2D Array

Matrix

- When data is stored in 2 Dimensions ROW * Columns

▼ Spiral Matrix Problem:

- Approach : Cover the outer boundary then move to the inner boundary towards the center of the matrix .
- Moving in a single direction until a single cell is left
- After every successful rotation the cells to be covered next would logically be n/2 of the original i.e. ‘n’
- Find out : These values would change with every iteration
    - Starting Row
    - Ending Row
    - Starting Column
    - Ending Column
- Update Condition : After every iteration our Starting Row / Starting Column would get incremented where as our End Row / Ending Column would be decremented i.e.
    - StartRow ++
    - EndRow - -
    - StartCol ++
    - EndCol - -
- Top → Right → Bottom → Left
    - Repeat after every Iteration

- Diagonal sum :

  Finding the Sum of the elements present on the diognal of an 2D arrays

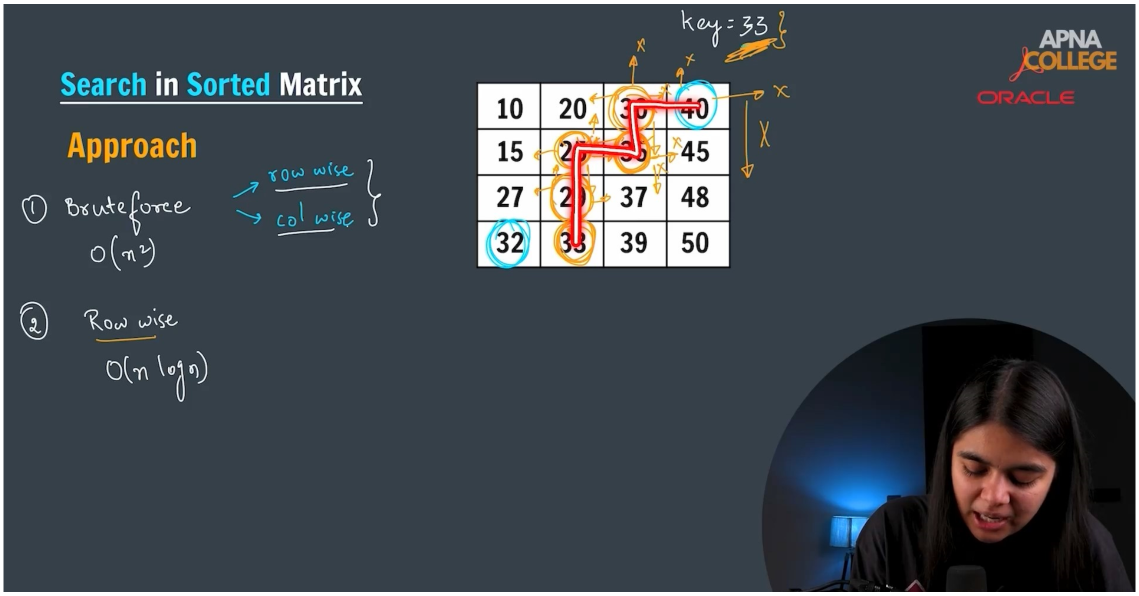
  - Only applicable of n=m matrix
    - Types of Diagonals
      - Primary
      - Secondary
  - Types of Diagonal Sum matrix
    - n = m = Even
      - No overlapping element is there
    - n = m = Odd
      - The element at the center is overlapping
  - Cell (Column , Row)
    - Column = Row for every diagonal cell
  - Brute Force Method

    Time Complexity : $O(n^2)$

  - Optimized Method

    Time Complexity : $O(n)$

- Search in a Sorted Matrix
  - Searching a key in a Sorted Matrix



- ▼ Bruteforce Approach
  - Row wise search using Binary search . In this case we were unable to use the sorted columns of the matrix
    - Time Complexity : $O(nlogn)$
- ▼ Optimized Approach :
  - In this approach we use both sorted components of the array and generate a pattern
  - We can notice that If the key is smaller than out current element we can go left
  - And if the key is comparatively larger we can go down until we have reached our key
    - Time Complexity : $O(n+m)$