Video Lectures On

Artificial Intelligence

Lecture 09

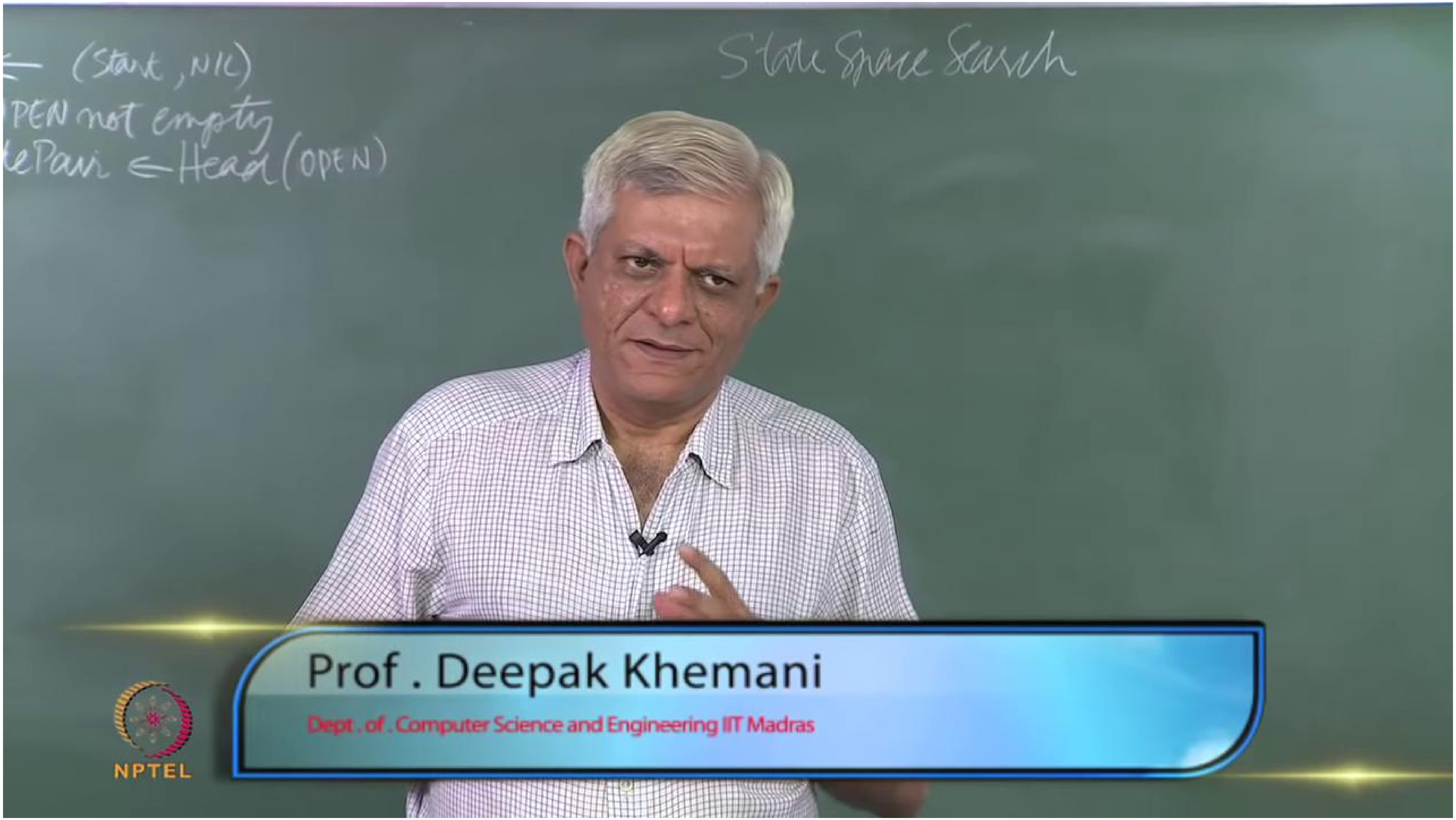
Z

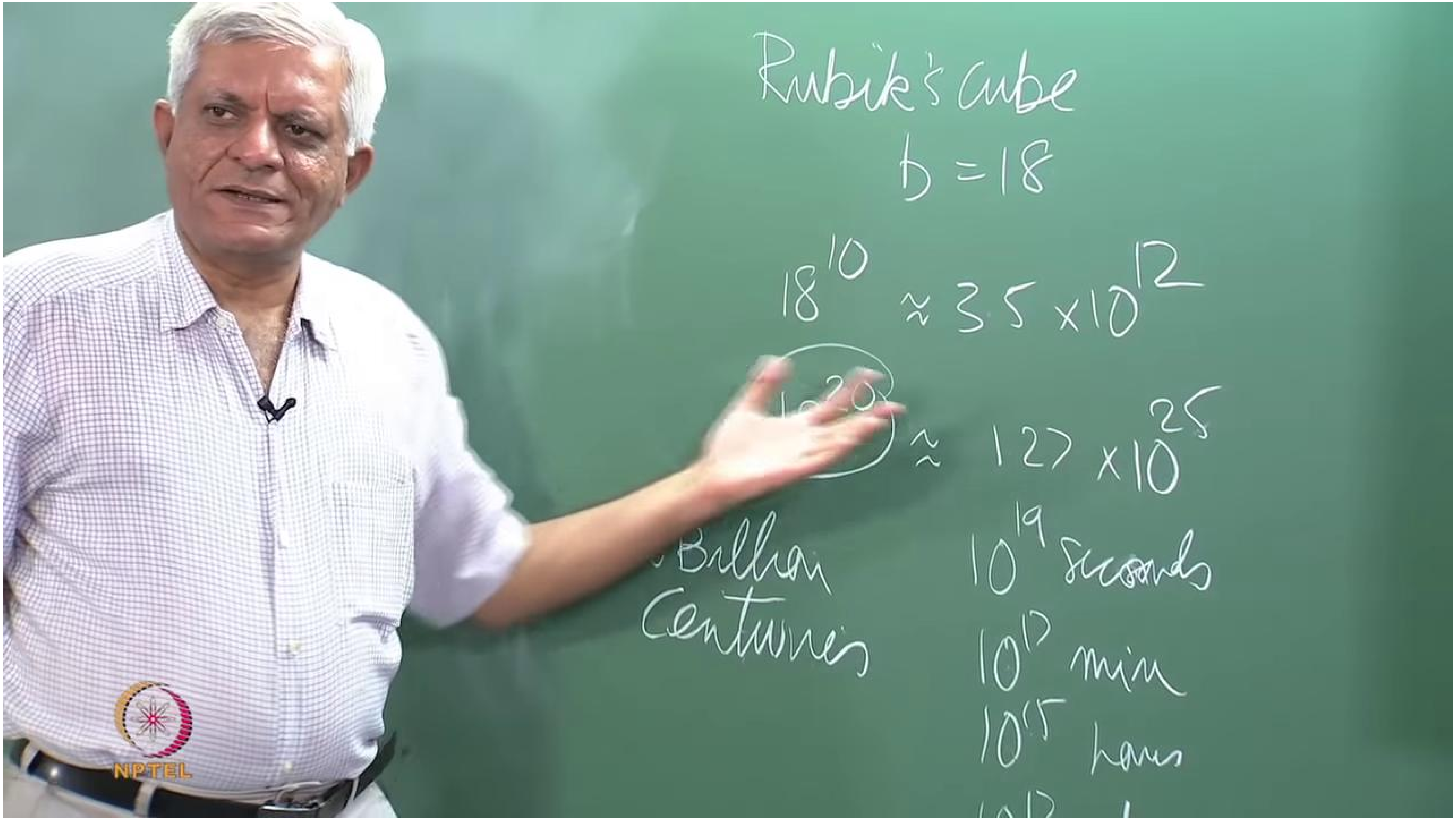
Prof. Deepak Khemani

Department of Computer Science and Engineering

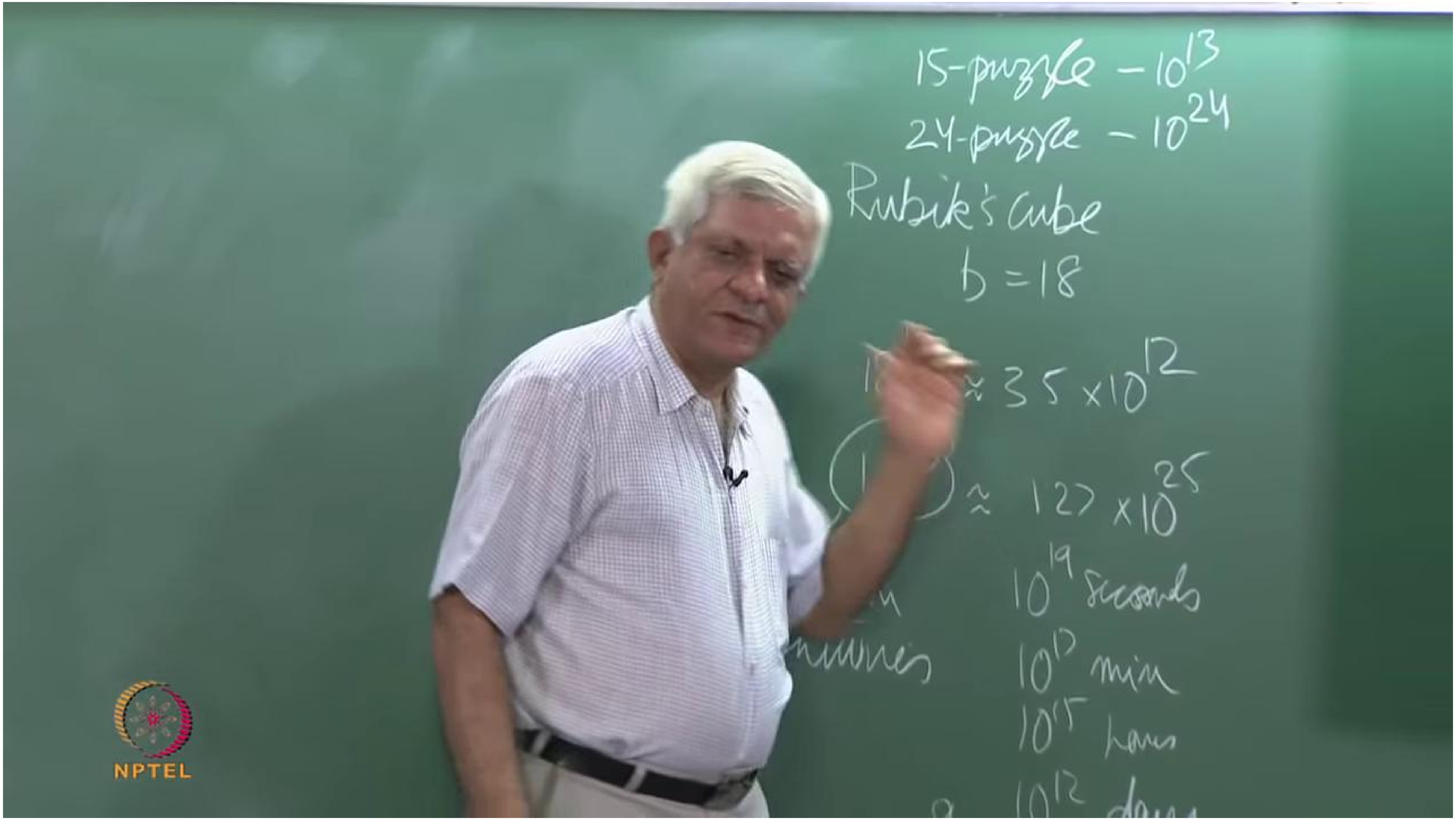
IIT Madras

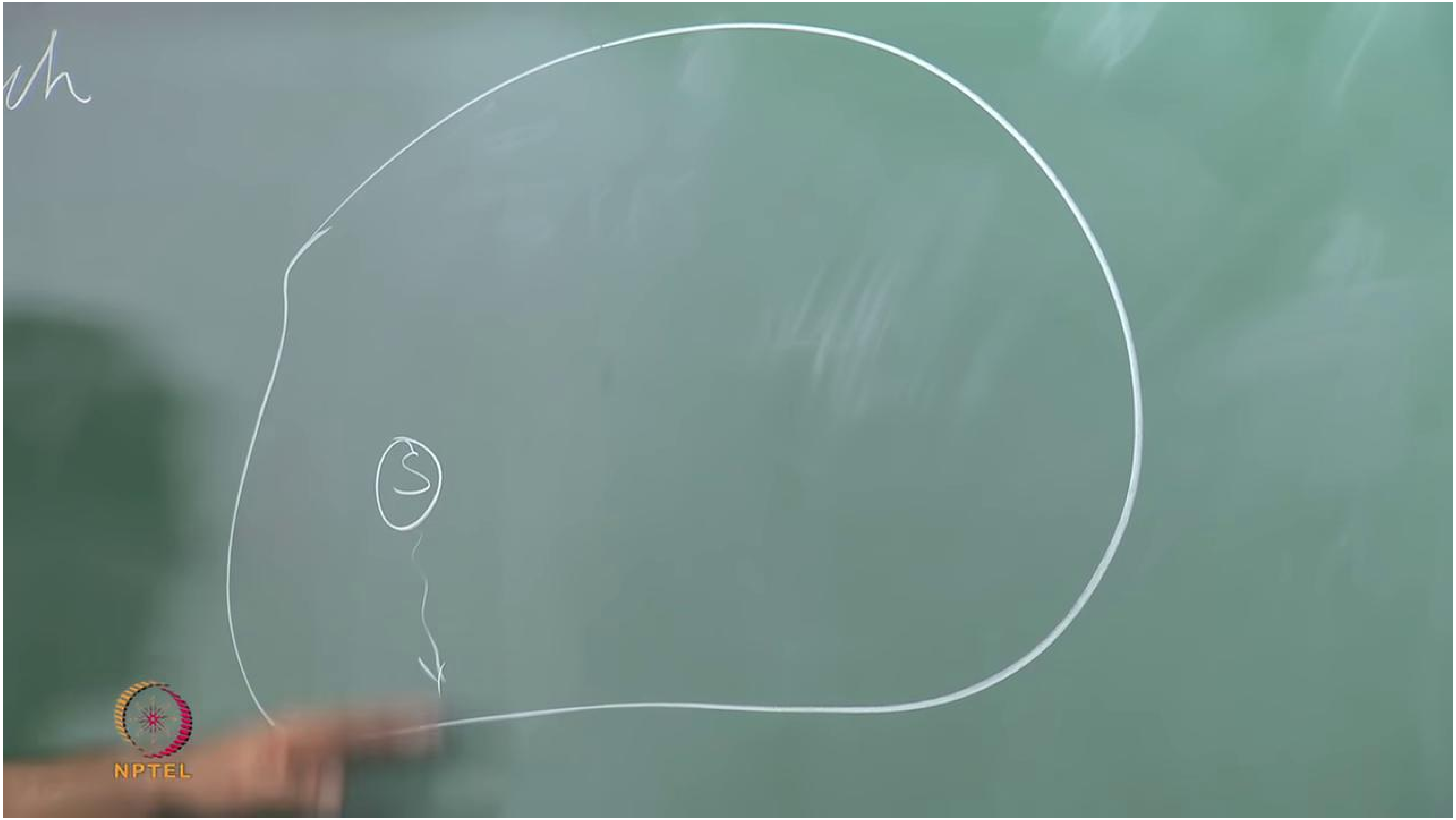
NPTEL

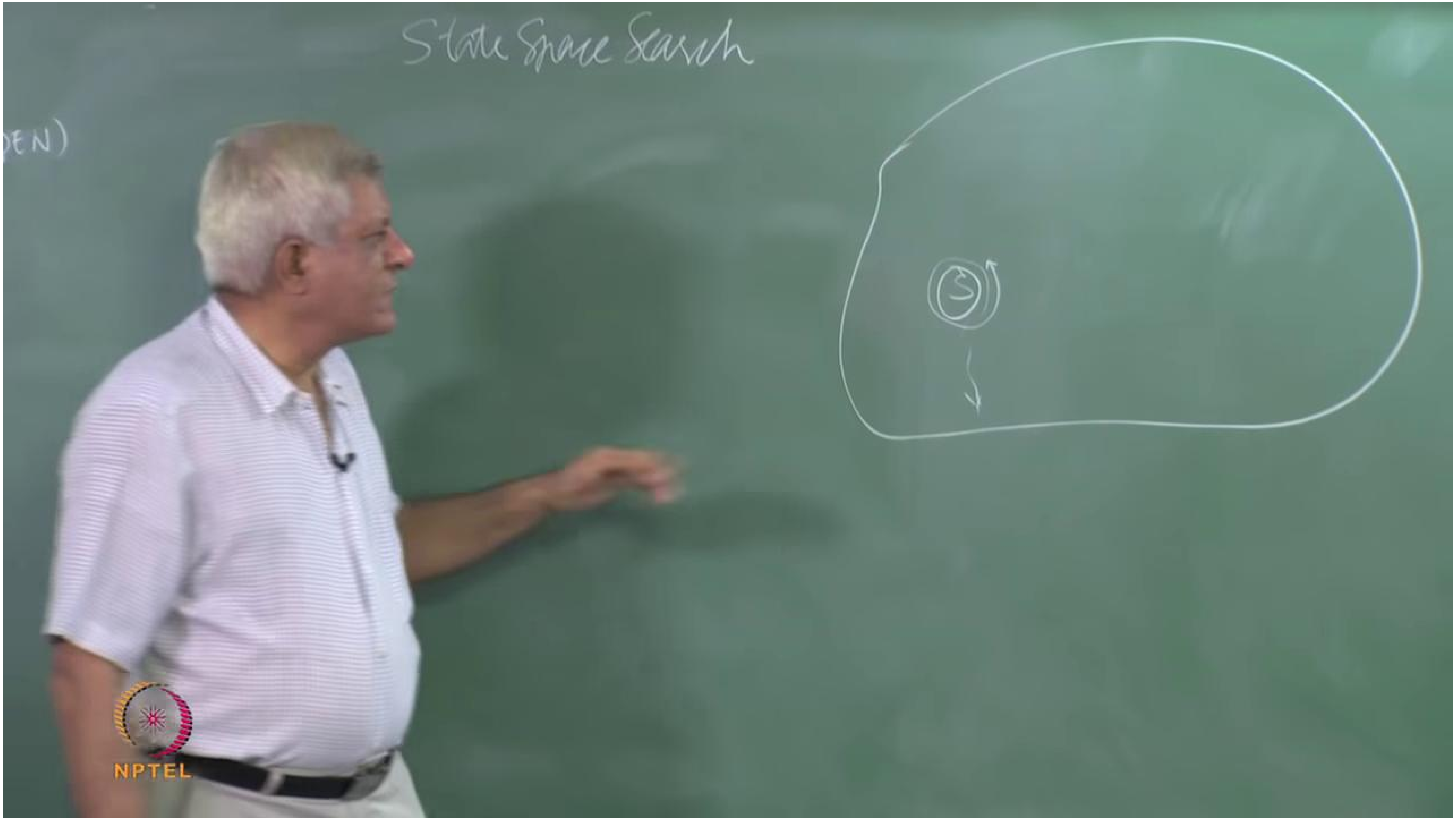

← (Start , NIL)
PEN not empty
ePair ← Head(OPEN)
State Space Search
Prof . Deepak Khemani
Dept . of . Computer Science and Engineering IIT Madras
NPTEL

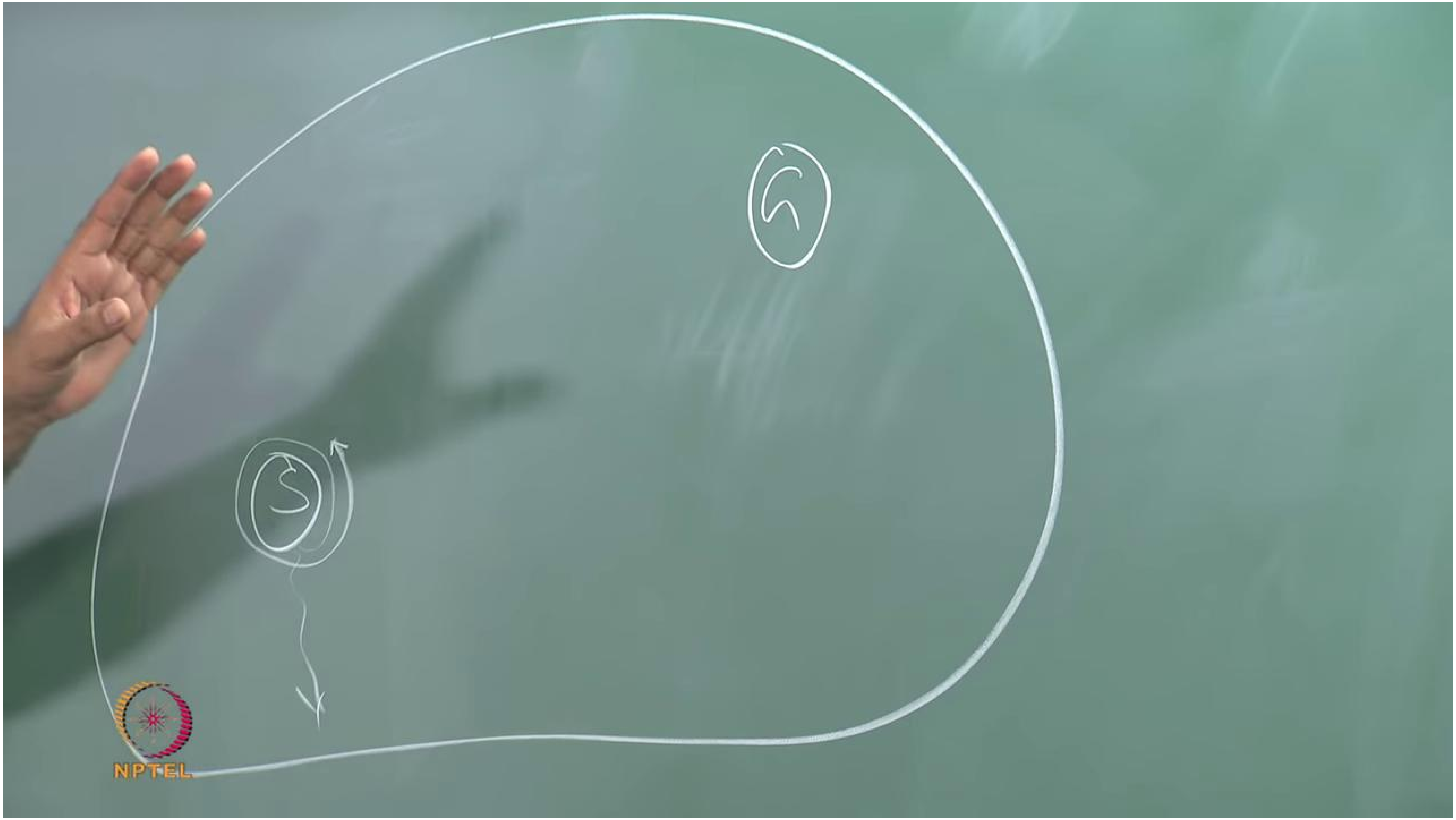


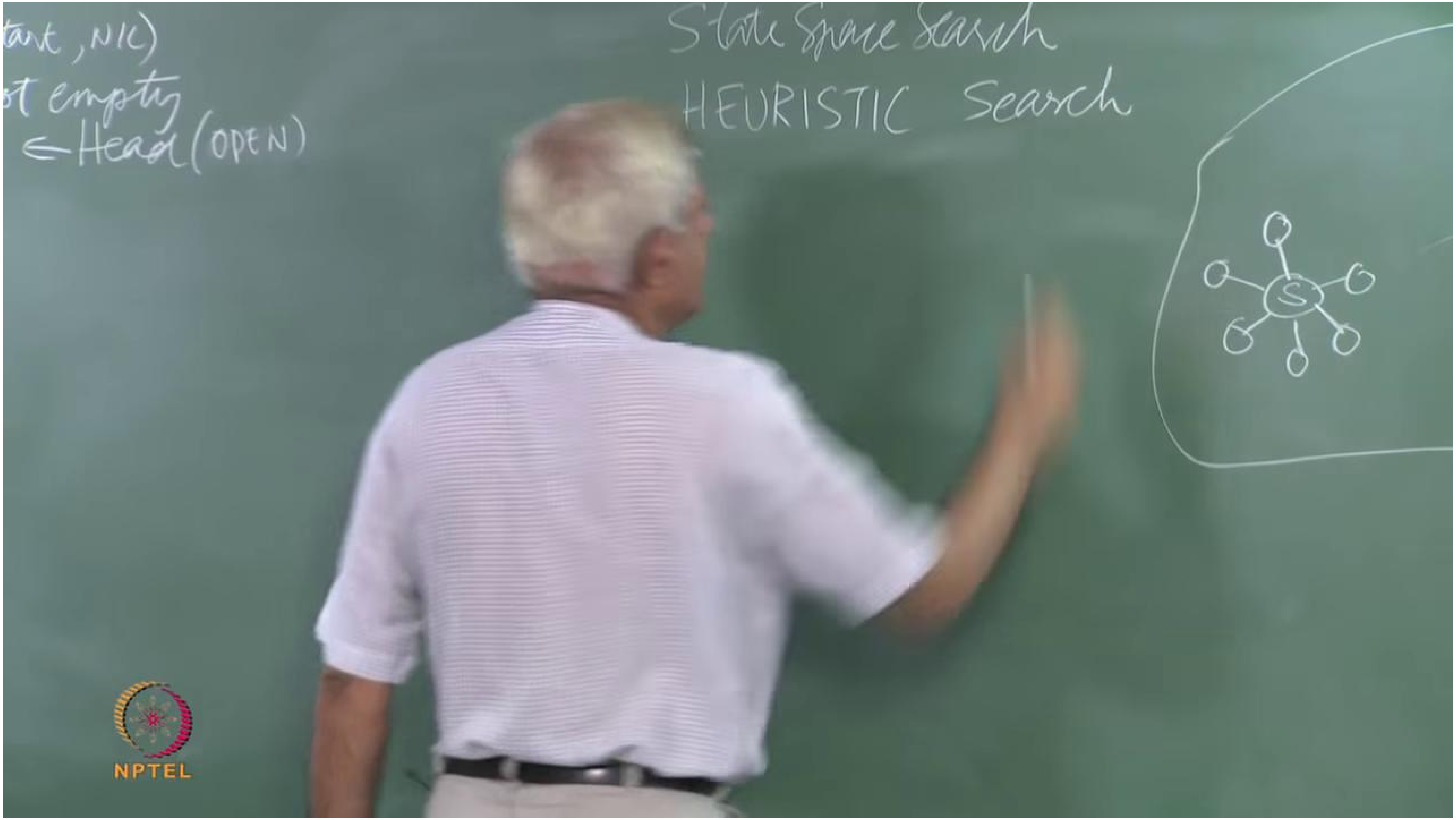
Rubik's cube
b = 18
18^10 ≈ 3.5 × 10^12
≈ 127 × 10^25
Billion
Centuries
10^19 seconds
10^17 min
10^15 hours
NPTEL


15-puzzle – 10^13
24-puzzle – 10^24
Rubik's cube
b = 18
≈ 3.5 × 10^12
≈ 1.27 × 10^25
10^19 seconds
10^17 min
10^15 hours
10^12 days
NPTEL


S


State Space Search
S
NPTEL


S
G


tart, NIL)
ot empty
← Head(OPEN)
State Space Search
HEURISTIC Search
S
NPTEL

HEURISTIC Search

HEURISTIC Search

Eurisko

Heuriskein

HEURISTIC Search

Eurisko

Eureka

Heuriskein

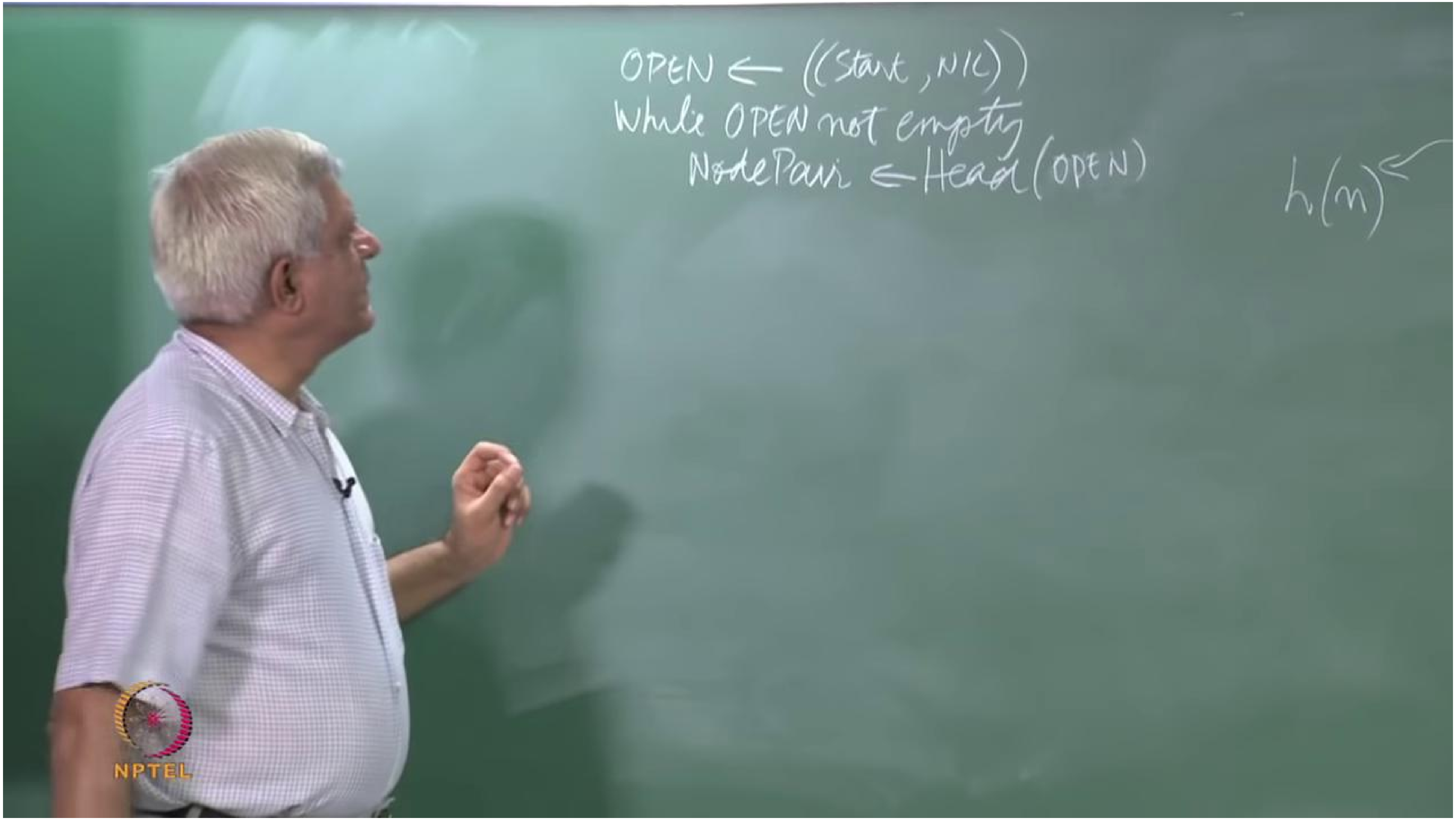
HEURISTIC Search

$h(n)$

Eurisko

Eureka

Heuriskein


OPEN ← ((Start, NIL))
While OPEN not empty
NodePair ← Head(OPEN)
h(n)
NPTEL

OPEN $\leftarrow$ ((Start, NIL))

While OPEN not empty

NodePair $\leftarrow$ Head(OPEN)

⋮

New

OPEN $\leftarrow$ $Sort_h$(Append(New Tail(OPEN)))

HEURISTI

$h(n)$

Eureka

Eu

NPTEL

OPEN ← ((Start, NIL))
While OPEN not empty
NodePair ← Head(OPEN)

⋮

New

OPEN ← $Sort_h$(Append(New Tail(OPEN)))

HEURIST

$h(n)$

Eu

Eureka

NPTEL

OPEN ← ((Start, NIL))
While OPEN not empty
  NodePair ← Head(OPEN)

HEURIST

$h(n)$

Eu

Eureka

New

OPEN ← Sort_h(Append(New Tail(OPEN)))

NPTEL

OPEN ← ((Start, NIL))

While OPEN not empty

NodePair ← Head(OPEN)

heuristic function

HEURISTI

$h(n)$

Eu

Eureka

New

OPEN ← $Sort_h$(Append(New Tail(OPEN)))

NPTEL

OPEN ← ((Start, NIL))
While OPEN not empty
NodePair ← Head(OPEN)

heuristic function

HEURISTIC

h(n)

Eurisko

Eureka

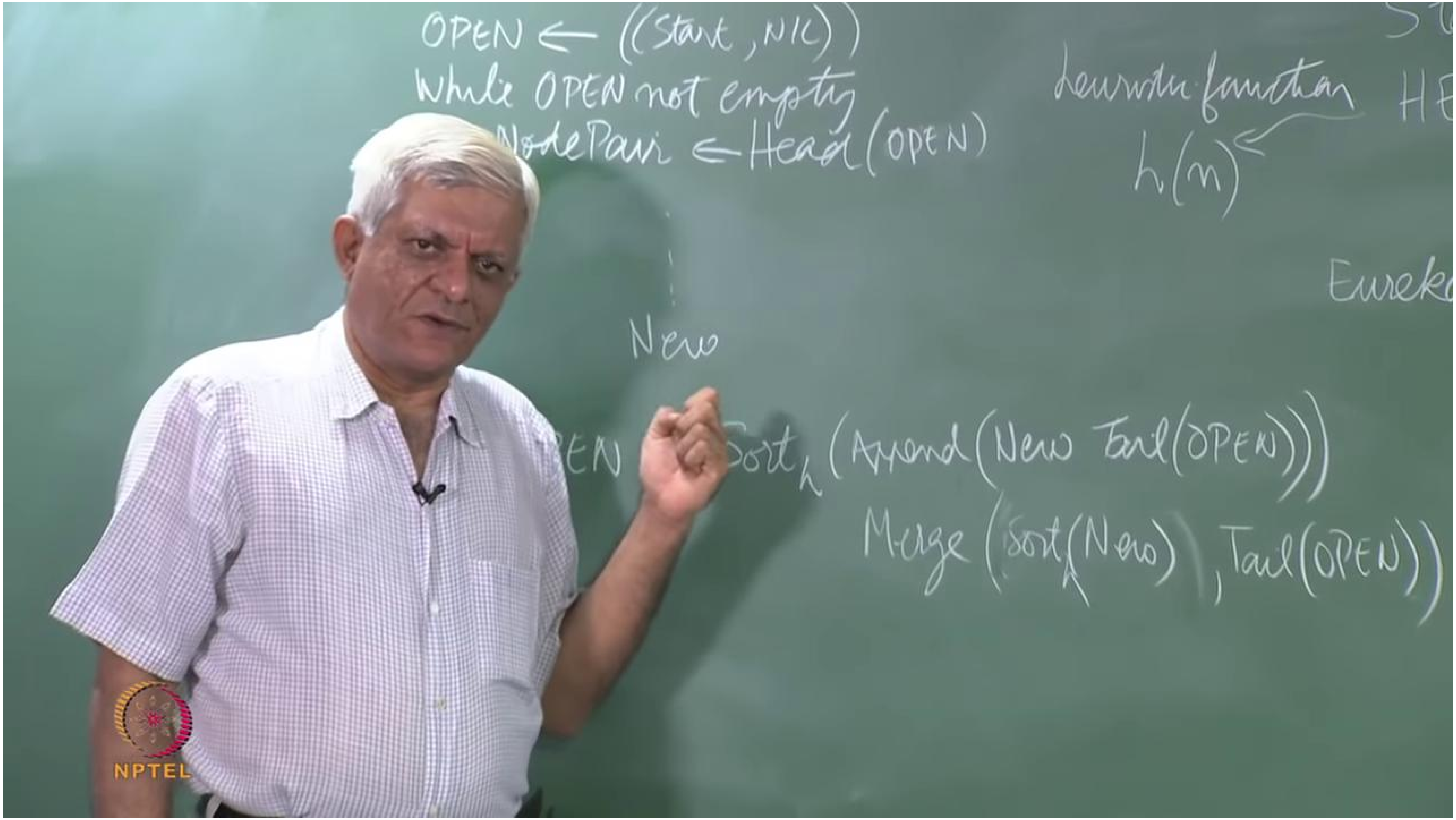
Heur

New

OPEN ← Sort_h(Append(New Tail(OPEN)))

Merge(Sort(New), Tail(OPEN))

NPTEL



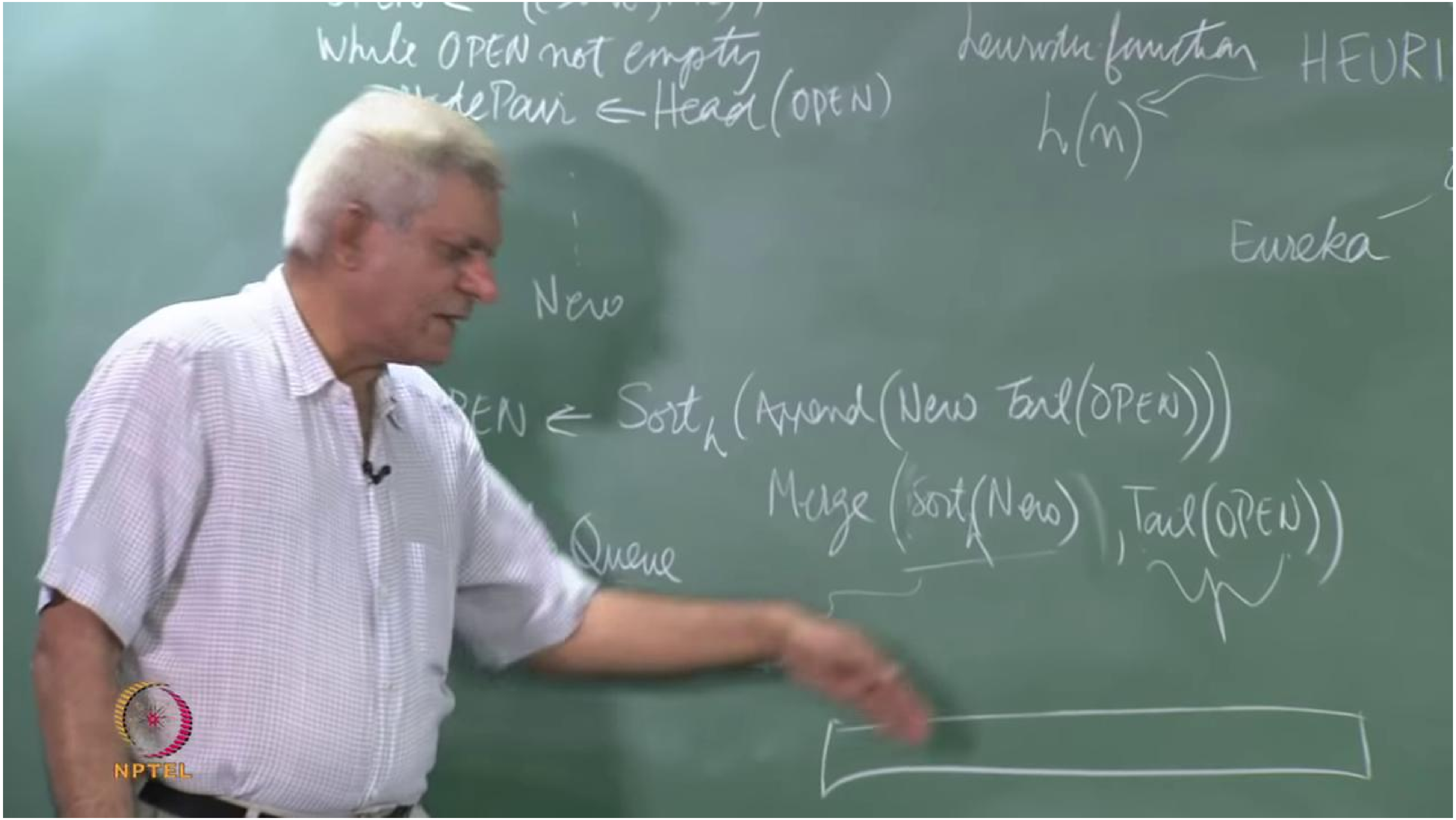
OPEN ← ((Start, NIL))
While OPEN not empty
NodePair ← Head(OPEN)
heuristic function
h(n)
New
Sort_h(Append(New Tail(OPEN)))
Merge(Sort_h(New), Tail(OPEN))
NPTEL

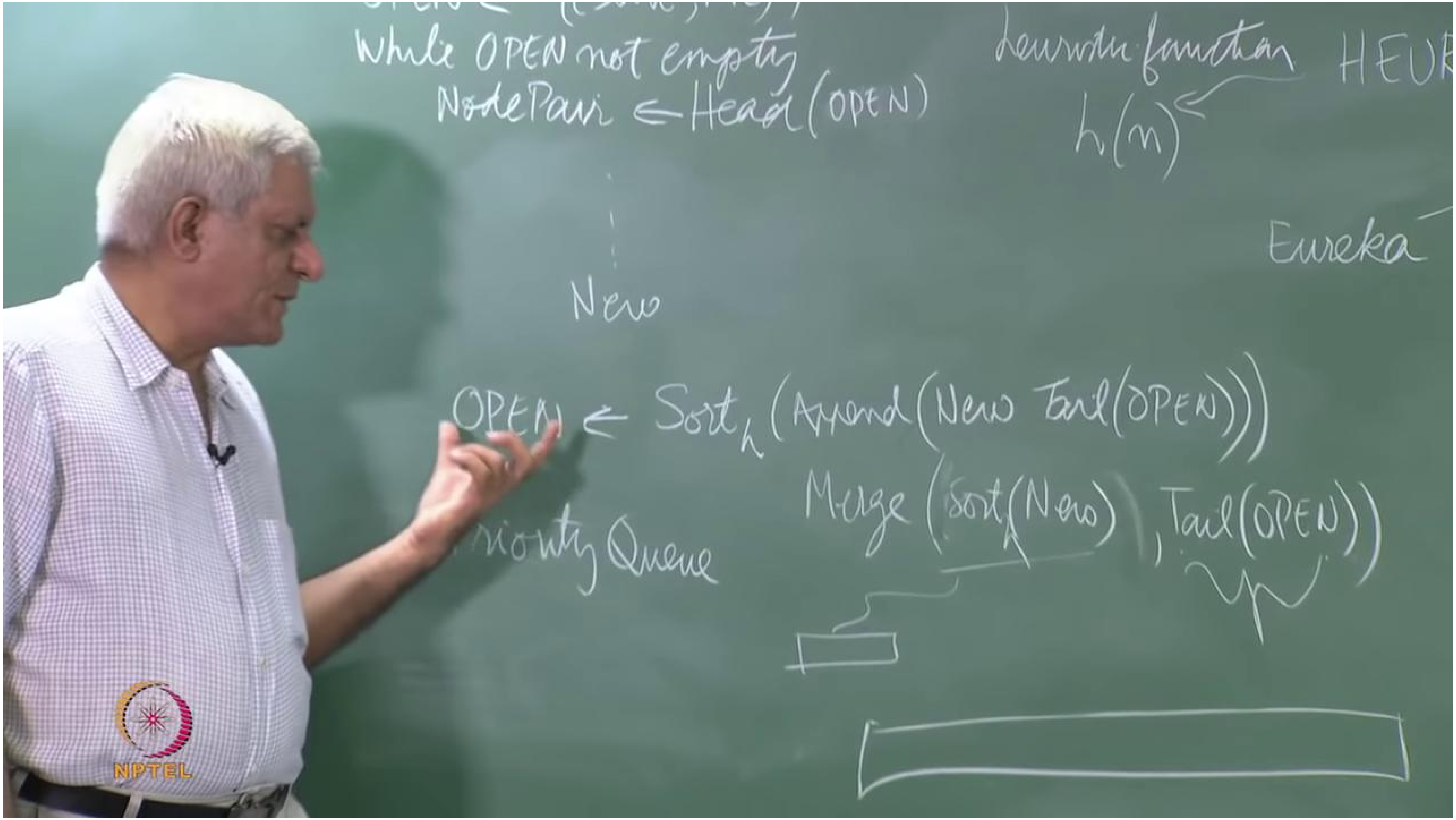
New

$$OPEN \leftarrow Sort_h(Append(New\ Tail(OPEN)))$$

$$Merge(Sort_h(New), Tail(OPEN))$$

Priority Queue


While OPEN not empty
dePair ← Head(OPEN)
heuristic function
HEURI
h(n)
Eureka
New
EN ← Sort_h(Append(New Tail(OPEN)))
Merge(Sort_h(New), Tail(OPEN))
Queue


While OPEN not empty
NodePair ← Head(OPEN)
heuristic function
HEUR
h(n)
Eureka
New
OPEN ← Sort_h(Append(New Tail(OPEN)))
Merge(Sort_h(New), Tail(OPEN))
riority Queue

New

$$OPEN \leftarrow Sort_h(Append(New\ Tail(OPEN)))$$

Priority Queue

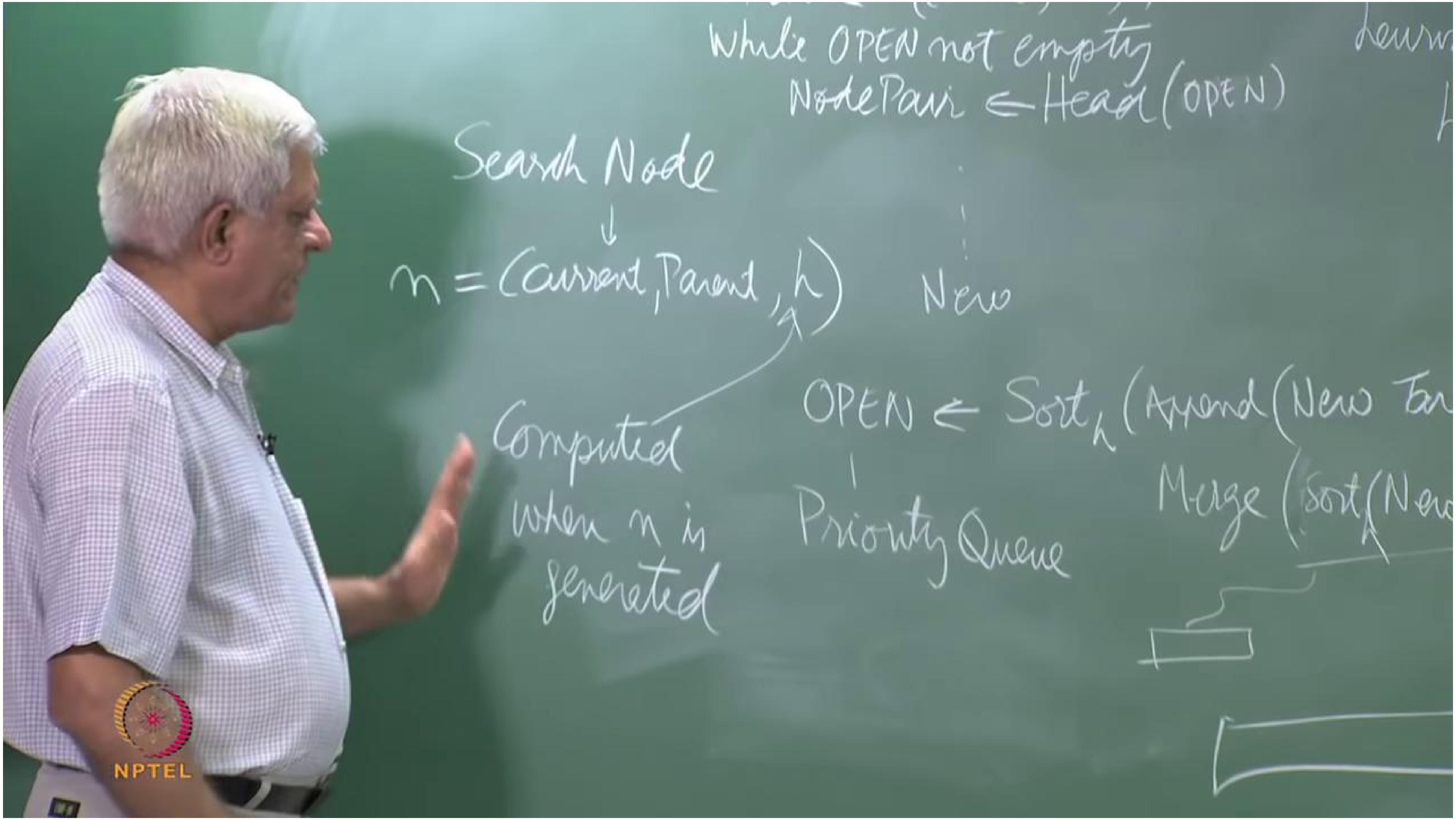
$$Merge(\underline{Sort_h(New)}, Tail(OPEN))$$

Search Node

$\downarrow$

(Current, Parent, h)

No


While OPEN not empty
NodePair ← Head(OPEN)
Search Node
n = (Current, Parent, h)
New
Computed
when n is
generated
OPEN ← Sort
Priority Queue
Merge
NPTEL

heuristic function HEURISTIC Search

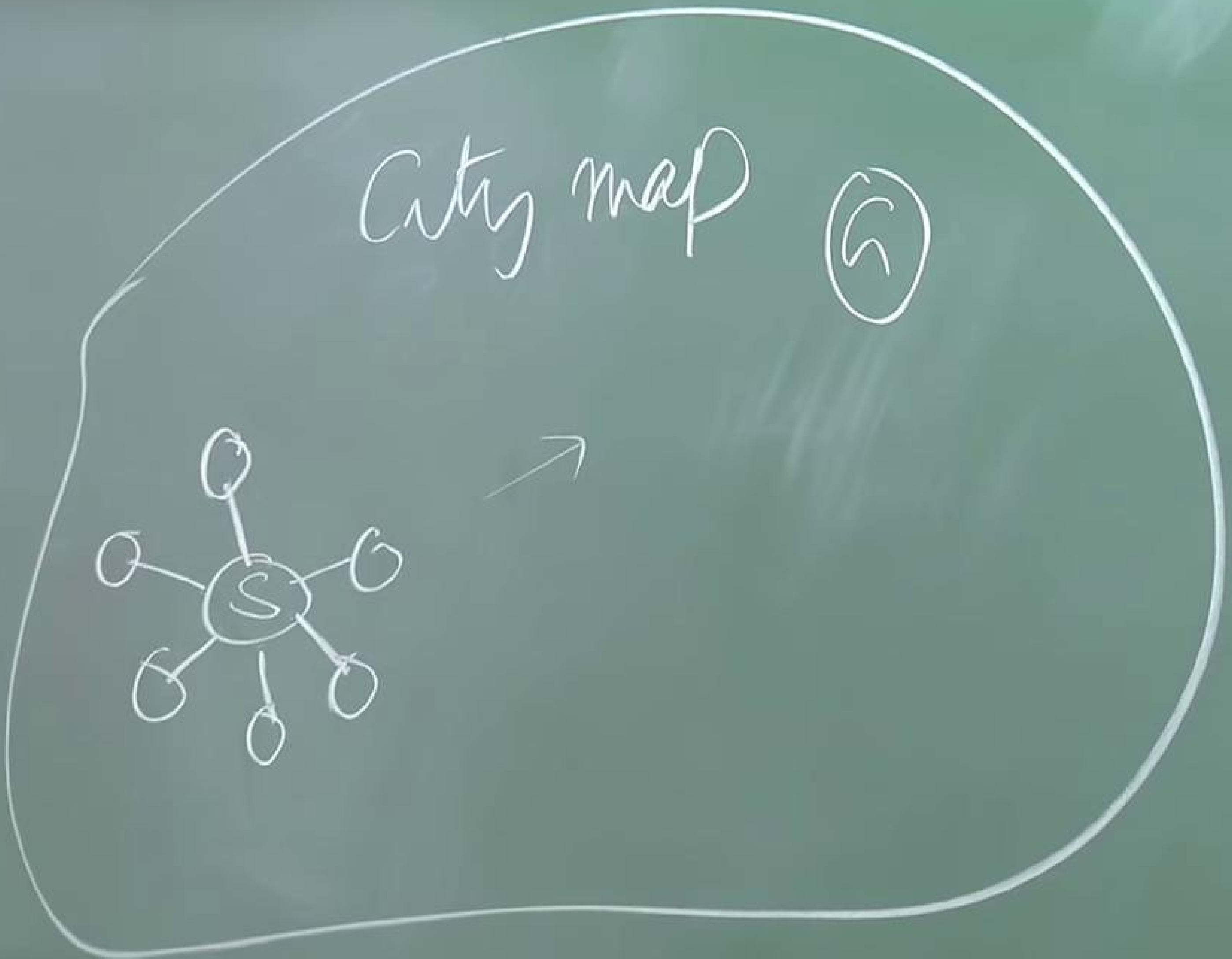
$h(n)$

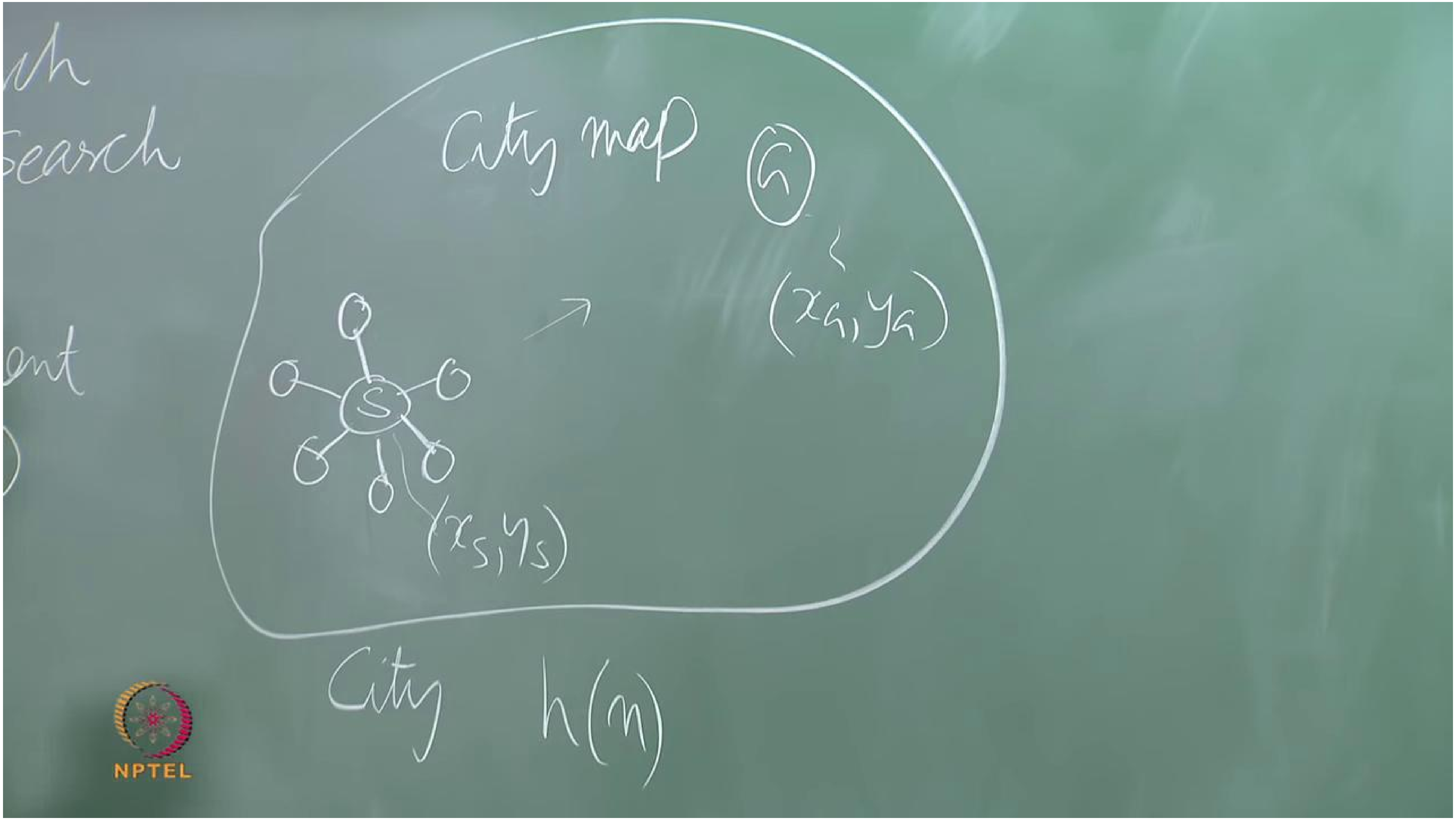
Domain Dependent

(static)

ch
Search

ent



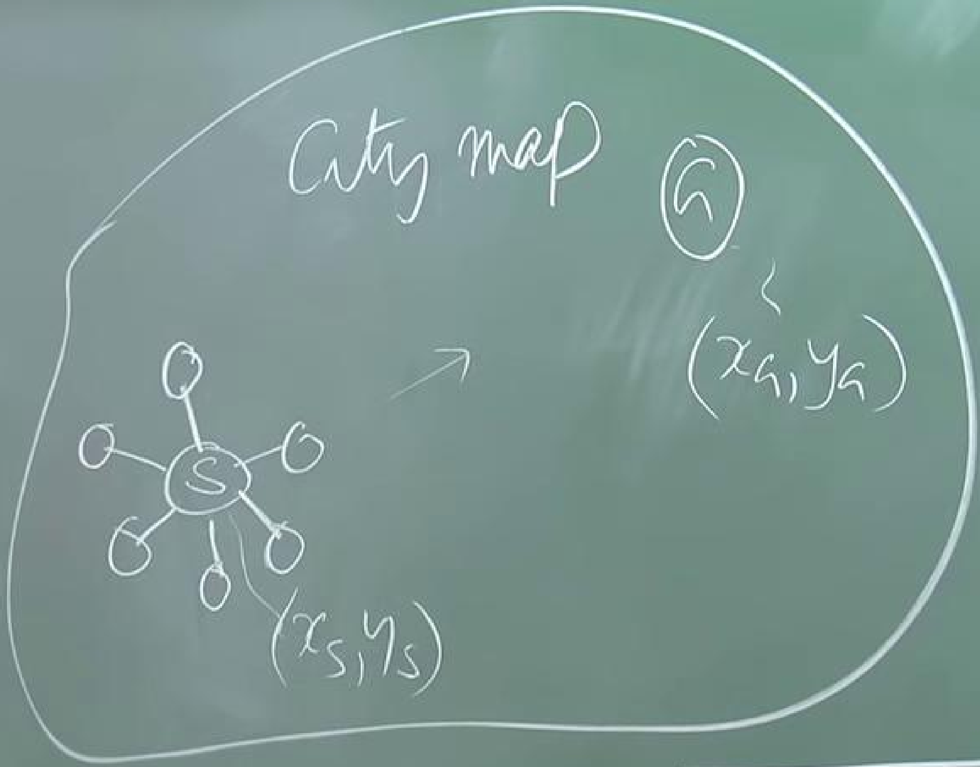

Search
City map
G
(x_G, y_G)
S
(x_S, y_S)
City
h(n)
NPTEL

...ce Search

...TIC Search

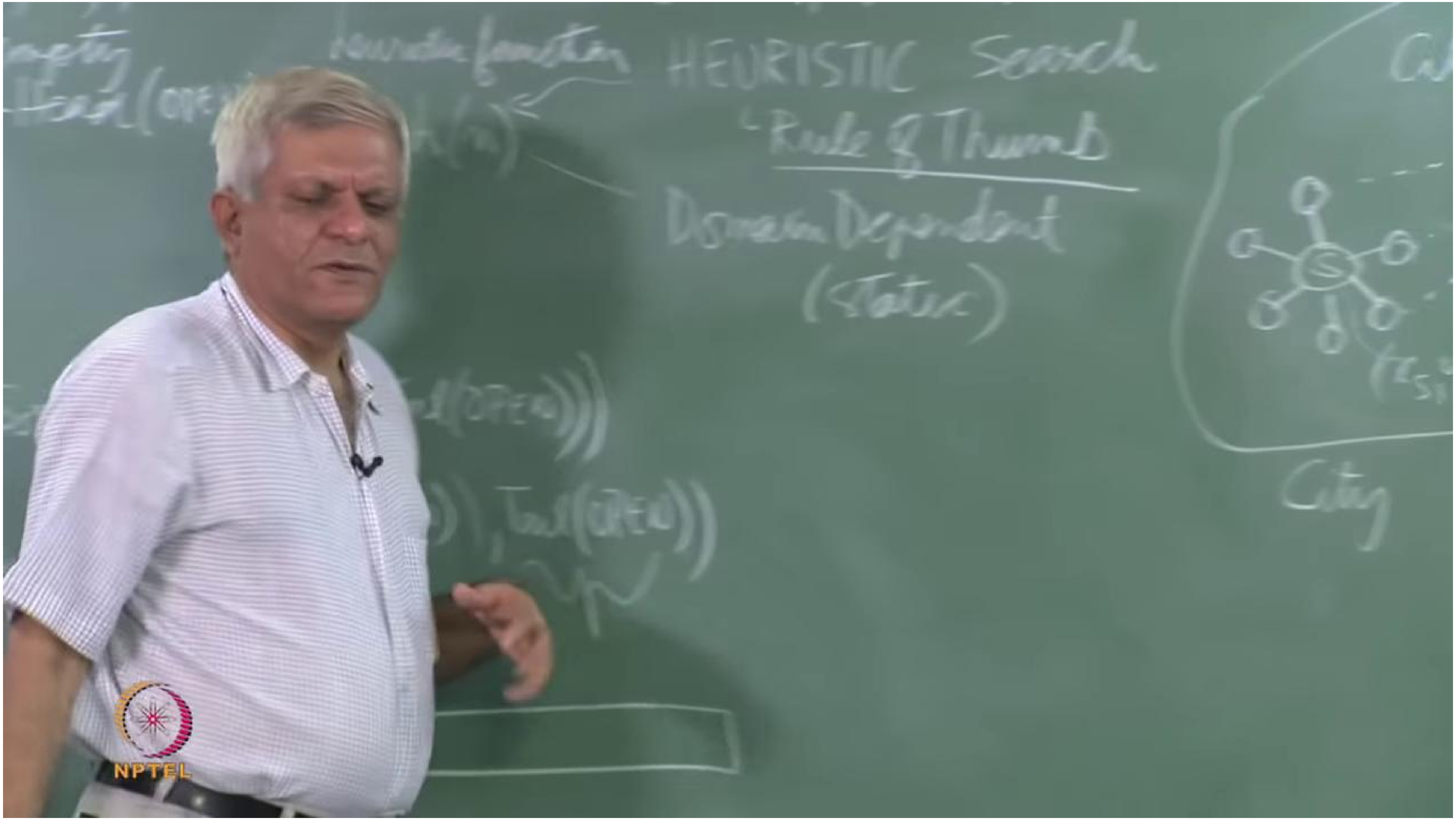
Dependent

(static)



City $h(n) = \sqrt{(x_S - x_G)^2 + (y_S - y_G)^2}$

Euclidean.


heuristic function
h(n)
HEURISTIC Search
Rule of Thumb
Domain Dependent
(States)
City
NPTEL

State Space Search

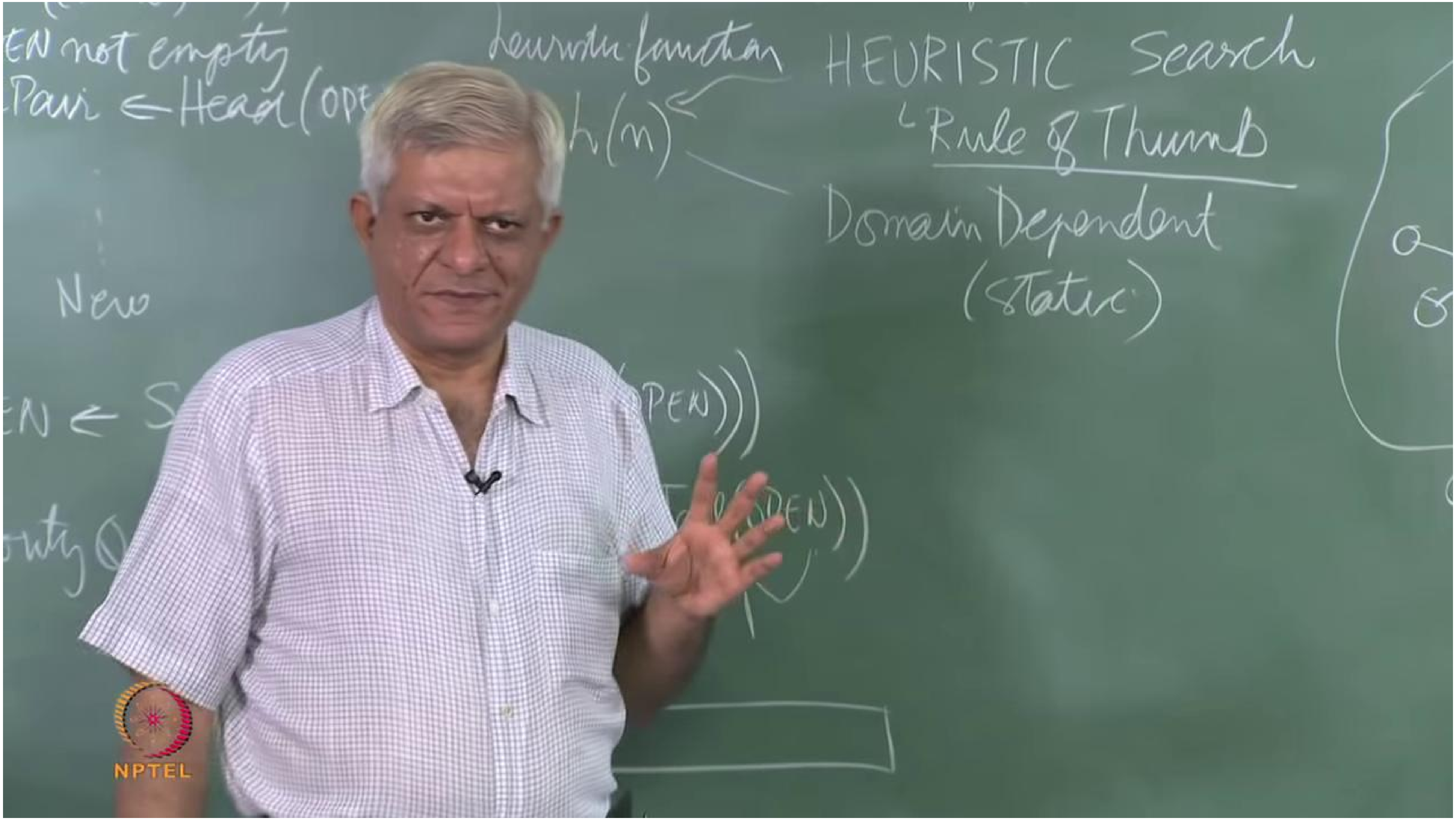
heuristic function HEURISTIC Search

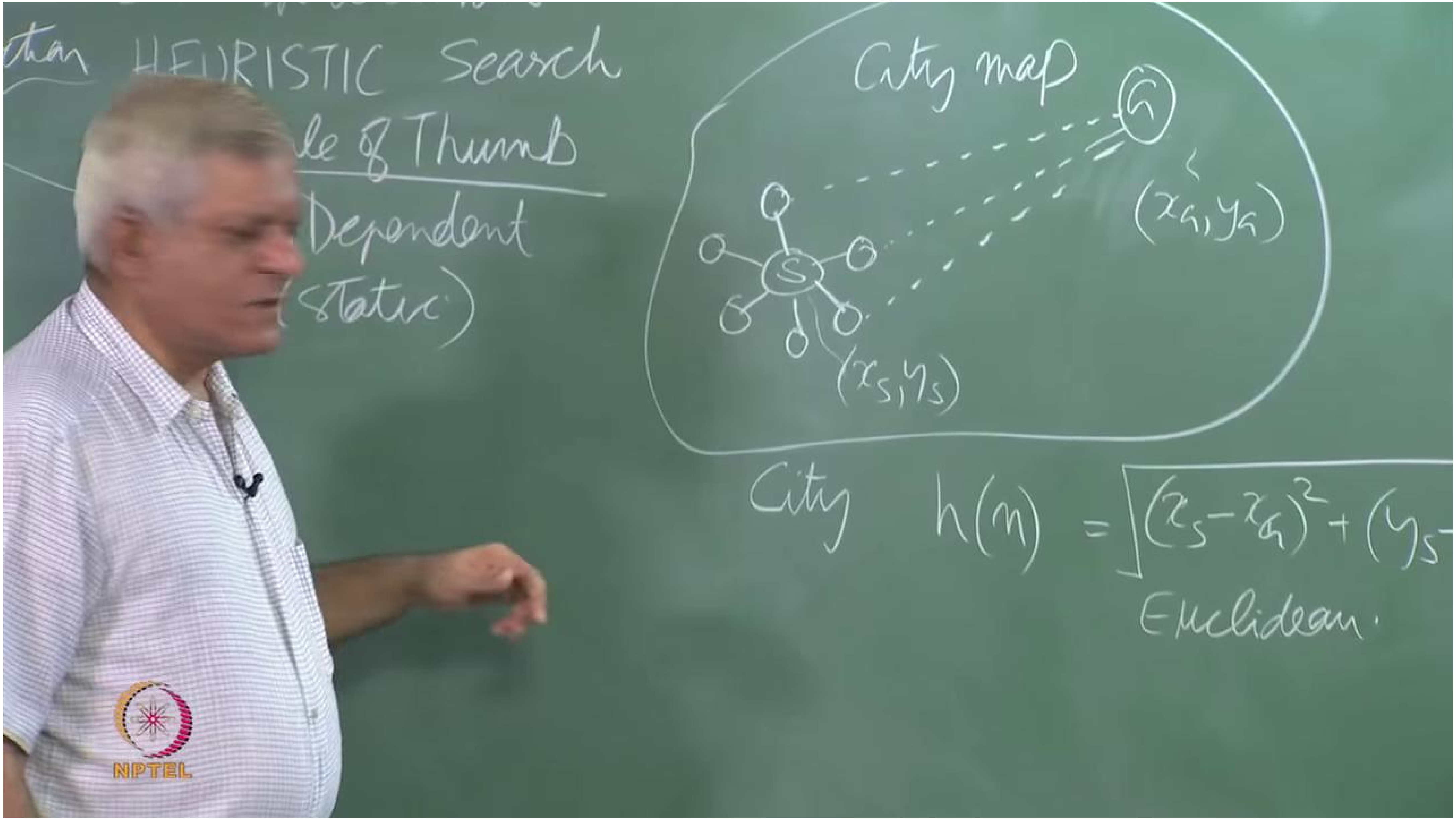
$h(n)$

"Rule of Thumb"

Domain Dependent

(static)


heuristic function
HEURISTIC Search
Rule of Thumb
h(n)
Domain Dependent
(static)
New
NPTEL


HEURISTIC Search
Thumb
Dependent
(static)
City map
G
(x_G, y_G)
S
(x_S, y_S)
City
h(n) = √((x_S − x_G)² + (y_S −
Euclidean
NPTEL

$$h(n) = \sqrt{(x_s - x_g)^2 + (y_s - y_g)^2}$$

Euclidean.

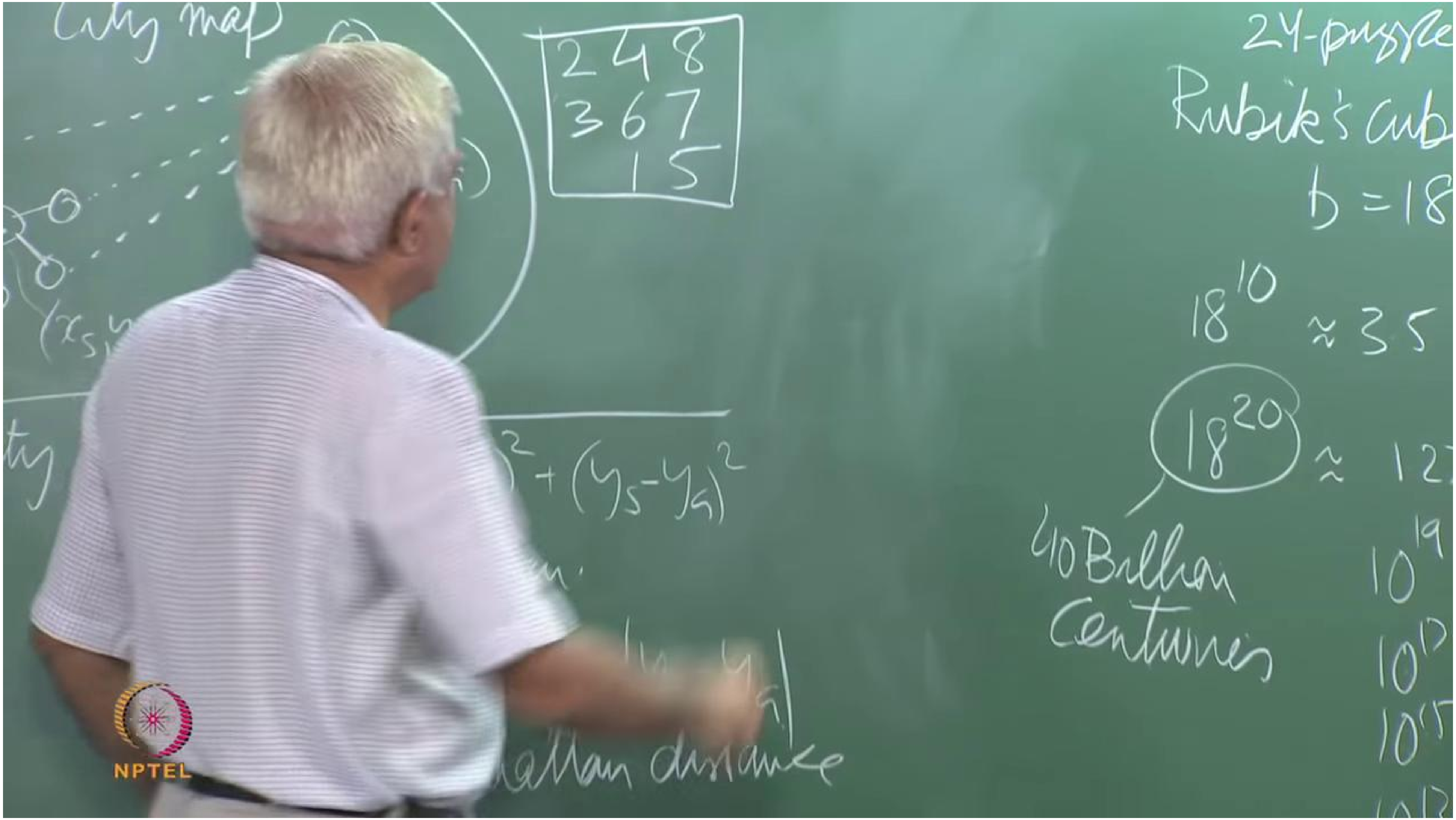
$$= |x_s - x_g| + |y_s - y_g|$$

City $h(n) = \sqrt{(x_s - x_g)^2 + (y_s - y_g)^2}$

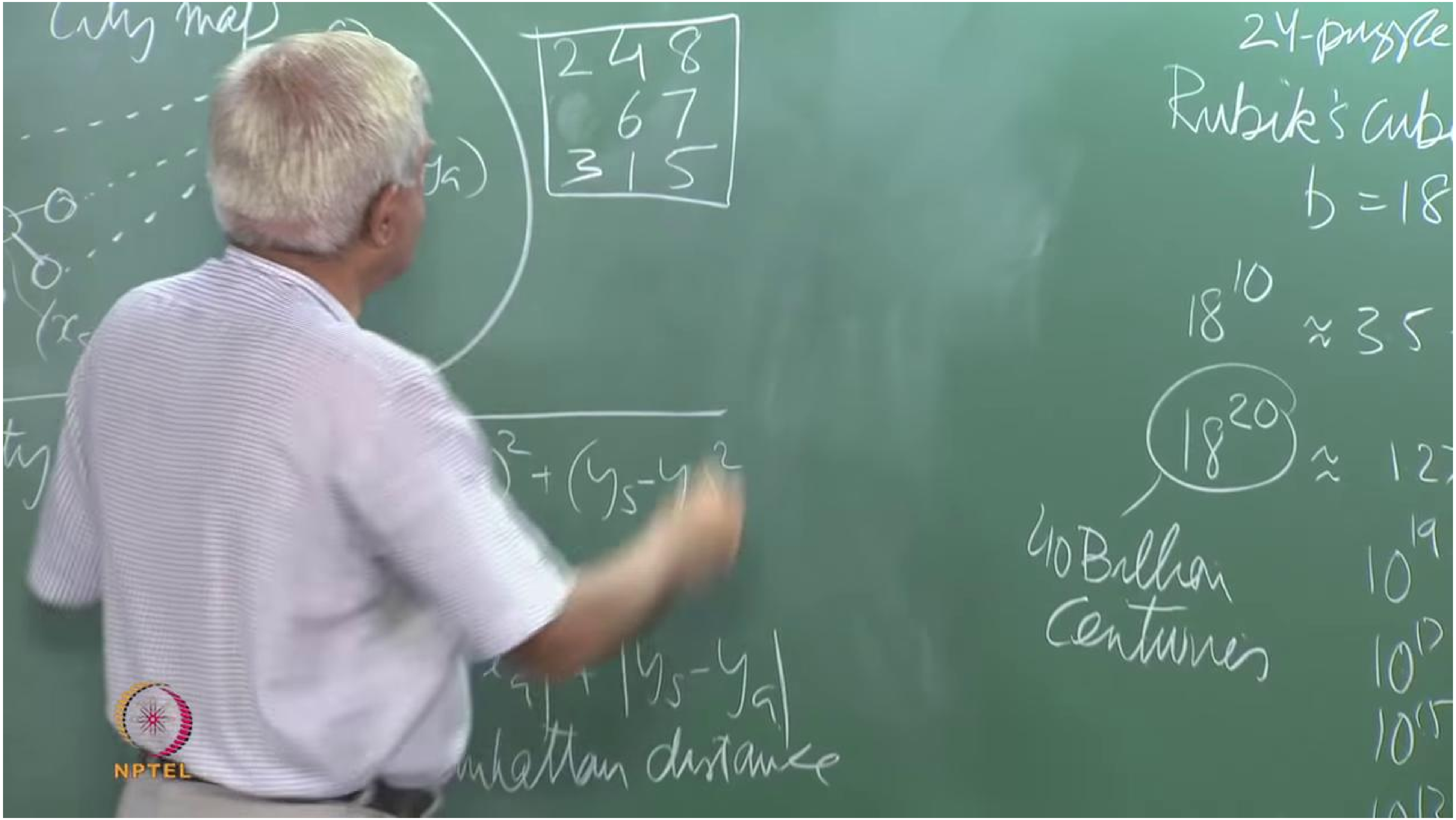
Euclidean.

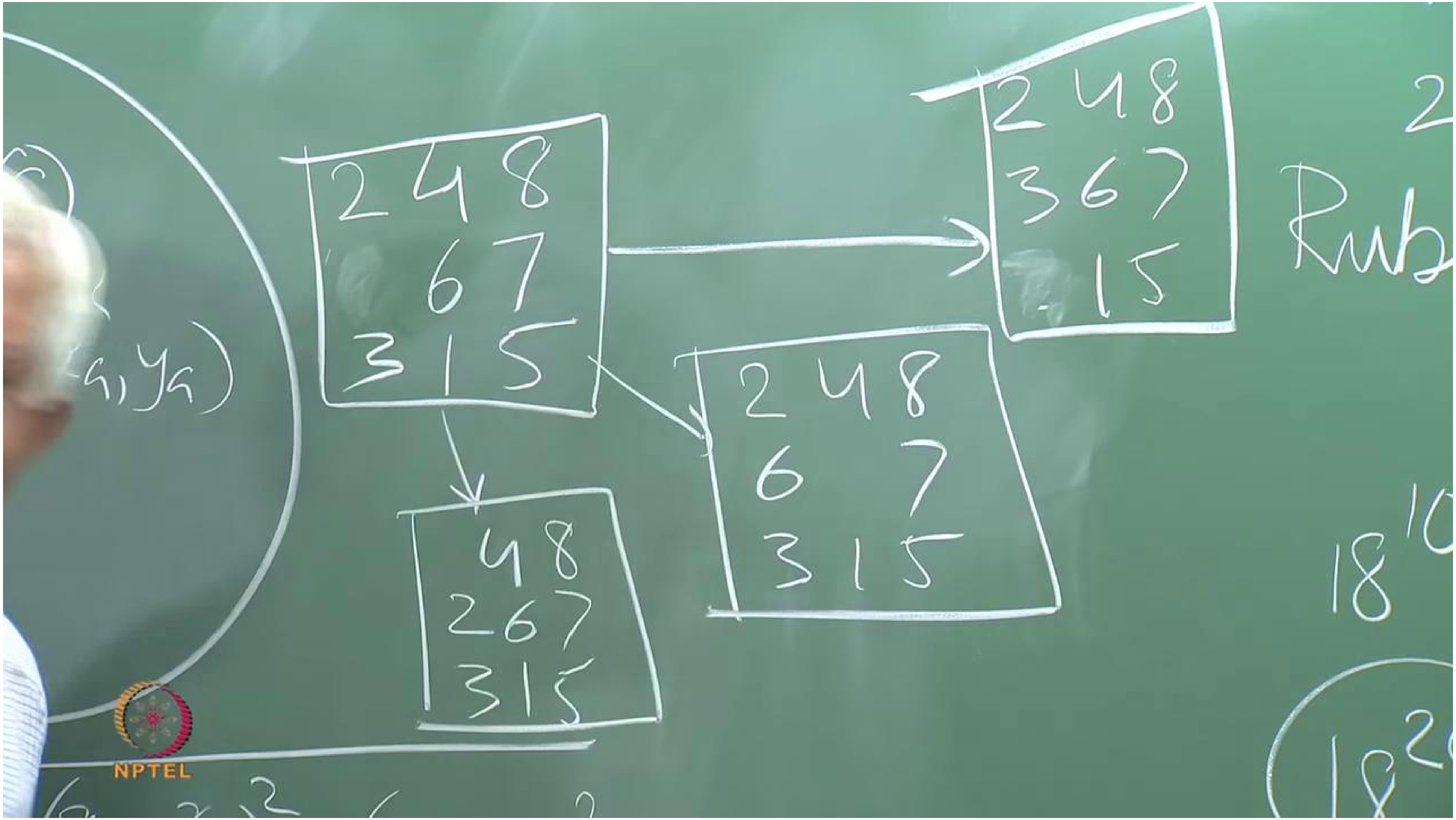
$= |x_s - x_g| + |y_s - y_g|$

Manhattan distance

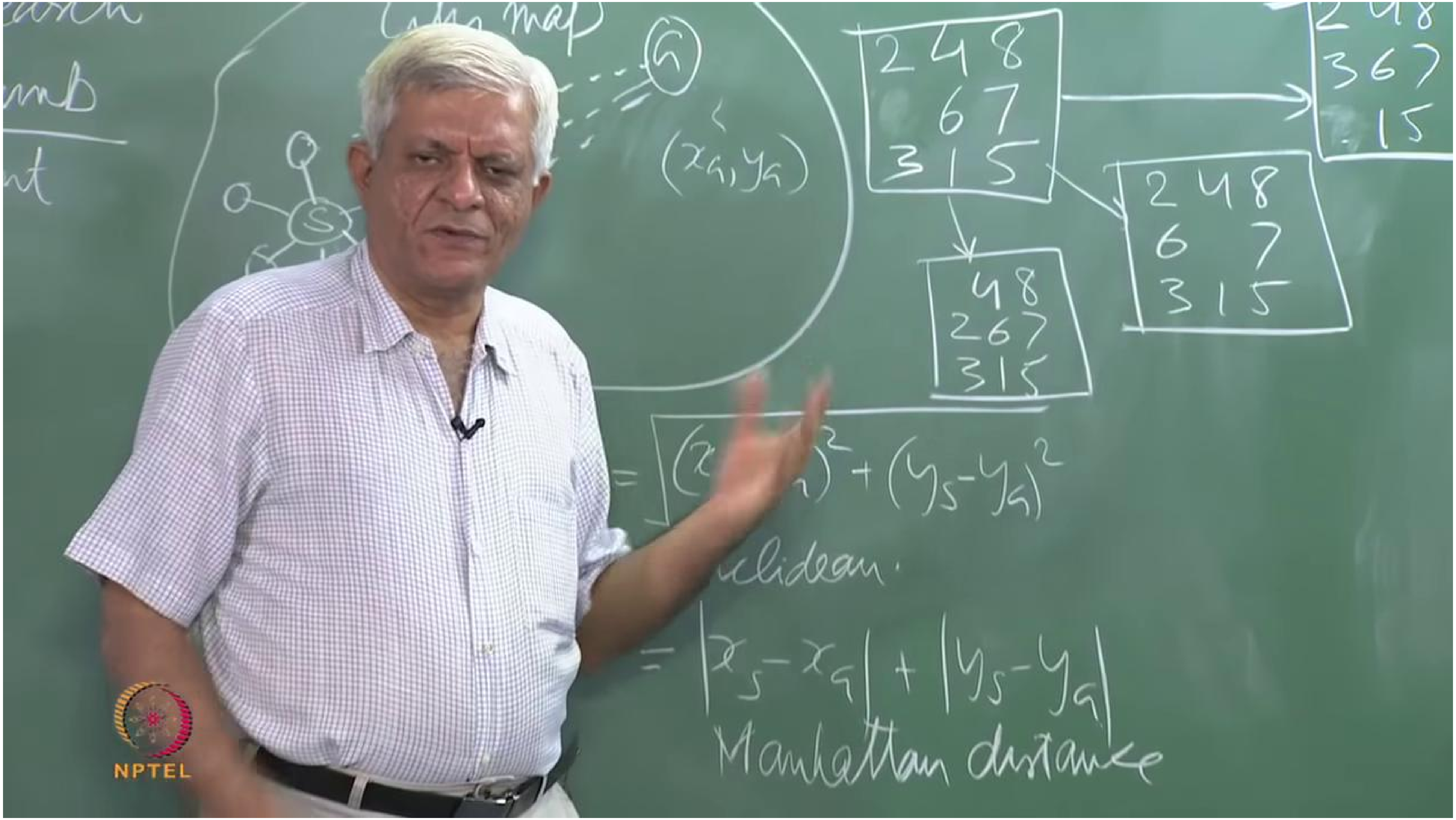


City map
2 4 8
3 6 7
1 5
24-puzzle
Rubik's cube
b = 18
18^10 ≈ 35
18^20 ≈ 12
40 Billion Centuries
NPTEL
distance

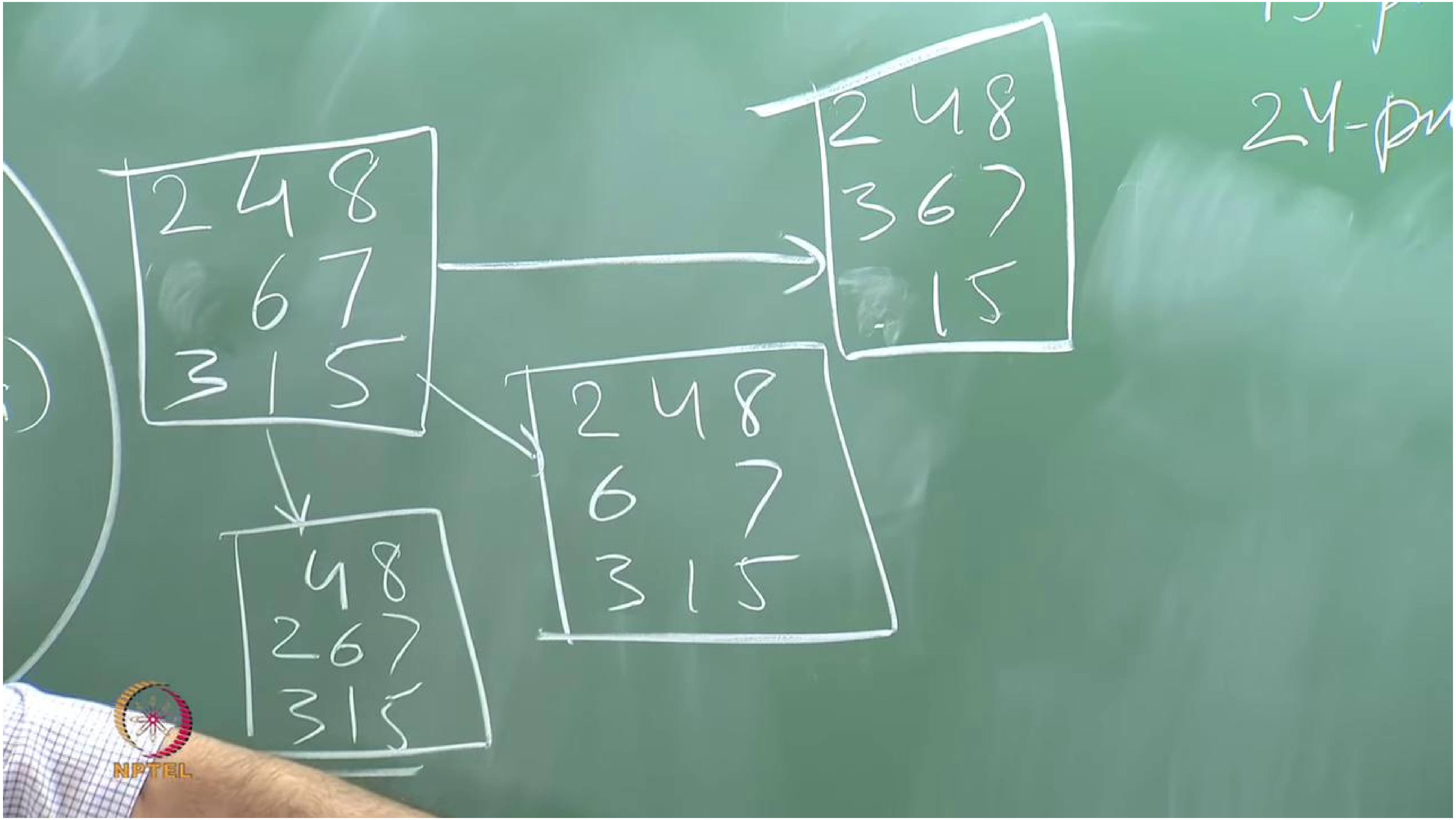

2 4 8
6 7
3 1 5
24-puzzle
Rubik's cube
b = 18
18^10 ≈ 35
18^20 ≈ 12
40 Billion Centuries
10^19
+ (y_5 - y
| y_5 - y_a |
distance
NPTEL



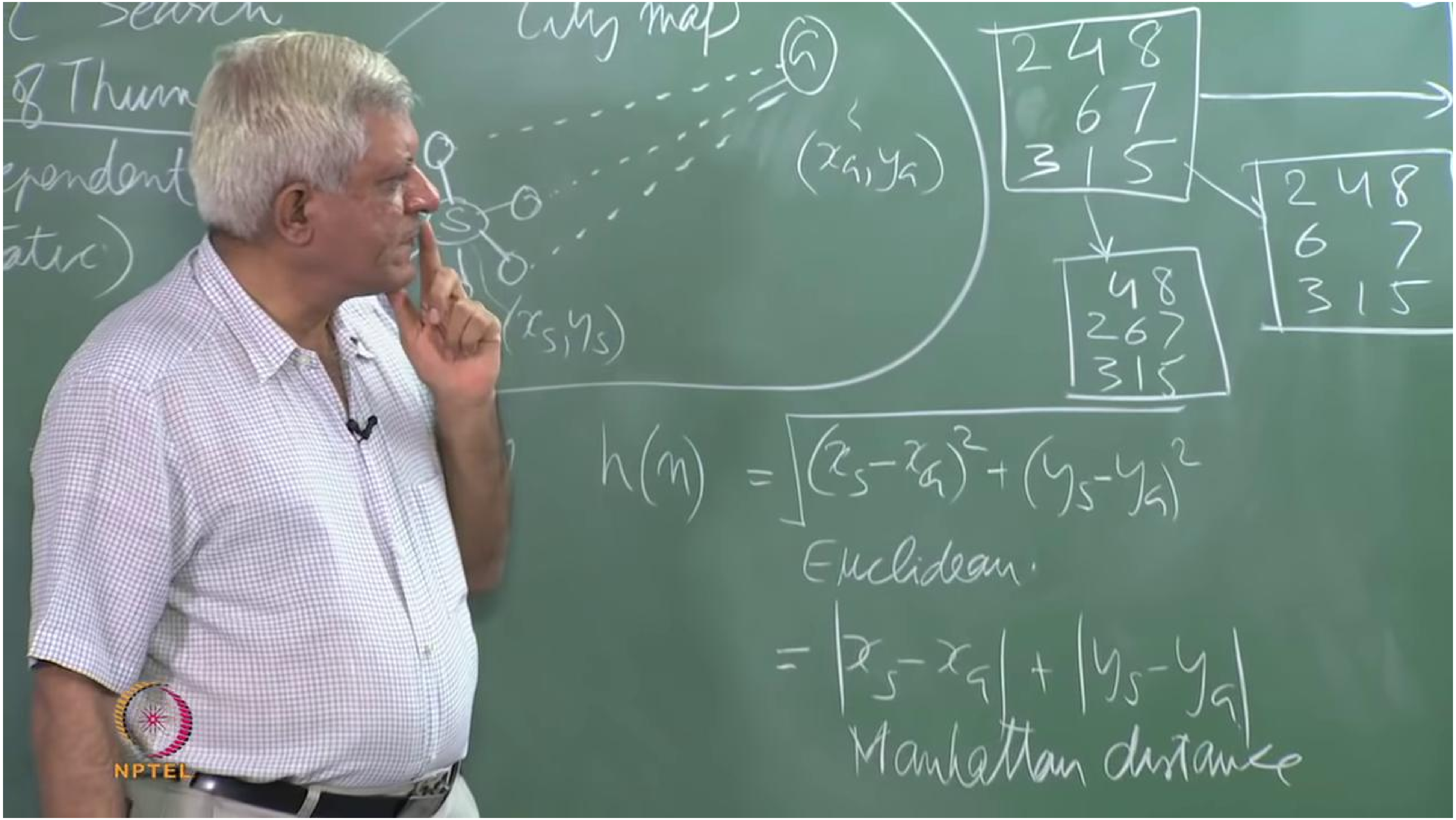
248
67
315
248
367
15
48
267
315
248
6 7
315
Rub
18
NPTEL

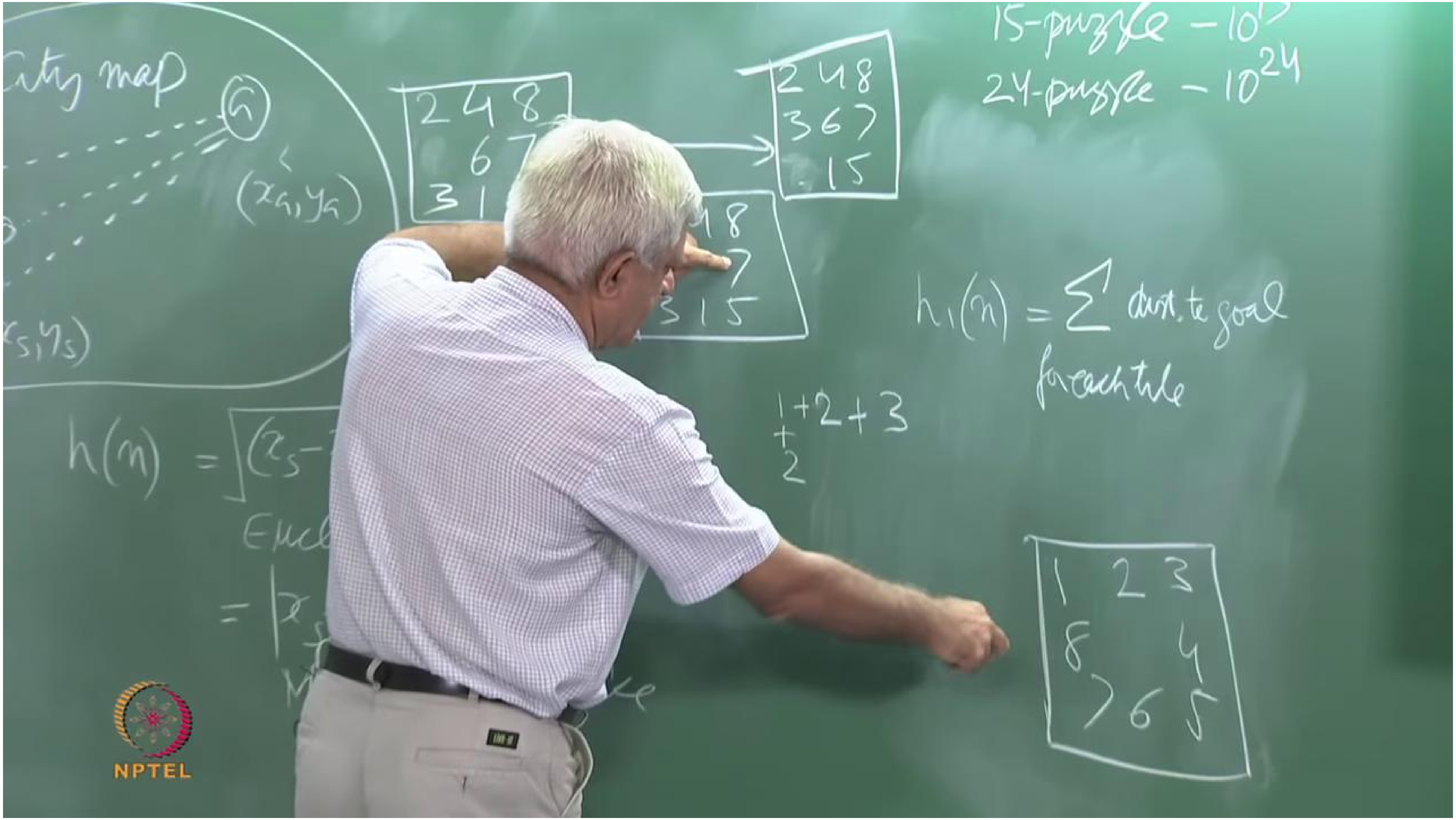


(x_g, y_g)
2 4 8
6 7
3 1 5
= |x_s − x_g| + |y_s − y_g|
Manhattan distance
NPTEL

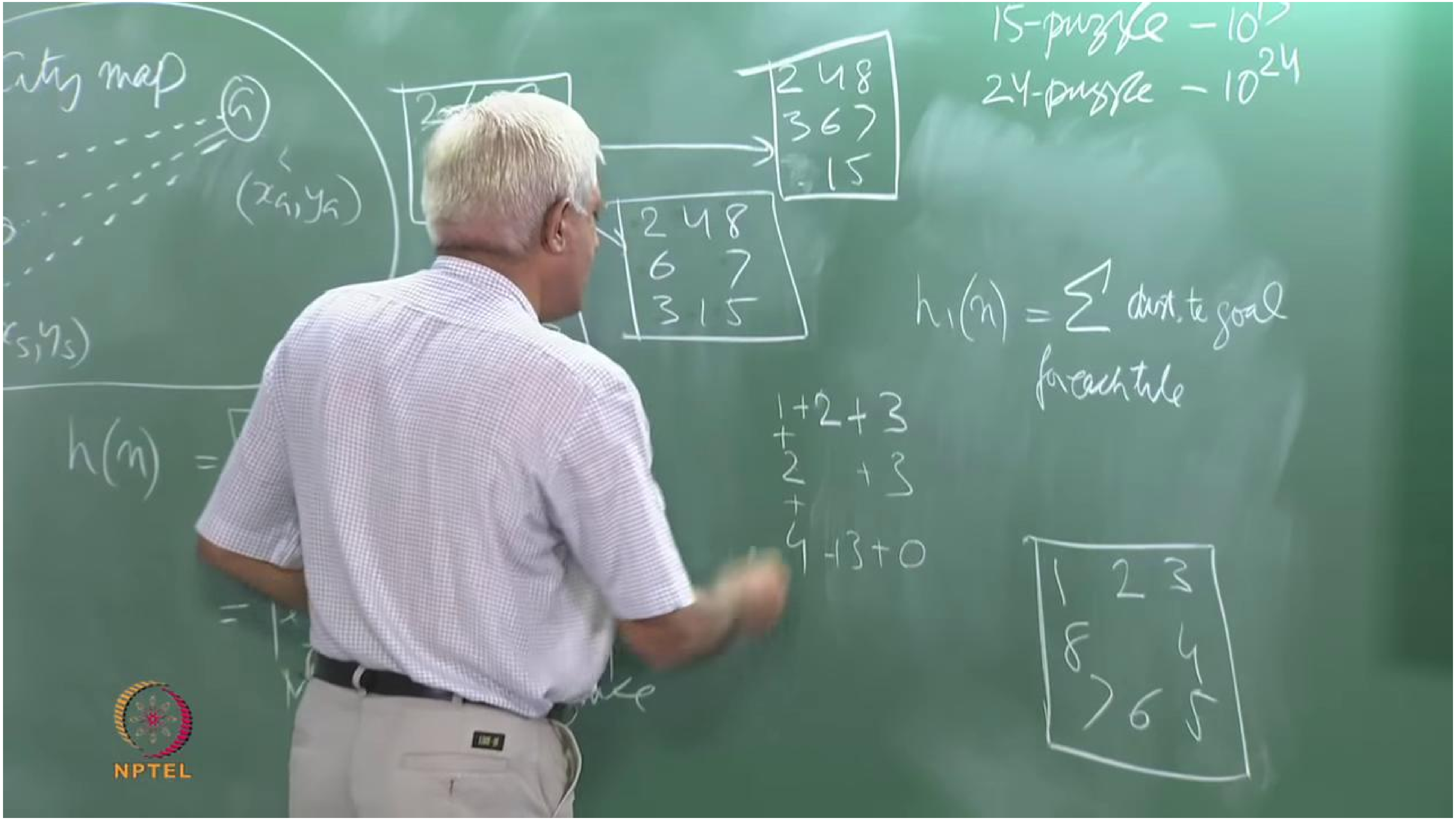


2 4 8
6 7
3 1 5
2 4 8
3 6 7
1 5
2 4 8
6 7
3 1 5
4 8
2 6 7
3 1 5

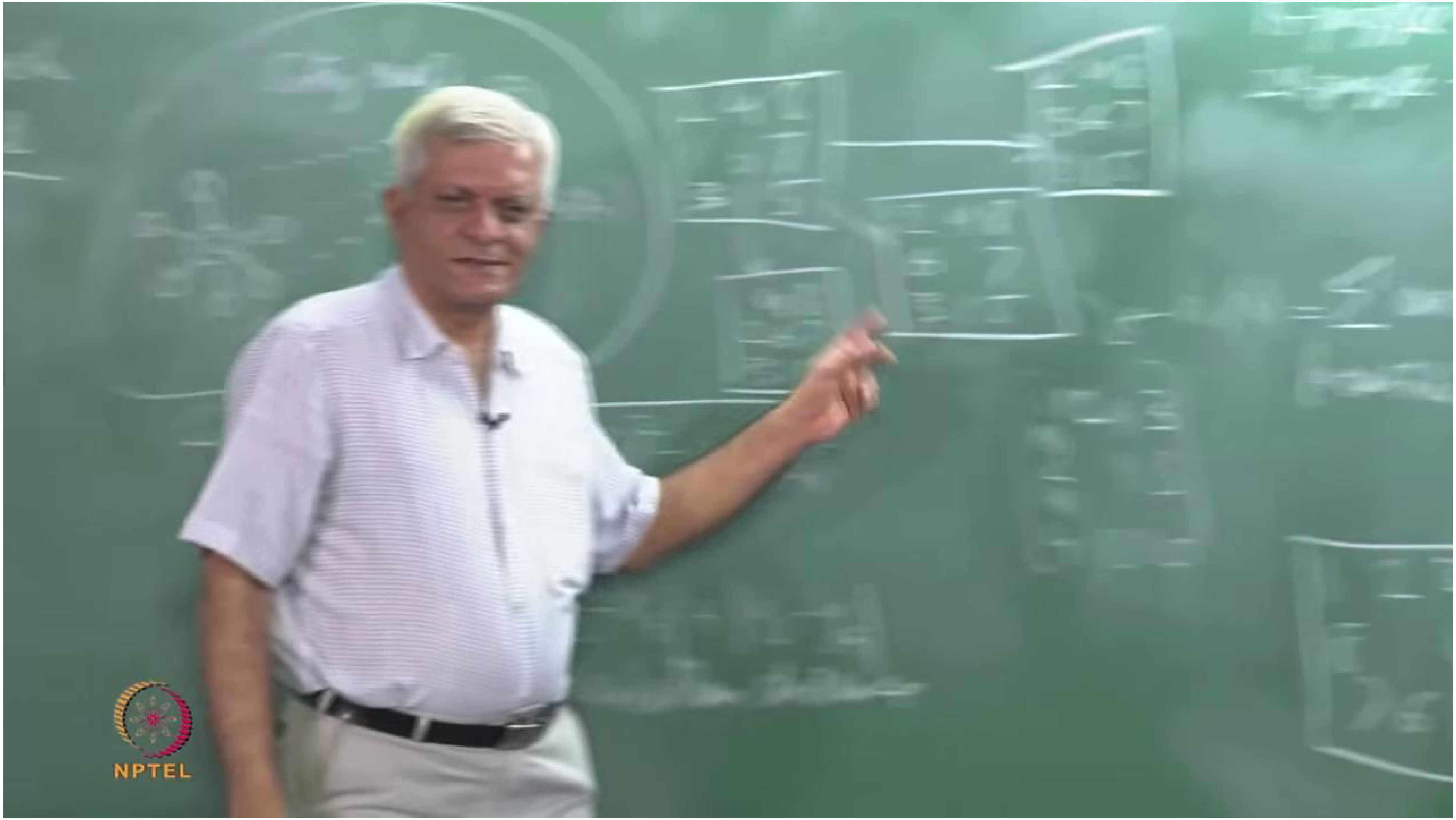

Search
of Thum
ependent
atic)
City map
G
(x_G, y_G)
S
(x_S, y_S)
2 4 8
6 7
3 1 5
4 8
2 6 7
3 1 5
2 4 8
6 7
3 1 5
h(n) = √((x_S − x_G)² + (y_S − y_G)²)
Euclidean
= |x_S − x_G| + |y_S − y_G|
Manhattan distance
NPTEL



City map
(x_G, y_G)
h(n) =
15-puzzle – 10
24-puzzle – 10^24
2 4 8
3 6 7
1 5
h_1(n) = Σ dist. to goal for each tile
1+2+3
+
2
1 2 3
8 4
7 6 5
NPTEL



City map
G
(x_G, y_G)
(x_S, y_S)
$h(n) =$
15-puzzle – 10^{13}
24-puzzle – 10^{24}
2 4 8
3 6 7
1 5
2 4 8
6 7
3 1 5
$h_1(n) = \sum$ dist. to goal for each tile
1 + 2 + 3
+ 2 + 3
+ 4 + 3 + 0
1 2 3
8 4
7 6 5
NPTEL


NPTEL

3 1 5 $\leftarrow h_1(n) = \Sigma$

for each

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & & 3 \\ 4 & 3 & 0 \end{bmatrix}$$

1 2

NPTEL

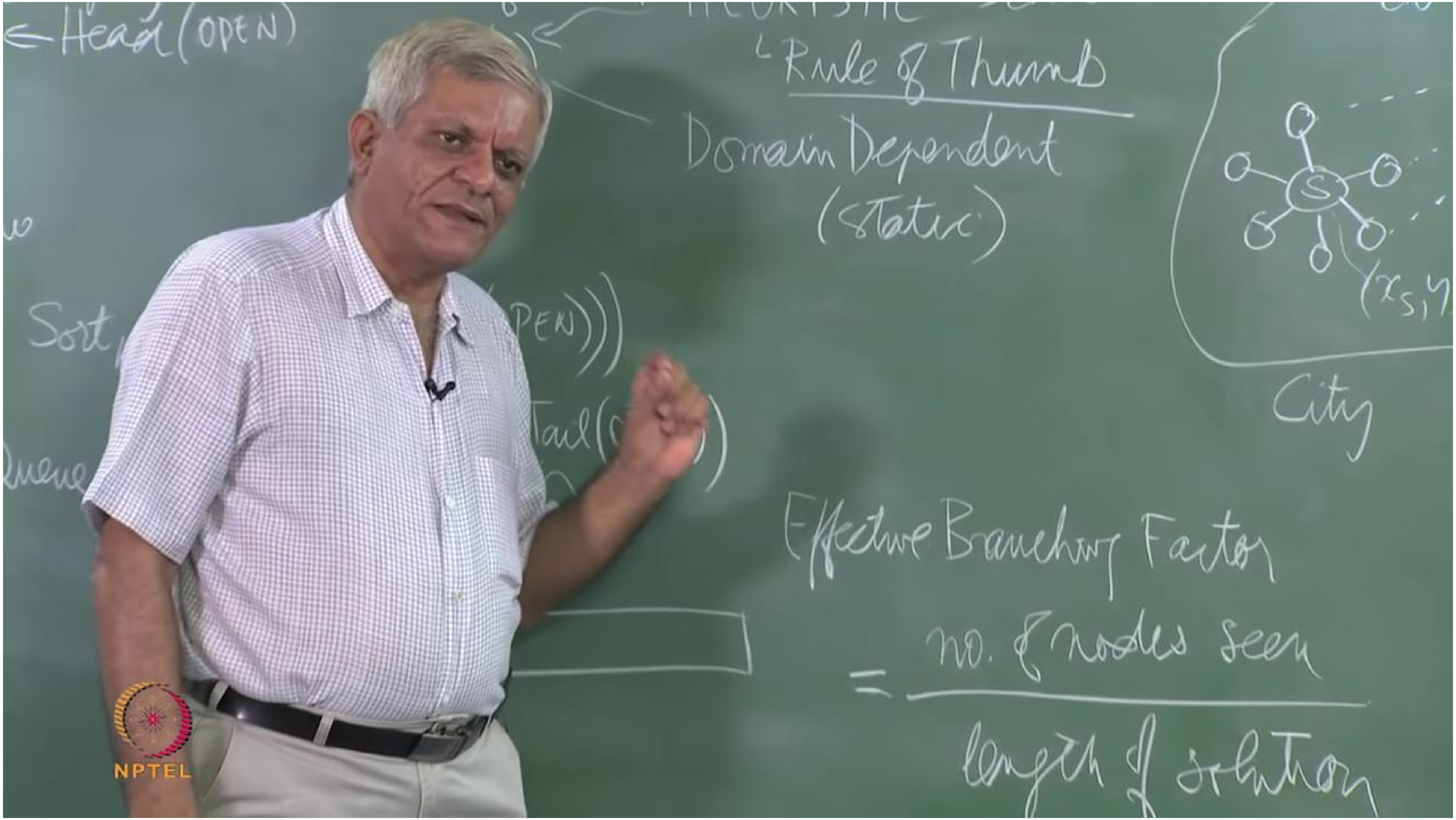
for each tile

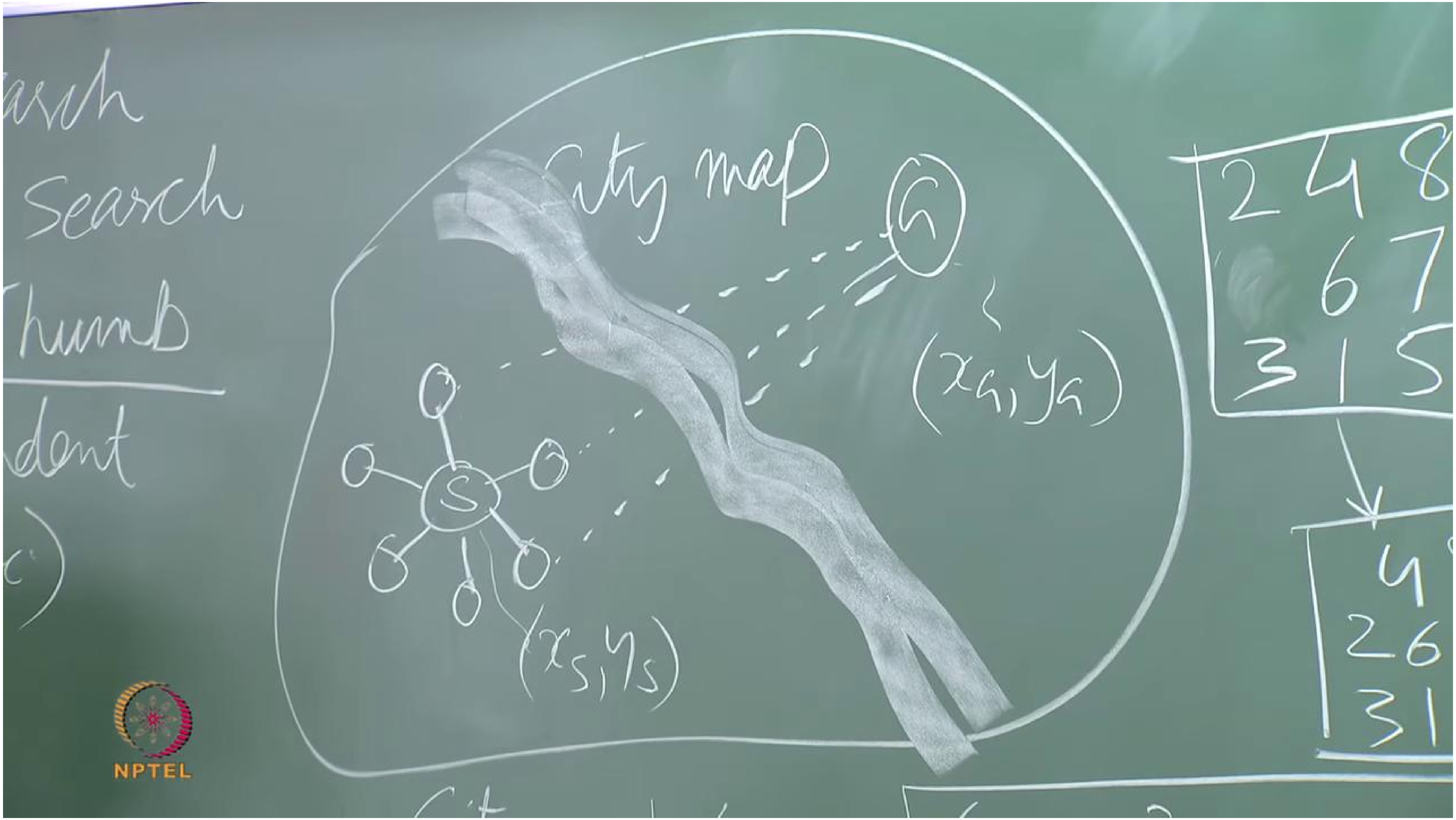
$h_2(n)$ = no. of misplaced tiles

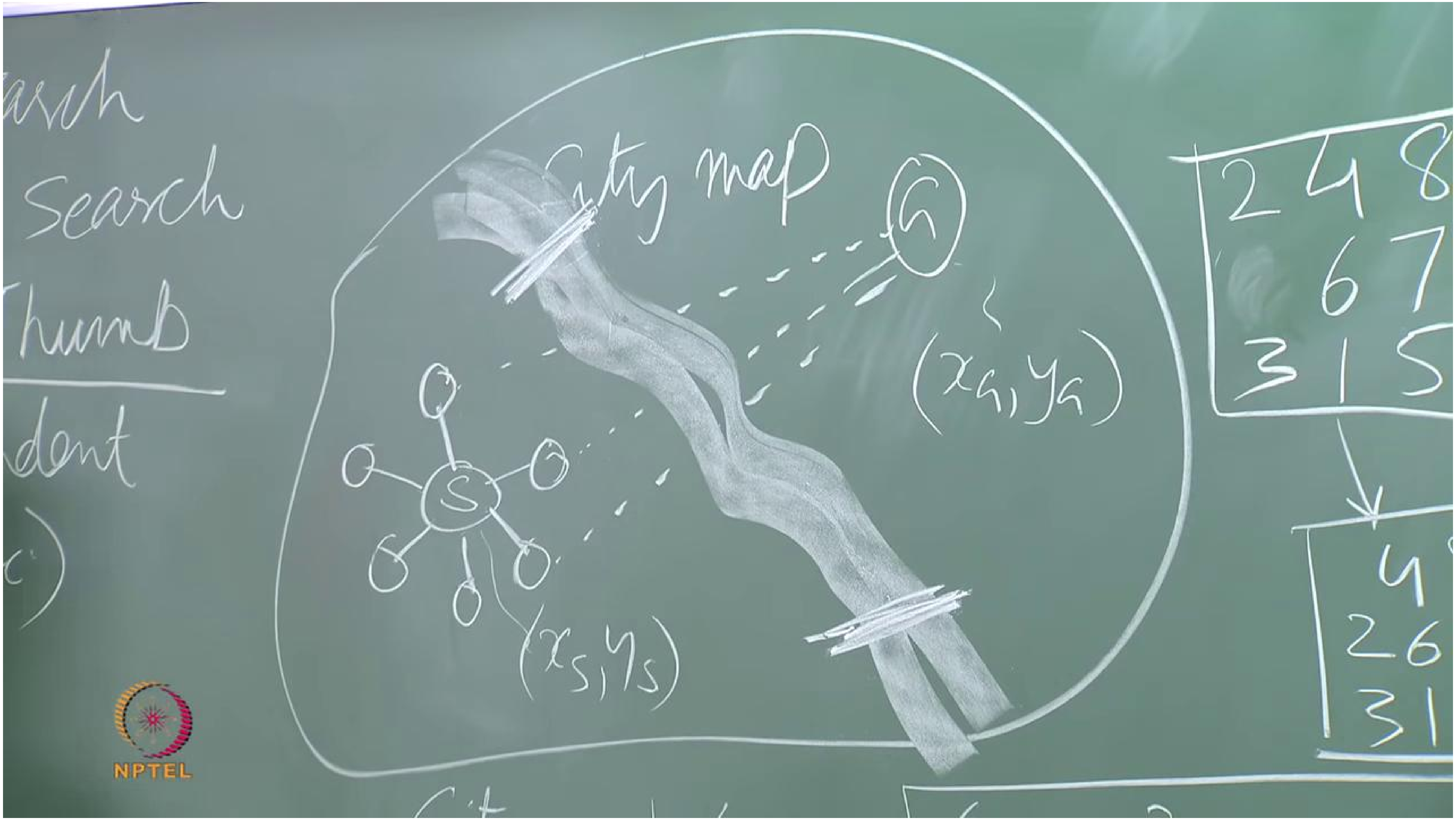
1	2	3

Effective Branching Factor

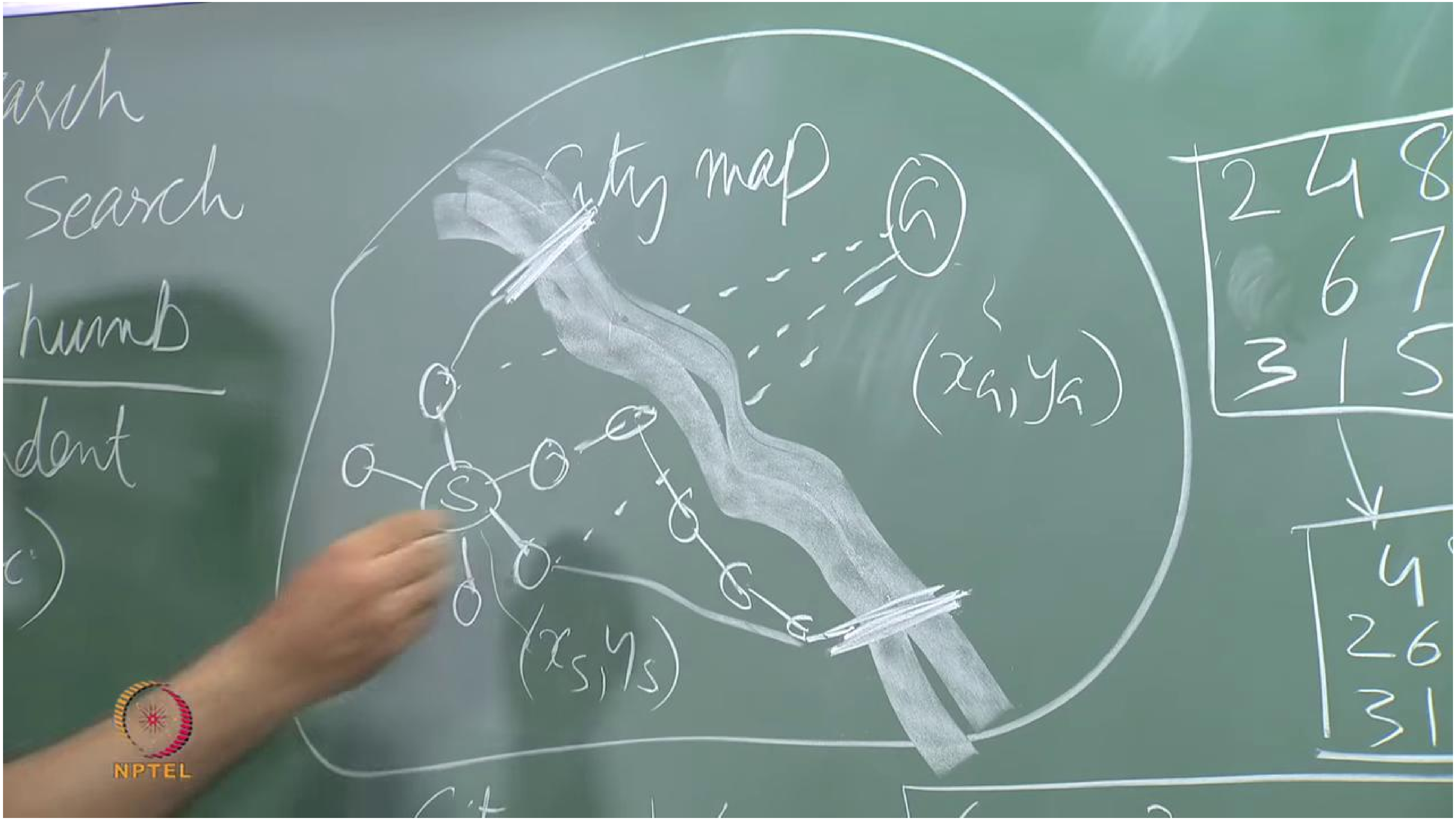
$$= \frac{\text{no. of nodes seen}}{\text{length of solution}}$$

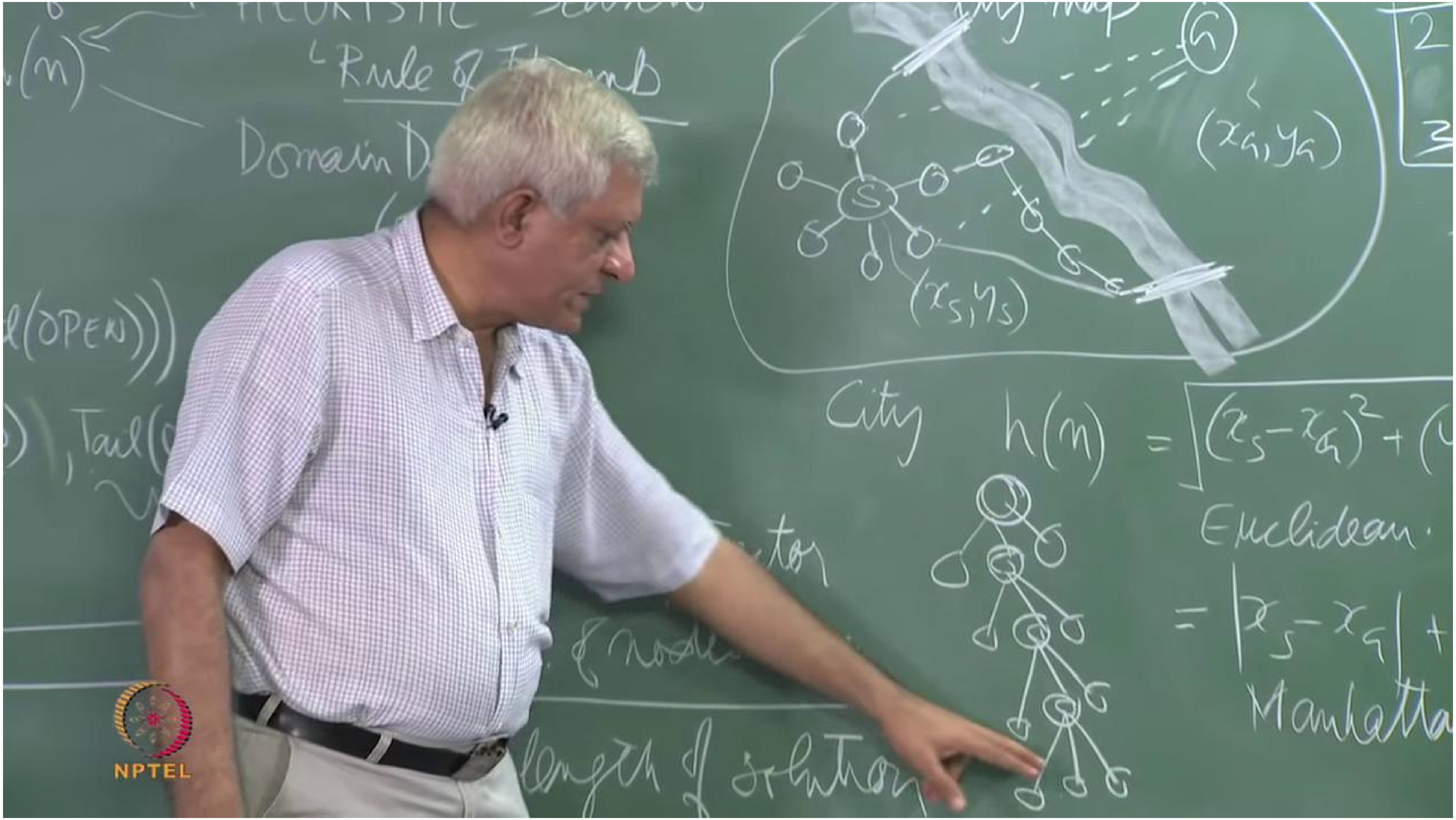

Head(OPEN)
Rule of Thumb
Domain Dependent
(static)
Sort
Queue
S
City
Effective Branching Factor
= no. of nodes seen / length of solution
NPTEL


Search
Thumb
City map
G
(x_G, y_G)
S
(x_S, y_S)
2 4 8
6 7
3 1 5
4
2 6
3 1
NPTEL



arch
Search
Thumb
dent
City map
G
(x_G, y_G)
S
(x_S, y_S)
2 4 8
6 7
3 1 5
4
2 6
3 1


arch
Search
Thumb
dent
City map
G
(xG, yG)
S
(xS, yS)
2 4 8
6 7
3 1 5
4
26
31
NPTEL



Domain D
(OPEN))
City
h(n)
(x_s, y_s)
(x_G, y_G)
S
G
Euclidean
Manhatta
length of solution
NPTEL

City $h(n) = \sqrt{(x_s - x_g)^2 + (y_s - y_g}$

Euclidean.

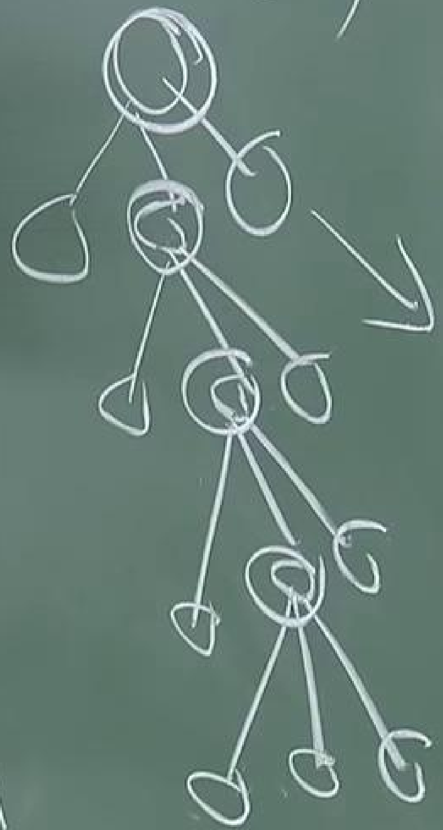
$= |x_s - x_g| + |y_s -$

Manhattan di

anching Factor

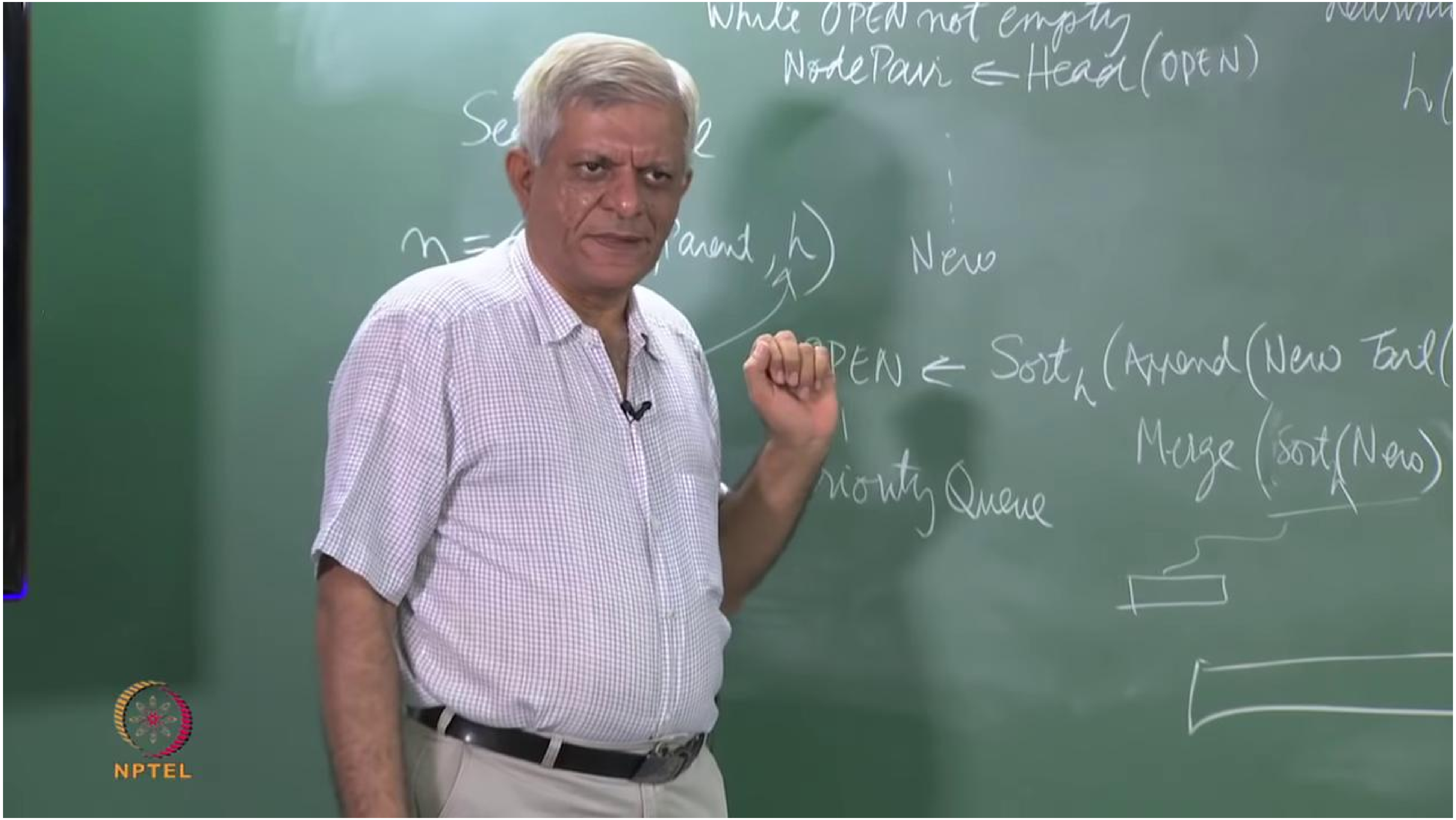
. of nodes seen

ength of solution



Best First Search

Completeness

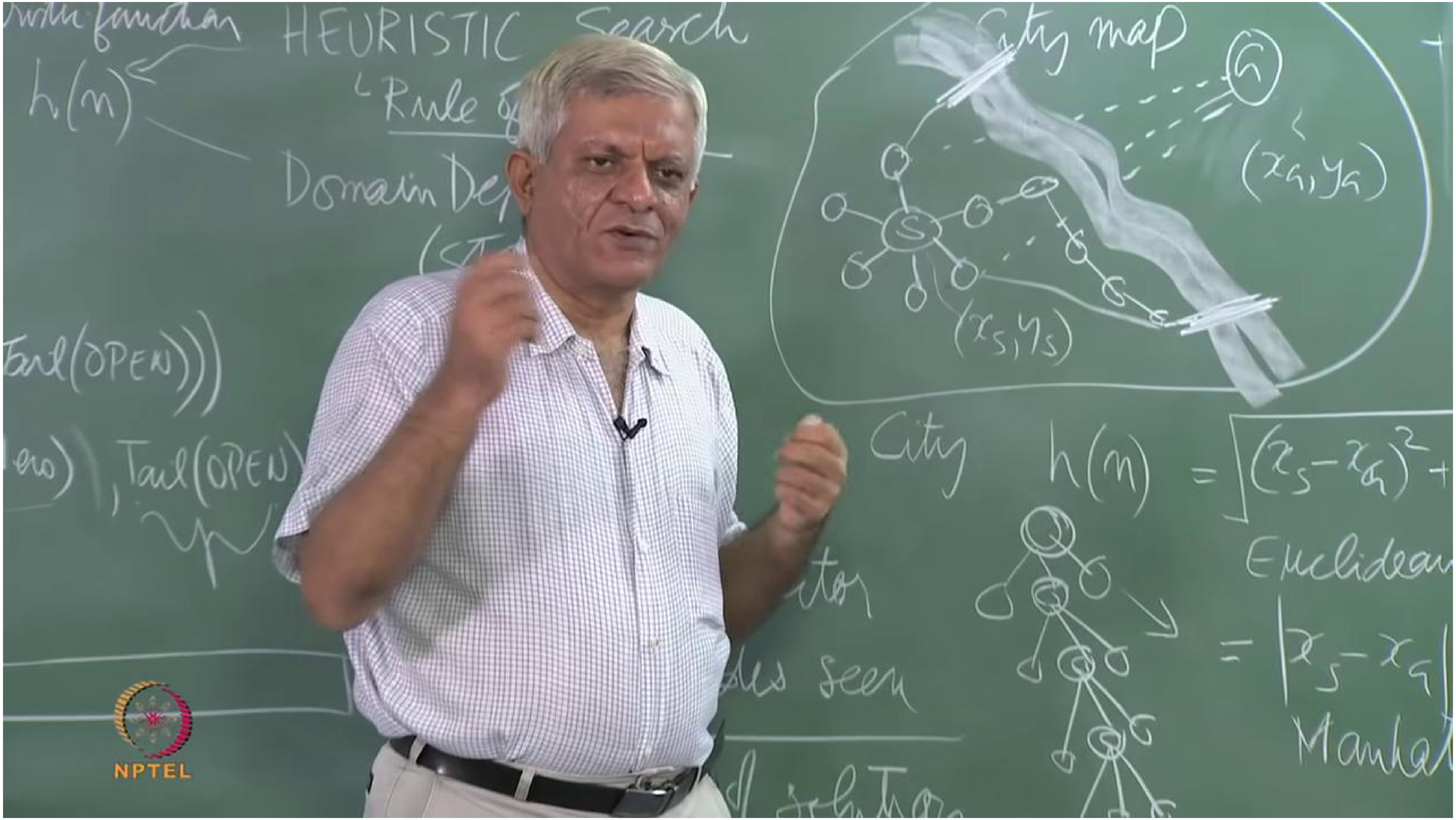


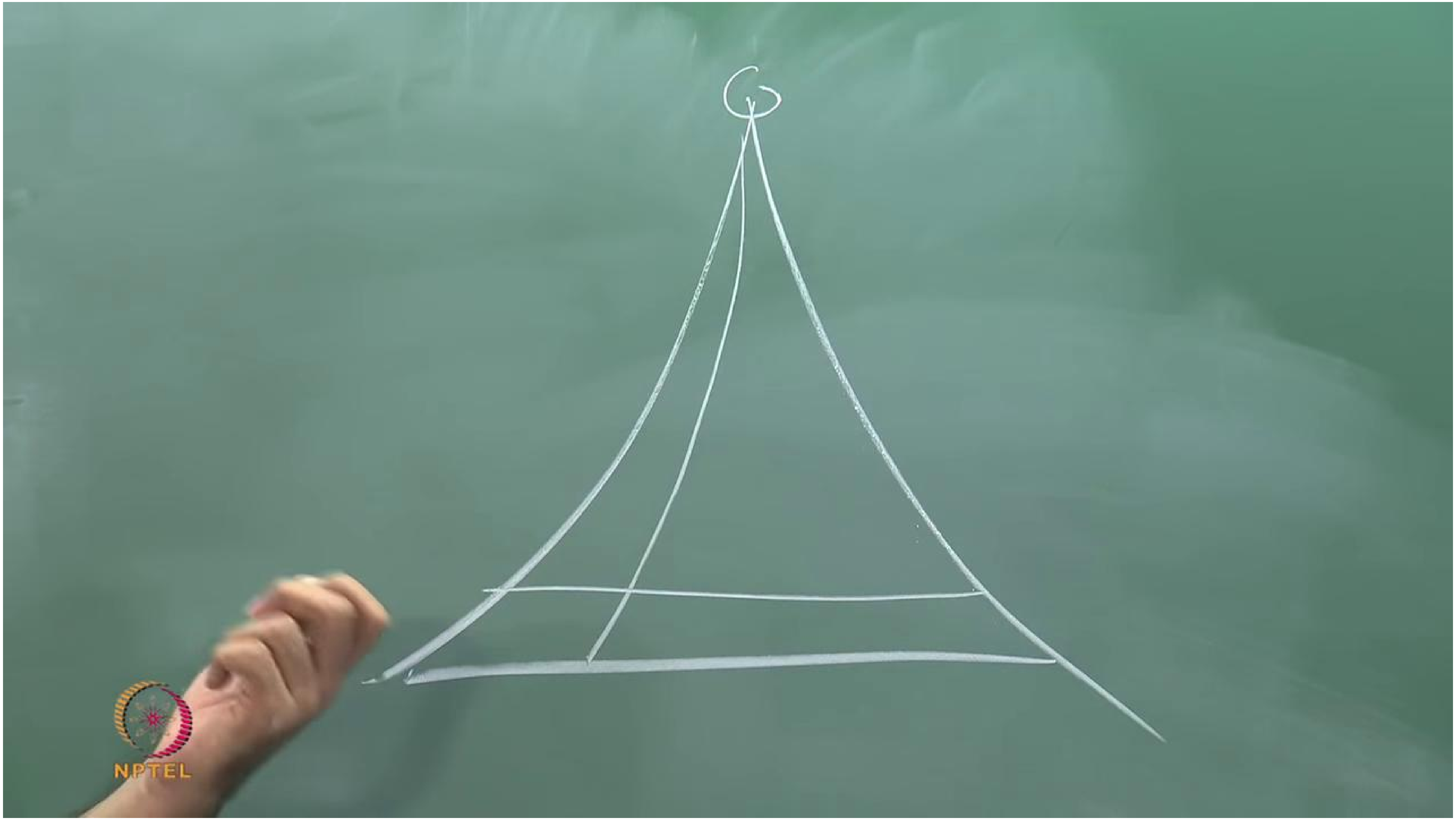
While OPEN not empty
NodePair ← Head(OPEN)
New
OPEN ← Sort(Append(New Tail(
Merge(Sort(New)
Queue
NPTEL

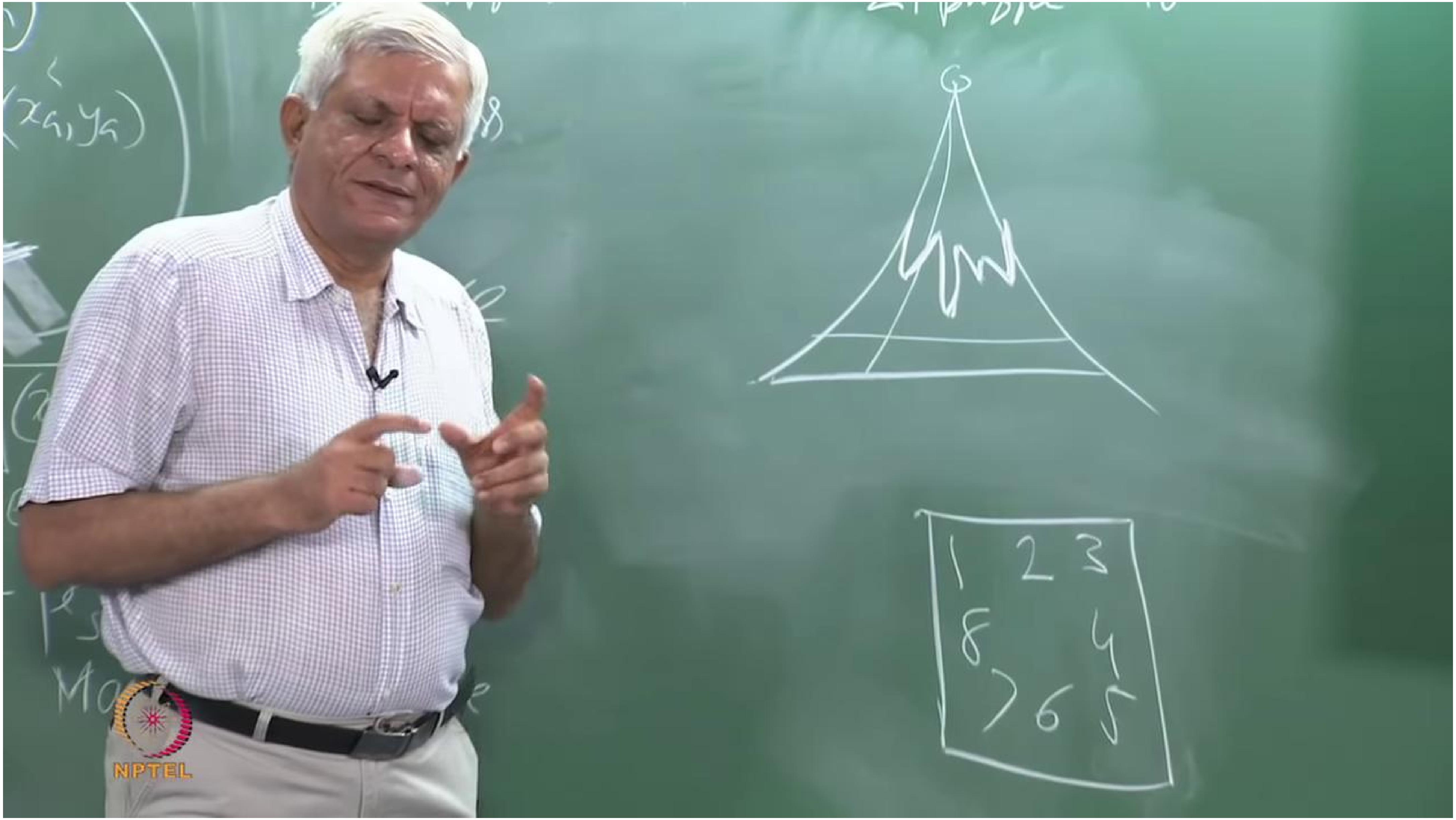
Best First Search

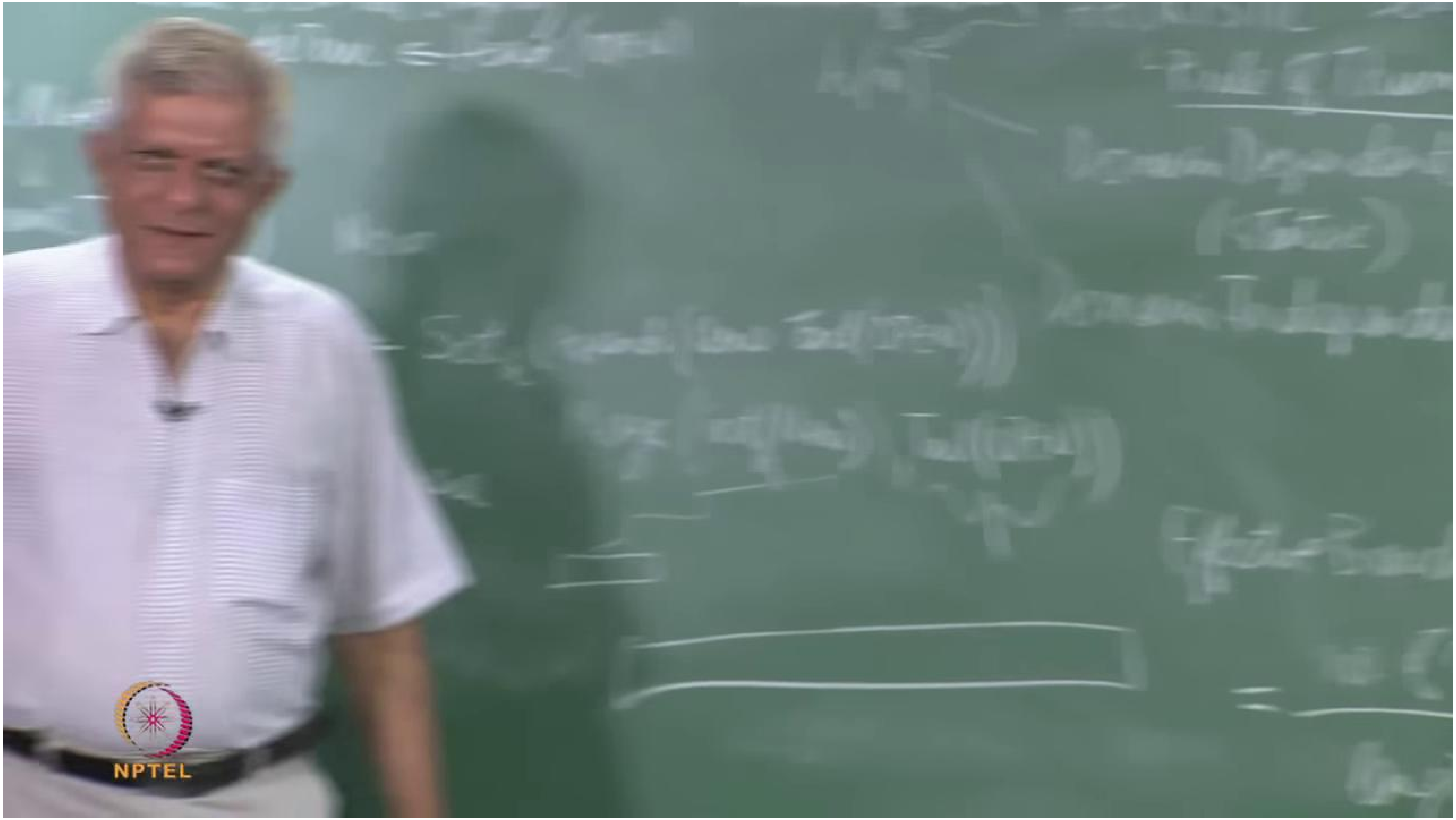
Completeness

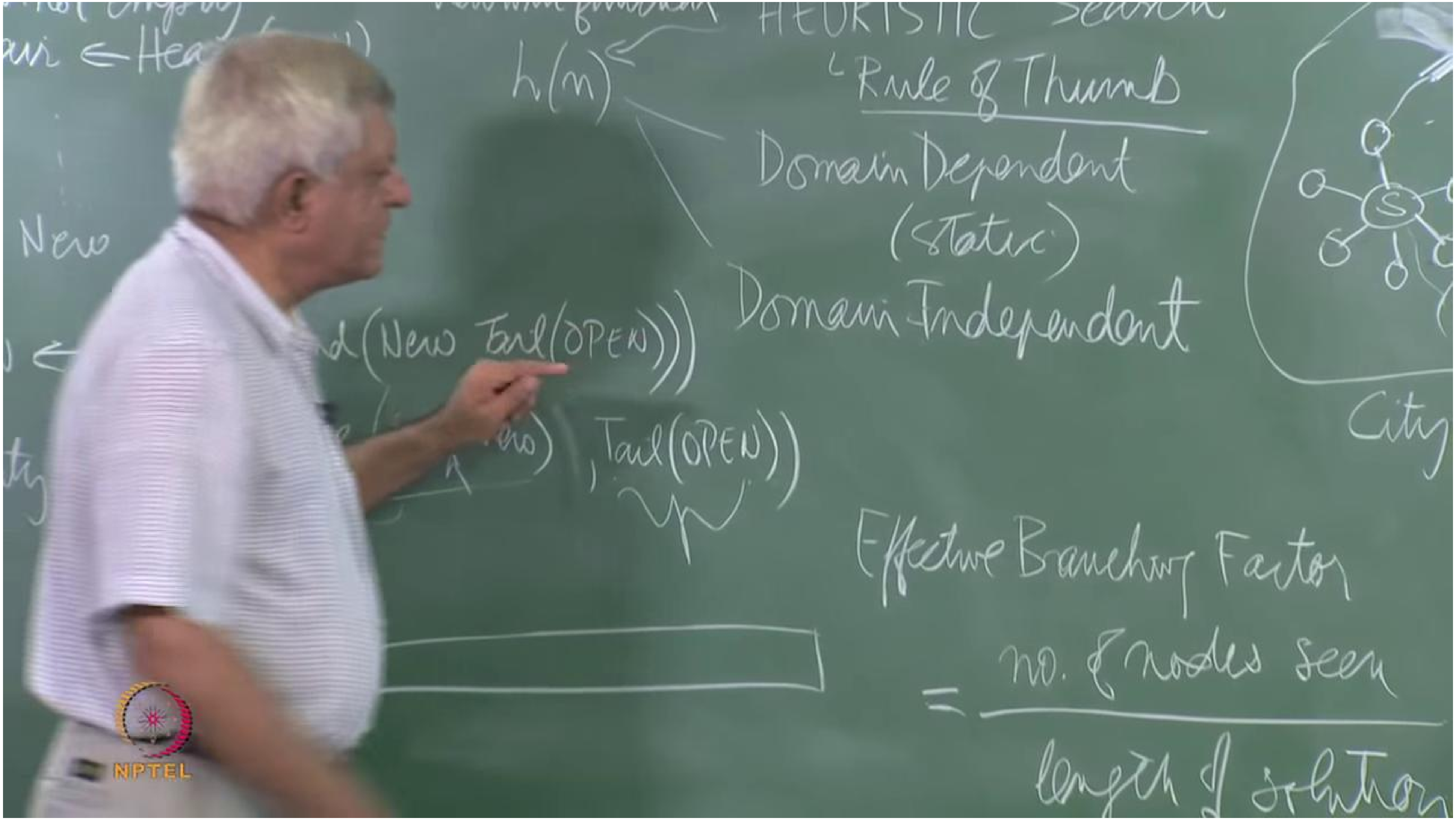
Time / Space

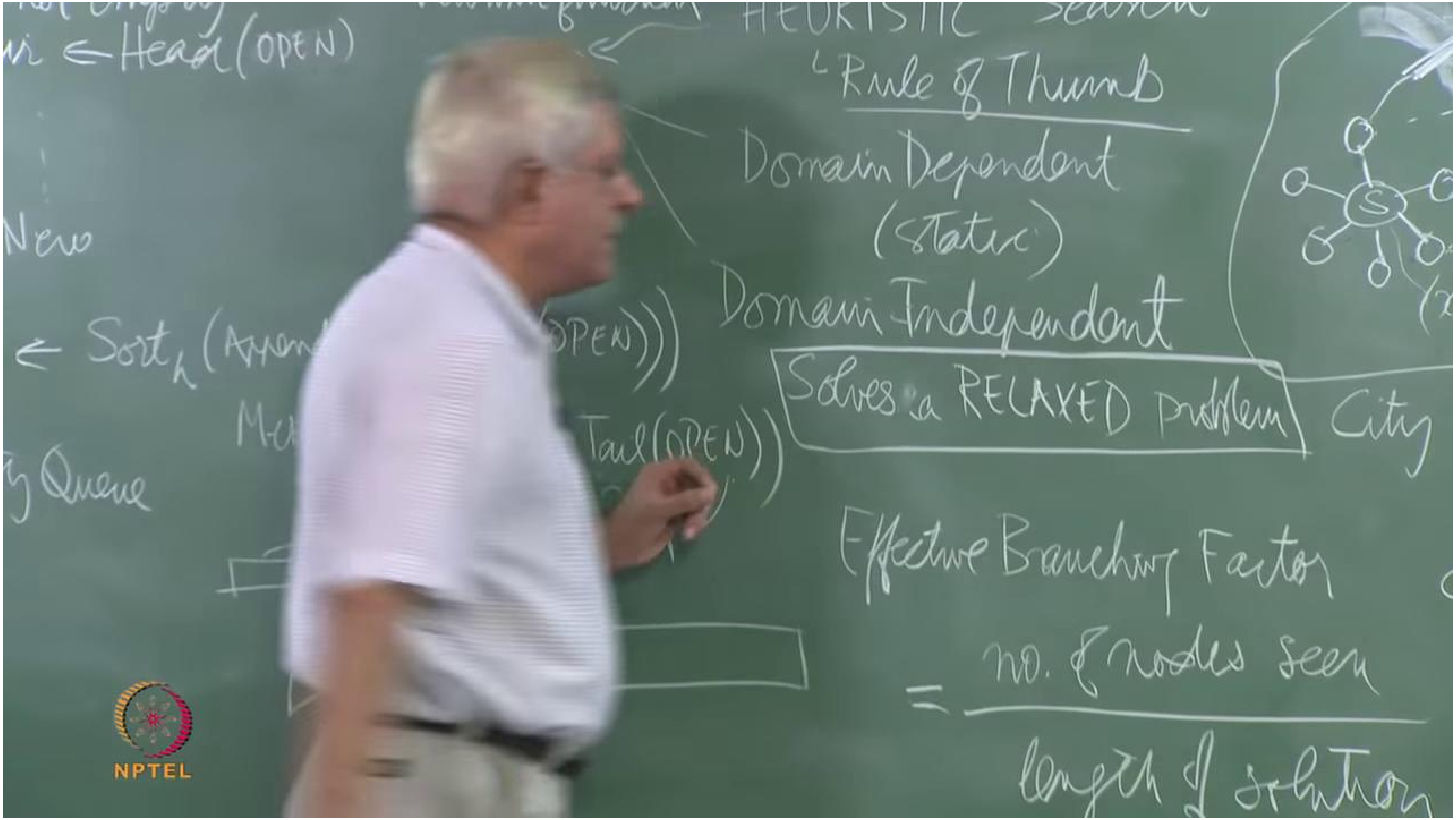

HEURISTIC Search
City map
h(n)
Domain Dep
Tail(OPEN)))
Tail(OPEN)
City
h(n)
Euclidean
Manhat
NPTEL


G
NPTEL

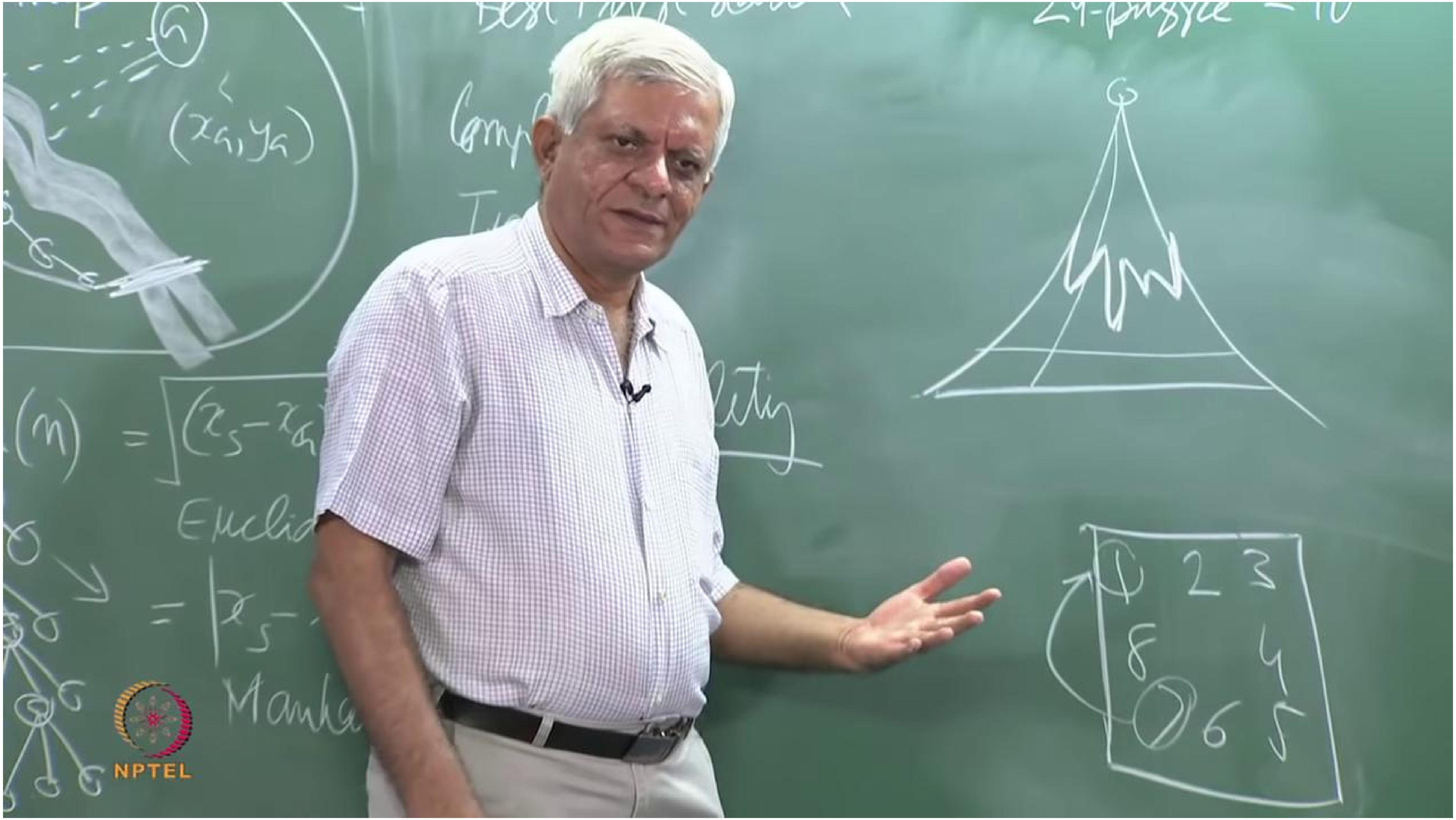

(xn, yn)
1 2 3
8 4
7 6 5
NPTEL

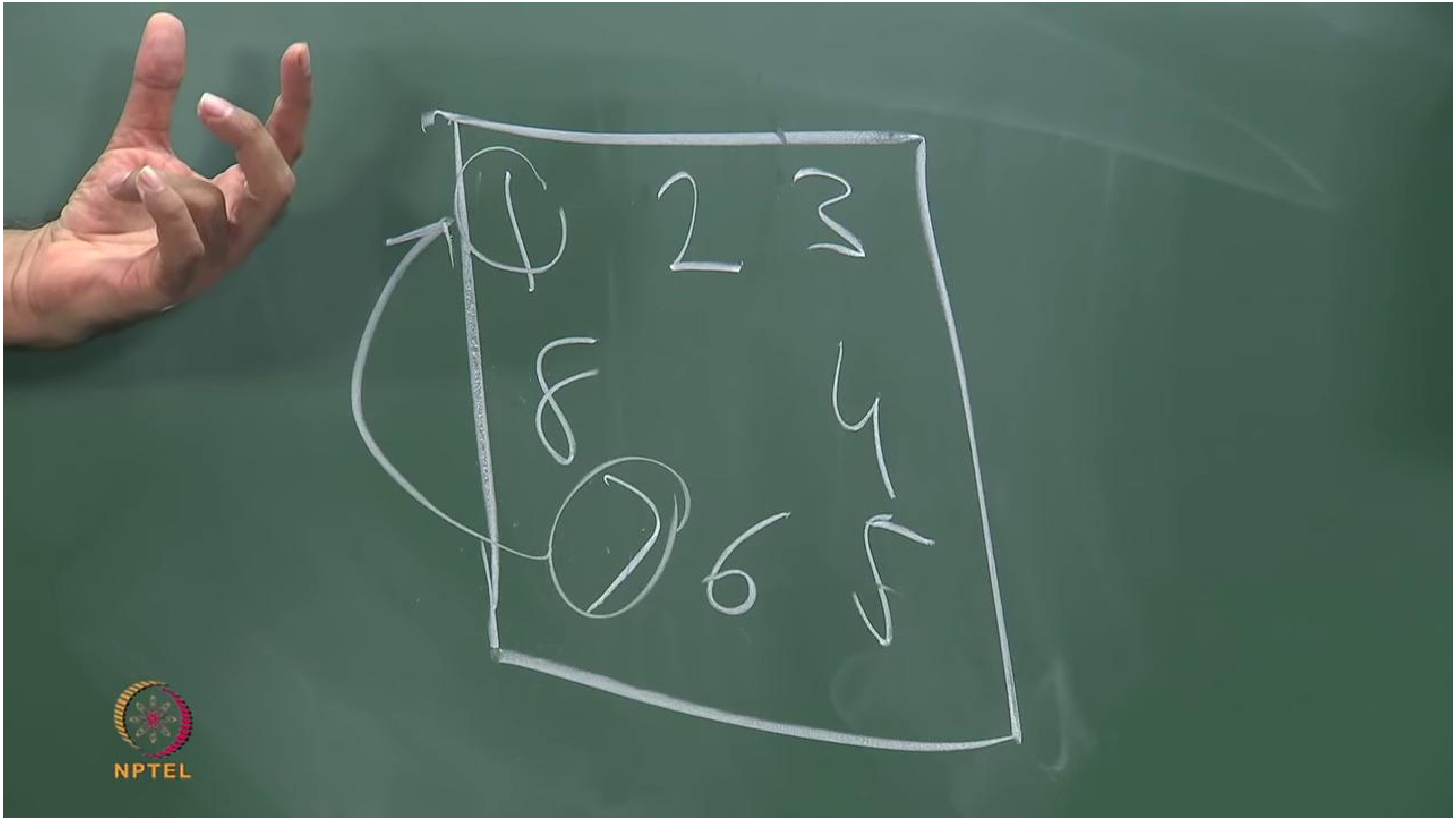


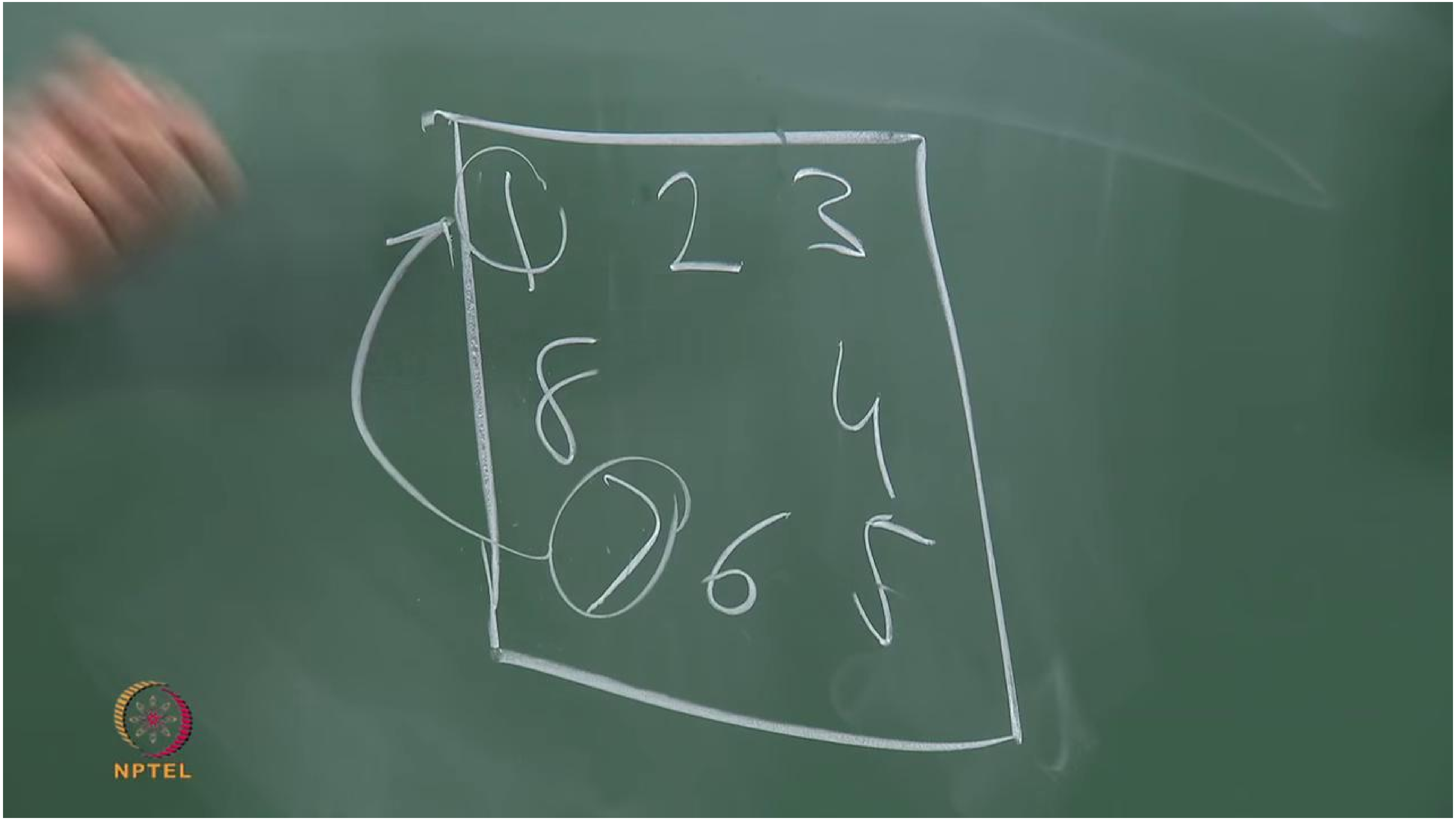

HEURISTIC Search
h(n)
Rule of Thumb
Domain Dependent
(static)
Domain Independent
New
(New Tail(OPEN)))
Tail(OPEN))
S
City
Effective Branching Factor
= no. of nodes seen / length of solution



HEURISTIC Search
Rule of Thumb
Domain Dependent
(static)
Domain Independent
Solves a RELAXED problem
City
Effective Branching Factor
= no. of nodes seen
length of solution
Head(OPEN)
New
Sort
Tail(OPEN))
Queue
S
NPTEL


(xa, ya)
Euclid
Manha
1 2 3
8 4
6 5
NPTEL


1 2 3
8 4
7 6 5
NPTEL



1 2 3
8 4
7 6 5
NPTEL