|  |  |
| --- | --- |
| **NAME :** | **Ms. Neerja Doshi** |
| **UID :** | **2021300029** |
| **DIV :** | **SE CE – A , BATCH B** |
| **DAA EXPT - 3** | |

|  |  |
| --- | --- |
| **AIM** | Use Divide and Conquer Approach : Strassen’s Matrix Multiplication |
| **THEORY** | 1. Time complexity of normal matrix multiplication is given as :   T(N) = 8T(N/2) + O(N2)   1. From Master's Theorem, time complexity of above method is   O(N3)   1. In the normal method, the main component for high time complexity is 8 recursive calls. 2. The idea of Strassen’s method is to reduce the number of recursive calls to 7. 3. Time Complexity of Strassen’s Method :   T(N) = 7T(N/2) + O(N2)   1. From Master's Theorem, time complexity of above method is   O(NLog7) which is approximately O(N2.8074) |
| **CODE** | #include <stdio.h>  #include <time.h>  *void* main()  {  *int* a[2][2], b[2][2], c[2][2], i, j;  *int* p[7];  *int* s[10];      clock\_t start, end;      printf("Enter the elements of 1st matrix:\n");      for (i = 0; i < 2; i++)      {          for (j = 0; j < 2; j++)          {              scanf("%d", &a[i][j]);          }      }      printf("Enter the elements of 2nd matrix:\n");      for (i = 0; i < 2; i++)      {          for (j = 0; j < 2; j++)          {              scanf("%d", &b[i][j]);          }      }      start = clock();      s[0] = b[0][1] - b[1][1];      s[1] = a[0][0] + a[0][1];      s[2] = a[1][0] + a[1][1];      s[3] = b[1][0] - b[0][0];      s[4] = a[0][0] + a[1][1];      s[5] = b[0][0] + b[1][1];      s[6] = a[0][1] - a[1][1];      s[7] = b[1][0] + b[1][1];      s[8] = a[0][0] - a[1][0];      s[9] = b[0][0] + b[0][1];      // 7 strssen calculations      p[0] = s[0] \* a[0][0];      p[1] = s[1] \* b[1][1];      p[2] = s[2] \* b[0][0];      p[3] = s[3] \* a[1][1];      p[4] = s[4] \* s[5];      p[5] = s[6] \* s[7];      p[6] = s[8] \* s[9];      // 4 final output      c[0][0] = p[4] + p[3] - p[1] + p[5];      c[0][1] = p[0] + p[1];      c[1][0] = p[2] + p[3];      c[1][1] = p[4] + p[0] - p[2] - p[6];      for (i = 0; i < 10; i++)      {          printf("\nS[%d] = %d ", i + 1, s[i]);      }      printf("\n");      for (j = 0; j < 7; j++)      {          printf("\np[%d] = %d ", j + 1, p[j]);      }      printf("\n\n");      printf("MATRIX A: \n");      for (i = 0; i < 2; i++)      {          printf("\n");          for (j = 0; j < 2; j++)          {              printf("%d\t", a[i][j]);          }      }      printf("\n");      printf("MATRIX B: \n");      for (i = 0; i < 2; i++)      {          printf("\n");          for (j = 0; j < 2; j++)          {              printf("%d\t", b[i][j]);          }      }      printf("\n");      printf("MATRIX C: \n\n");      printf("%d\t %d\n%d\t %d\n", c[0][0], c[0][1], c[1][0], c[1][1]);      end = clock();      printf("The time taken by the program: ");      printf("%lf", (*double*)(end - start) / CLOCKS\_PER\_SEC);  } |
| **OUTPUT** |  |
| **CONCLUSION** | By performing the above experiment I have successfully understood Divide and conquer algorithm to perform Strassens Multiplication. |