**NAME : ROHAN NYATI**

**ROOL NO : R177219148**

**SAP ID : 500075940**

**BATCH : 5 (AI & ML)**

**LAB 4**

**STRASSEN’S MULTIPLICATION MATRIX**

**CODE**

#include<stdio.h>

#include<time.h>

#include<malloc.h>

void mat\_multint(int n,int \*\*a,int \*\*b,int \*\*c) //normal matrix multiplication

{

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

{

for (int k = 0; k < n; k++)

{

c[i][j] += a[i][k] \* b[k][j];

}

}

}

}

int \*\* matrix\_allocation(int n) //for initialisation & allocation of matrices

{

int \*\* array = (int\*\*)malloc(n\*sizeof(int \*));

for(int i = 0;i < n; i++)

{

array[i] = (int\*)malloc(n\*sizeof(int));

for(int j = 0; j < n; j++)

{

array[i][j] = 0;

}

}

return array;

}

void construct(int\*\* matrix,int\*\* result,int row,int col,int n)

{

int c=col;

for(int i=0;i<n;i++)

{

c=col;

for(int j=0;j<n;j++)

{

result[row][c]=matrix[i][j];

c++;

}

row++;

}

}

int\*\* divide(int \*\* matrix,int n, int row,int col) //for splitting the respective matrices

{

int new\_len=n/2;

int \*\* array = matrix\_allocation(new\_len);

int c=col;

for(int i = 0;i < new\_len; i++)

{

c=col;

for(int j = 0; j < new\_len; j++)

{

array[i][j] = matrix[row][c];

c++;

}

row++;

}

return array;

}

int \*\*add(int \*\*a,int \*\*b,int n) //function for addition

{

int\*\*c = (int\*\*)malloc(n\*sizeof(int\*));

for(int i=0;i<n;i++)

{

c[i] = (int\*)malloc(n\*sizeof(int));

}

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

c[i][j] = a[i][j] + b[i][j];

}

return c;

}

int \*\*subtract(int \*\*a,int \*\*b,int n) //function for subtraction

{

int\*\*c = (int\*\*)malloc(n\*sizeof(int\*));

for(int i=0;i<n;i++)

{

c[i] = (int\*)malloc(n\*sizeof(int));

}

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++)

c[i][j] = a[i][j] - b[i][j];

}

return c;

}

int \*\*strassen\_mult(int \*\*a,int \*\*b,int n)

{

int \*\* c = matrix\_allocation(n);

if(n>1)

{

//allocation and splitting for a

int \*\*a00 = divide(a,n,0,0);

int \*\*a01 = divide(a,n,0,n/2);

int \*\*a10 = divide(a,n,n/2,0);

int \*\*a11 = divide(a,n,n/2,n/2);

//allocation and splitting for b

int \*\*b00 = divide(b,n,0,0);

int \*\*b01 = divide(b,n,0,n/2);

int \*\*b10 = divide(b,n,n/2,0);

int \*\*b11 = divide(b,n,n/2,n/2);

int \*\*m1 = strassen\_mult(add(a00,a11,n/2),add(b00,b11,n/2),n/2);

int \*\*m2 = strassen\_mult(add(a10,a11,n/2),b00,n/2);

int \*\*m3 = strassen\_mult(a00,subtract(b01,b11,n/2),n/2);

int \*\*m4 = strassen\_mult(a11,subtract(b10,b00,n/2),n/2);

int \*\*m5 = strassen\_mult(add(a00,a01,n/2),b11,n/2);

int \*\*m6 = strassen\_mult(subtract(a10,a00,n/2),add(b00,b01,n/2),n/2);

int \*\*m7 = strassen\_mult(subtract(a01,a11,n/2),add(b10,b11,n/2),n/2);

//de-allocating the dynamically allocated pointers storing divided matrices of A & B

free(a00);

free(a01);

free(a10);

free(a11);

free(b00);

free(b10);

free(b01);

free(b11);

int \*\*c00 = add(subtract(add(m1, m4,n/2), m5,n/2), m7,n/2);

int \*\*c01 = add(m3, m5,n/2);

int \*\*c10 = add(m2, m4,n/2);

int \*\*c11 = add(subtract(add(m1, m3,n/2), m2,n/2), m6,n/2);

//constructing the resultant matrix

construct(c00,c,0,0,n/2);

construct(c01,c,0,n/2,n/2);

construct(c10,c,n/2,0,n/2);

construct(c11,c,n/2,n/2,n/2);

//de-allocating the dynamically allocated space for variables and pointers

free(m1);

free(m2);

free(m3);

free(m4);

free(m5);

free(m6);

free(m7);

free(c00);

free(c01);

free(c10);

free(c11);

}

else

{

c[0][0]= a[0][0] \* b[0][0];

}

return c;

}

void print\_Matrix(int \*\* mat,int n) //function for printing matrix

{

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++)

{

printf("%d ",mat[i][j]);

}

printf("\n");

}

}

int main()

{

clock\_t start,end;

double time1,time2;

int n;

printf("enter the order of the arrays: ");

scanf("%d",&n);

int \*\*A = matrix\_allocation(n);

int \*\*B = matrix\_allocation(n);

int \*\*C = matrix\_allocation(n);

printf("\nenter the elements of array a:\n");

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++)

{

scanf("%d",&A[i][j]);

}

}

printf("enter the elements of array b:\n");

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++)

{

scanf("%d",&B[i][j]);

}

}

printf("\n1st matrix:-\n"); //printing matrix A

print\_Matrix(A,n);

printf("\n2nd matrix:-\n"); //printing matrix B

print\_Matrix(B,n);

start=clock();

int \*\* resultant\_matrix = strassen\_mult(A,B,n);

printf("\nStrassen Multiplication Output:\n");

print\_Matrix(resultant\_matrix,n);

end=clock();

time1=((double)(end-start)\*10e9)/CLOCKS\_PER\_SEC;

start=clock();

mat\_multint(n,A,B,C);

end=clock();

printf("output by normal multiplication:\n");

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++)

{

printf("%d ",C[i][j]);

}

printf("\n");

}

time2=((double)(end-start)\*10e9)/CLOCKS\_PER\_SEC;

printf("\ntime taken by brute force = %.2f",time1);

printf("\ntime taken by strassen's algo = %.2f",time2);

return 0;

}

**Output 1:-**

enter the order of the arrays: 2

enter the elements of array a: 5 6 1 7

enter the elements of array b: 6 2 8 7

1st matrix:-

5 6

1 7

2nd matrix:-

6 2

8 7

Strassen Multiplication Output:

78 52

62 51

output by normal multiplication:

78 52

62 51

time taken by brute force = 80000.00

time taken by strassen's algo = 10000.00

**output 2:-**

enter the order of the arrays: 4

enter the elements of array a:

enter the elements of array b:

1st matrix:-

5 2 6 1

0 6 2 0

3 8 1 4

1 8 5 6

2nd matrix:-

7 5 8 0

1 8 2 6

9 4 3 8

5 3 7 9

Strassen Multiplication Output:

96 68 69 69

24 56 18 52

58 95 71 92

90 107 81 142

output by normal multiplication:

96 68 69 69

24 56 18 52

58 95 71 92

90 107 81 142

time taken by brute force = 450000.00

time taken by strassen's algo = 10000.00

**Graph:-**