EE4013 Assignment-1

Shaik Mastan Vali - EE18BTECH11039

Download all codes from

```
https://github.com/Mastan1301/EE4013/tree/main/assignment-1/codes
```

and latex-tikz codes from

```
https://github.com/Mastan1301/EE4013/tree/main/assignment-1
```

1 Problem

Consider the following matrix:

$$R = \begin{bmatrix} 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \\ 1 & 5 & 25 & 125 \end{bmatrix}$$

Calculate the absolute value of the Eigen values of R.

2 Solution

We use the following property -

$$\prod_{i=0}^{k-1} \lambda_i = det(R)$$

where λ_i are the Eigen values and $det(.)$ is the determinant operator.
To find the determinant, we use the following recurrence relation.

$$det(R, n) = \sum_{j=0}^{n-1} (-1)^j \times R[0][j] \times det(cof(R, 0, j), n-1)$$

where $cof(i, j)$ is the cofactor matrix of the position (i, j) in the matrix R.

The recursion, for our example, is given in the flow chart.

To understand the process, consider $C_1 = cof(R, 0, 0)$.

$$\boxed{det(R, 4)}$$

$$\downarrow$$

$$\boxed{\begin{gathered} R[0][0] * det(cof(R, 0, 0), 3) - \\ R[0][1] * det(cof(R, 0, 1), 3) + \\ R[0][2] * det(cof(R, 0, 2), 3) - \\ R[0][3] * det(cof(R, 0, 3), 3) \end{gathered}}$$

Fig. 0: The flow of recursion

$$\boxed{det(C_1, 3)}$$

$$\downarrow$$

$$\boxed{\begin{gathered} C_1[0][0] * det(cof(C_1, 0, 0), 2) - \\ C_1[0][1] * det(cof(C_1, 0, 1), 2) + \\ C_1[0][2] * det(cof(C_1, 0, 2), 2) \end{gathered}}$$

Fig. 0: A 3×3 sub-problem

Now, consider $C_2 = cof(C_1, 0, 0)$.

$$\boxed{det(C_2, 2)}$$

$$\downarrow$$

$$\boxed{\begin{gathered} C_2[0][0] * det(cof(C_2, 0, 0), 1) - \\ C_2[0][1] * det(cof(C_2, 0, 1), 1) \end{gathered}}$$

Fig. 0: A 2×2 sub-problem

The determinant of a 1×1 matrix is the value of the element itself. We use this property as the base case.

The algorithm is in the following file -

```
https://github.com/Mastan1301/EE4013/tree/main/
    assignment−1/codes/code.c
```

3 Complexity of the algorithm

Time Complexity: At each stage of recursion, there are $O(n)$ recursive calls. In each of these function calls, we compute the co-factor matrix in $O(n^2)$ time. So, the total time complexity is $O(n^3)$.