|  |  |
| --- | --- |
| **NAME:** | Pranav Sadanand Kore |
| **UID:** | 2021300065 |
| **SUBJECT** | Design and Analysis of Algorithm |
| **EXPERIMENT NO :** | 3 |
| **AIM:** | Strassen’s Matrix Multiplication |
| **ALGORITHM:** | 1. Start.  2. Declare two matrices A and B and take the values from the user.  3. Find S1 to S10 using provided formulae.  4. Find P1 to P7 using provided formulae.  5. Find the elements of matrix C which is the multiplication of A and B.  6. Print the result. |
| **PROGRAM:** | #include<stdio.h>  int main()  {  int a[2][2], b[2][2], c[2][2], i, j;  int m1, m2, m3, m4, m5, m6, m7;  printf("Enter the 4 elements of first matrix:\n");  for(i = 0; i < 2; i++)  for(j = 0; j < 2; j++)  scanf("%d", &a[i][j]);  printf("Enter the 4 elements of second matrix:\n");  for(i = 0; i < 2; i++)  for(j = 0; j < 2; j++)  scanf("%d", &b[i][j]);  printf("\nThe first matrix is\n");  for(i = 0; i < 2; i++)  {  printf("\n");  for(j = 0; j < 2; j++)  printf("%d\t", a[i][j]);  }  printf("\nThe second matrix is\n");  for(i = 0; i < 2; i++)  {  printf("\n");  for(j = 0; j < 2; j++)  printf("%d\t", b[i][j]);  }  m1= (a[0][0] + a[1][1]) \* (b[0][0] + b[1][1]);  m2= (a[1][0] + a[1][1]) \* b[0][0];  m3= a[0][0] \* (b[0][1] - b[1][1]);  m4= a[1][1] \* (b[1][0] - b[0][0]);  m5= (a[0][0] + a[0][1]) \* b[1][1];  m6= (a[1][0] - a[0][0]) \* (b[0][0]+b[0][1]);  m7= (a[0][1] - a[1][1]) \* (b[1][0]+b[1][1]);  c[0][0] = m1 + m4- m5 + m7;  c[0][1] = m3 + m5;  c[1][0] = m2 + m4;  c[1][1] = m1 - m2 + m3 + m6;  printf("\nAfter multiplication using Strassen's algorithm \n");  for(i = 0; i < 2 ; i++)  {  printf("\n");  for(j = 0; j < 2; j++)  printf("%d\t", c[i][j]);  }  return 0;  } |
| **RESULT ( SNAPSHOT)** | |
| **CONCLUSION:** | By doing this experiment I have learned to implement Strassen’s method to find the product of two matrices. As it has less no. of multiplications than normal method it’s time complexity is less i.e this method is efficient than normal. |