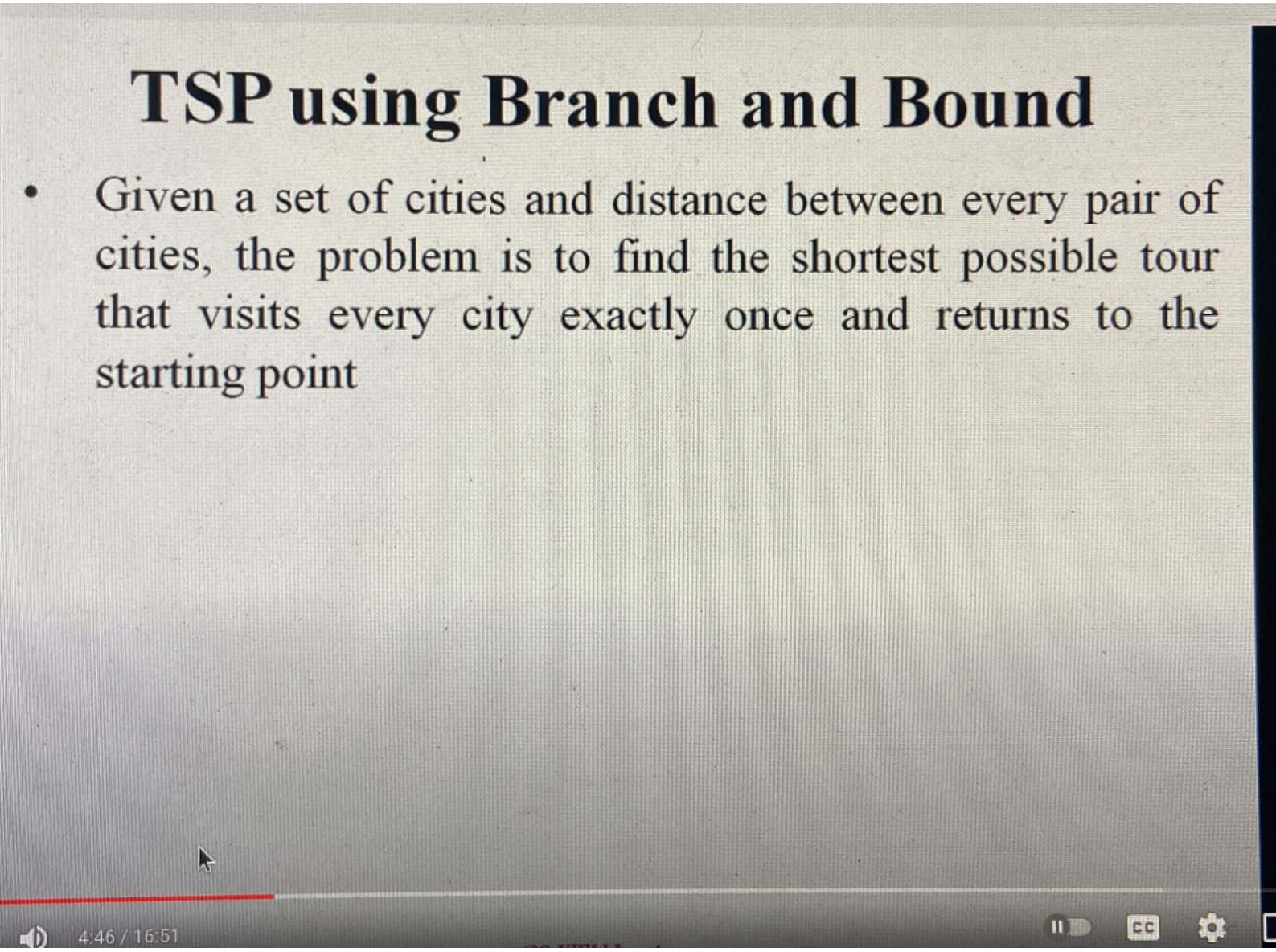
	1	2	3	4
1	0	8	∞	1
2	∞	0	1	∞
3	4	12	0	5
4	∞	2	9	0

Branch and Bound

- During state space tree generation E-node remains E-node until it is dead.
- Two strategies:
  - Breadth First Search(BFS)
    - It is called FIFO(First In First Out). Here the live nodes are placed in a queue.
  - Depth Search(D-Search)
    - It is called LIFO(Last In First Out). Here the live nodes are placed in a stack

TSP using Branch and Bound

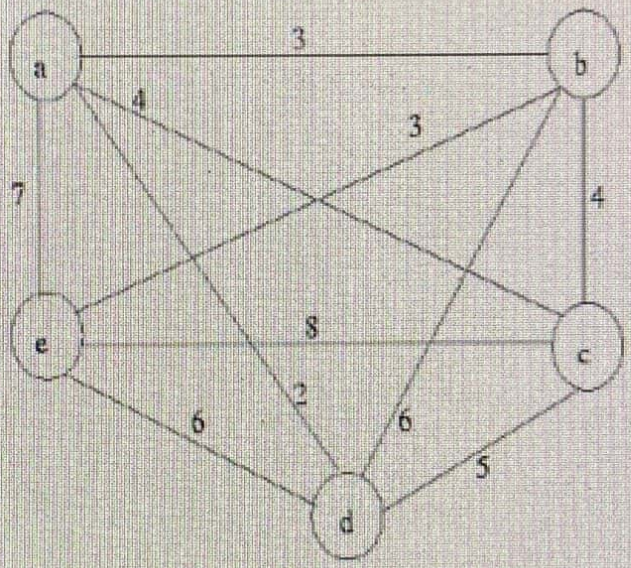
- Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible tour that visits every city exactly once and returns to the starting point



Step-01:

- Write the initial cost matrix and reduce it-

	A	B	C	D	E
A	∞	3	4	2	7
B	3	∞	4	6	3
C	4	4	∞	5	8
D	2	6	5	∞	6
E	7	3	8	6	∞



Rules

- To reduce a matrix, perform the row reduction and column reduction of the matrix separately.
- A row or a column is said to be reduced if it contains at least one entry '0' in it.

Step-02:

- We consider all other vertices one by one.
- We select the best vertex where we can land upon to minimize the tour cost.

Choosing To Go To Vertex-B: Node-2 (Path A → B)

- From the reduced matrix of step-01, M[A,B] = 1
- Set row-A and column-B to ∞
- Set M[B,A] = ∞

Now, resulting cost matrix is-

	A	B	C	D	E
A	—	1	1	0	5
B	0	—	0	3	0
C	0	0	—	1	4
D	0	4	2	—	4
E	4	0	4	3	—

	A	B	C	D	E
A	∞	∞	∞	∞	∞
B	∞	∞	0	3	0
C	0	∞	∞	1	4
D	0	∞	2	∞	4
E	4	∞	4	3	∞

- Now, We reduce this matrix.
- Then, we find out the cost of node-02.

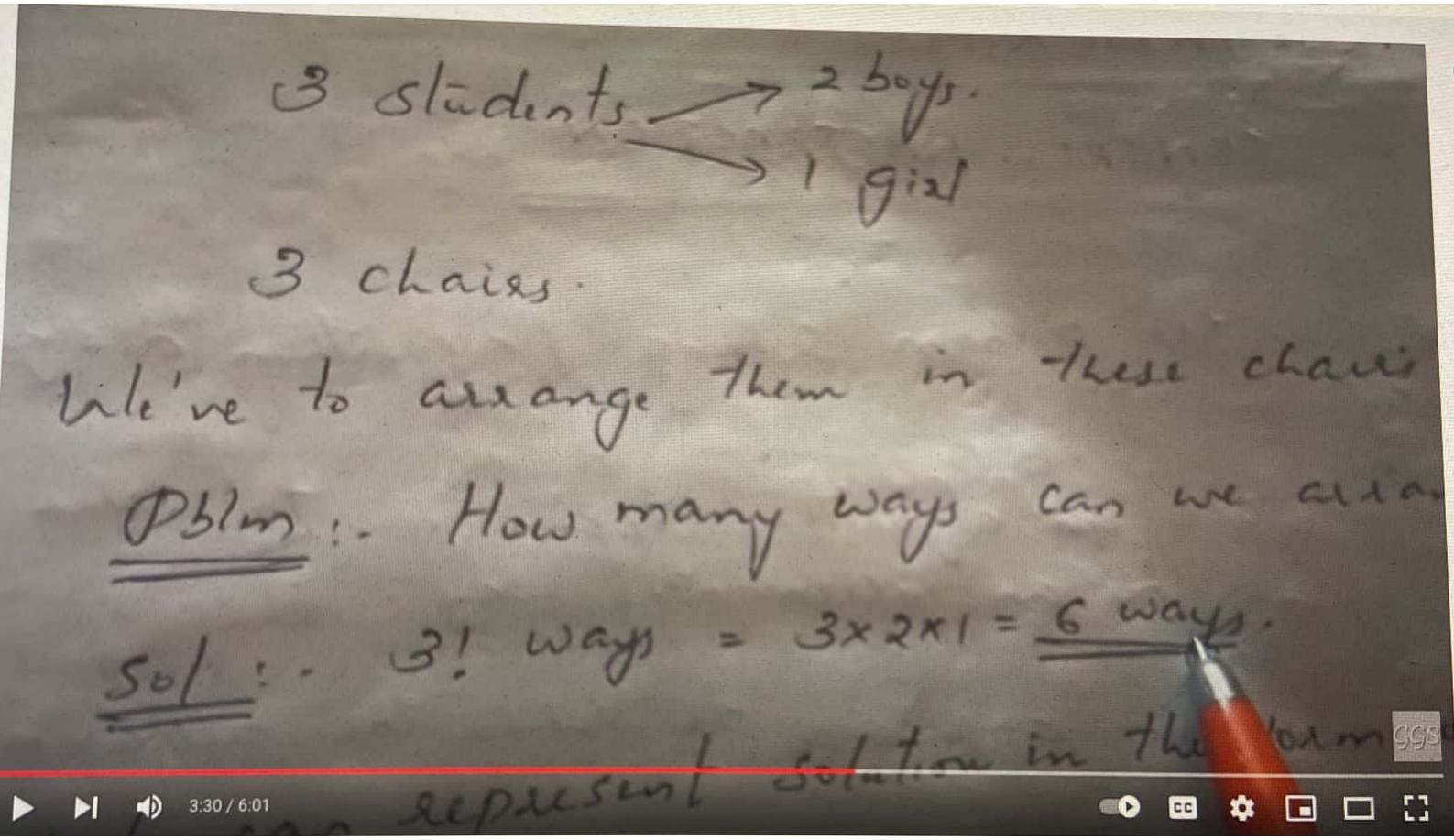
Backtracking

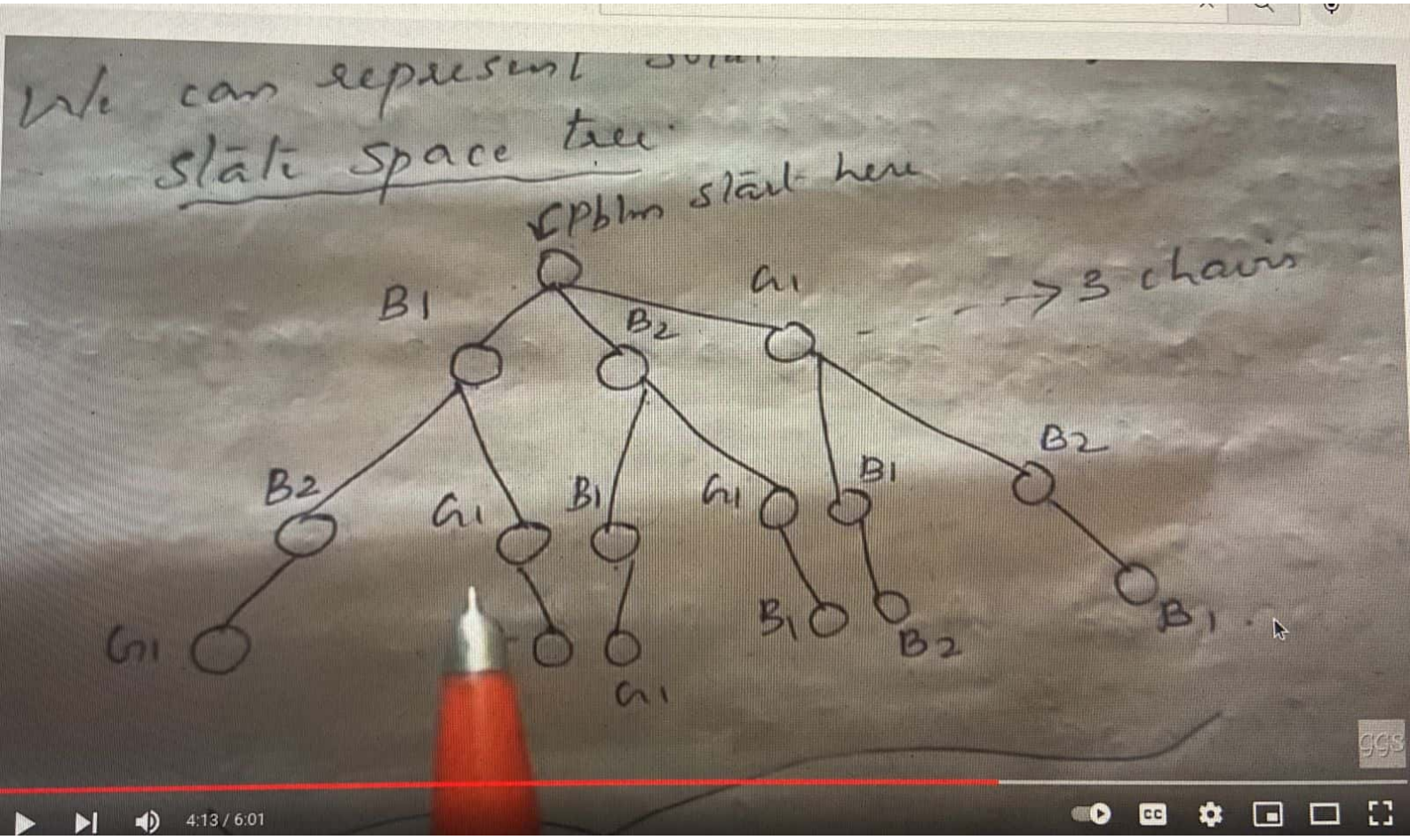
- Suppose you have to make a series of *decisions,* among various *choices,* where
  - You don't have enough information to know what to choose
  - Each decision leads to a new set of choices
  - Some sequence of choices (possibly more than one) may be a solution to your problem
- **Backtracking** is a methodical way of trying out various sequences of decisions, until you find one that "works"

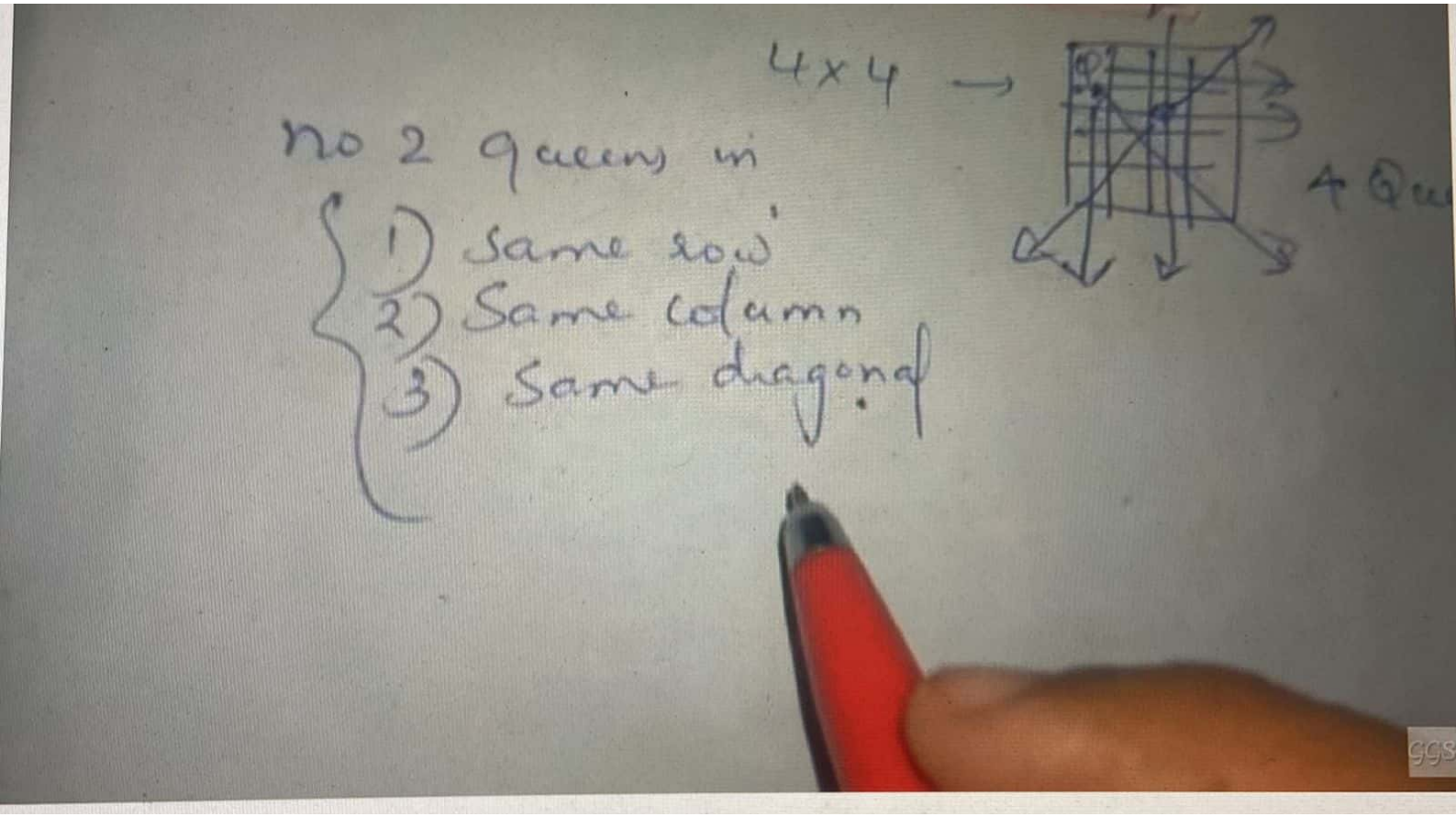
3) Enumeration problem:
possible feasible solutions

Applications of backtracking

1) N Queen's problem
2) Graph coloring
3) 0/1 knapsack pblm.
4) Hamiltonian cycles.
5) Sum of sub







Back Tracking – The N Queen's Problem.