|  |  |
| --- | --- |
|  | DEPARTMENT OF ARTIFICIAL INTELLIGNECE & DATA SCIENCE |

|  |  |
| --- | --- |
| Subject: Analysis of Algorithm | Course Code: CSC402 |
| Semester: 4 | Course: AI & DS |
| Laboratory No: 205 | Name of Subject Teacher: Pramod Bhavarthe |
| Name of Student: Sahil Shaikh | Roll Id: VU2S2223012 |

|  |  |
| --- | --- |
| Title of Practical | Implement Strassen Algorithm. |

**Theory –**

* The Strassen algorithm is a method for matrix multiplication that is more efficient than the standard matrix multiplication algorithm for large matrices.
* The standard algorithm has a time complexity of O(n^3), where n is the number of rows and columns in the matrices being multiplied. The Strassen algorithm has a lower time complexity of O(n^log2(7)), which is approximately O(n^2.81).
* The Strassen algorithm works by recursively breaking down the matrices into smaller sub-matrices, until they can be multiplied using a formula that involves only additions and subtractions. This reduces the number of multiplication operations required and speeds up the computation.

**Program -**

a = [[1,2,3,4],[2,3,4,5],[3,4,5,6],[4,5,6,7]]

b = [[5,5,5,5],[6,6,6,6],[7,7,7,7],[8,8,8,8]]

def new\_m(p, q): # create a matrix filled with 0s

   matrix = [[0 for row in range(p)] for col in range(q)]

   return matrix

def split(matrix): # split matrix into quarters

   a = matrix

   b = matrix

   c = matrix

   d = matrix

   while(len(a) > len(matrix)/2):

      a = a[:len(a)//2]

      b = b[:len(b)//2]

      c = c[len(c)//2:]

      d = d[len(d)//2:]

   while(len(a[0]) > len(matrix[0])/2):

      for i in range(len(a[0])//2):

         a[i] = a[i][:len(a[i])//2]

         b[i] = b[i][len(b[i])//2:]

         c[i] = c[i][:len(c[i])//2]

         d[i] = d[i][len(d[i])//2:]

   return a,b,c,d

def add\_m(a, b):

   if type(a) == int:

      d = a + b

   else:

      d = []

      for i in range(len(a)):

         c = []

         for j in range(len(a[0])):

            c.append(a[i][j] + b[i][j])

         d.append(c)

   return d

def sub\_m(a, b):

   if type(a) == int:

      d = a - b

   else:

      d = []

      for i in range(len(a)):

         c = []

         for j in range(len(a[0])):

            c.append(a[i][j] - b[i][j])

         d.append(c)

   return d

def strassen(a, b, q):

   # base case: 1x1 matrix

   if q == 1:

      d = [[0]]

      d[0][0] = a[0][0] \* b[0][0]

      return d

   else:

      #split matrices into quarters

      a11, a12, a21, a22 = split(a)

      b11, b12, b21, b22 = split(b)

      # p1 = (a11+a22) \* (b11+b22)

      p1 = strassen(add\_m(a11,a22), add\_m(b11,b22), q/2)

      # p2 = (a21+a22) \* b11

      p2 = strassen(add\_m(a21,a22), b11, q/2)

      # p3 = a11 \* (b12-b22)

      p3 = strassen(a11, sub\_m(b12,b22), q/2)

      # p4 = a22 \* (b21-b11)

      p4 = strassen(a22, sub\_m(b21,b11), q/2)

      # p5 = (a11+a12) \* b22

      p5 = strassen(add\_m(a11,a12), b22, q/2)

      # p6 = (a21-a11) \* (b11+b12)

      p6 = strassen(sub\_m(a21,a11), add\_m(b11,b12), q/2)

      # p7 = (a12-a22) \* (b21+b22)

      p7 = strassen(sub\_m(a12,a22), add\_m(b21,b22), q/2)

      # c11 = p1 + p4 - p5 + p7

      c11 = add\_m(sub\_m(add\_m(p1, p4), p5), p7)

      # c12 = p3 + p5

      c12 = add\_m(p3, p5)

      # c21 = p2 + p4

      c21 = add\_m(p2, p4)

      # c22 = p1 + p3 - p2 + p6

      c22 = add\_m(sub\_m(add\_m(p1, p3), p2), p6)

      c = new\_m(len(c11)\*2,len(c11)\*2)

      for i in range(len(c11)):

         for j in range(len(c11)):

            c[i][j] = c11[i][j]

            c[i][j+len(c11)] = c12[i][j]

            c[i+len(c11)][j] = c21[i][j]

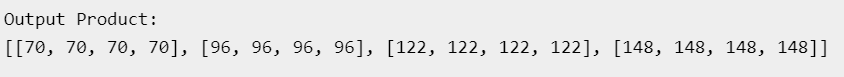
            c[i+len(c11)][j+len(c11)] = c22[i][j]

      return c

print("Output Product:")

print(strassen(a, b, 4))

**Output –**

****

**Time Complexity of Strassen’s Method**

Addition and Subtraction of two matrices takes O(N2) time.

**T(N) = 7T(N/2) + O(N2)**

From [Master's Theorem](https://www.geeksforgeeks.org/analysis-algorithm-set-4-master-method-solving-recurrences/), time complexity of above method is

**O(NLog7) which is approximately O(N2.8074)**

**Conclusion –**

**Therefore, we have successfully understood and Implemented Strassen Algorithm.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Grade and Dated Signature of Teacher** | **Total (10)** | **Remark** | **Dated signature of teacher** |
|  |  |  |