Data Structure
Chapter 1:Arrays

GATE CS PYQ

Solved By

Monalisa Pradhan

- GATE 2014 Set-1,Q50,2Mark: Consider the following C function in which size is the number of elements in the array E:

```
int MyX(int *E, unsigned int size)
{
    int Y = 0;
    int Z;
    int i, j, k;

    for(i = 0; i < size; i++)
        Y = Y + E[i];

    for(i = 0; i < size; i++)
        for(j = i; j < size; j++)
        {
            Z = 0;
            for(k = i; k <= j; k++)
                Z = Z + E[k];
            if (Z > Y)
                Z = Z;
        }
    return Y;
}
```

- Y=Y+E[i]; It calculates the sum of elements of the array E and stores it in Y.
- Z=Z+E[k]; It calculates the sum of all possible sub arrays starting from 0
- The loop will iterate for all the elements of array E and for every element, calculate sum of all sub arrays starting with E[i].
- Store the current sum in Z.
- If Z is greater than Y then update Y & return Y.
- So it returns the maximum possible sum of elements in any sub array of given array E.
- Ans : (A)

- The value returned by the function **MyX** is the
  (A) maximum possible sum of elements in any sub-array of array **E**.
  (B) maximum element in any sub-array of array **E**.
  (C) sum of the maximum elements in all possible sub-arrays of array **E**.
  (D) the sum of all the elements in the array **E**

GATE CS 2015 Set 3 | Question: 30

- Consider the following two C code segments. *Y* and *X* are one and two dimensional arrays of size n and n×n respectively, where 2≤ n ≤10. Assume that in both code segments, elements of *Y* are initialized to 0 and each element X[*i*][*j*] of array *X* is initialized to *i*+*j*. Further assume that when stored in main memory all elements of *X* are in same main memory page frame.
- Code segment 1:

```
// initialize elements of Y to 0
// initialize elements of X[i][j] of X to i+j
for (i=0; i<n; i++)
            Y[i] += X[0][i];
```

Code segment 2:

```
// initialize elements of Y to 0
// initialize elements of X[i][j] of X to i+j
for (i=0; i<n; i++)
Y[i] += X[i][0];
```

- Which of the following statements is/are correct?
- S1: Final contents of array *Y* will be same in both code segments
- S2: Elements of array X accessed inside the for loop shown in code segment 1 are contiguous in main memory.
- S3: Elements of array X accessed inside the for loop shown in code segment 2 are contiguous in main memory.
- A.Only S2 is correct
- B.Only S3 is correct
- C.Only S1 and S2 are correct
- D.Only S1 and S3 are correct

- Y one dimensional arrays of size n ,X two dimensional arrays of size n×n.2≤ n ≤10 ,Let n=4
- Code segment 1:
- // initialize elements of Y to 0
- // initialize elements of X[i][j] of X to i+j
- for (i=0; i<n; i++)
-             Y[i] += X[0][i];

Y	0	0	0	0
	0	1	2	3

X	0	1	2	3
0	0	1	2	3
1	1	2	3	4
2	2	3	4	5
3	3	4	5	6

C1:Y	0	1	2	3
	0	1	2	3

- i=0 ,Y[0]=Y[0]+X[0][0]=0+0=0
- i=1 ,Y[1]=Y[1]+X[0][1]=0+1=1
- i=2 ,Y[2]=Y[2]+X[0][2]=0+2=2
- i=3 ,Y[3]=Y[3]+X[0][3]=0+3=3
- i=0 ,Y[0]=Y[0]+X[0][0]=0+0=0
- i=1 ,Y[1]=Y[1]+X[1][0]=0+1=1
- i=2 ,Y[2]=Y[2]+X[2][0]=0+2=2
- i=3 ,Y[3]=Y[3]+X[3][0]=0+3=3

- Code segment 2:
- // initialize elements of Y to 0
- // initialize elements of X[i][j] of X to i+j
- for (i=0; i<n; i++)
-       Y[i] += X[i][0];

C2:Y	0	1	2	3
	0	1	2	3

- S1: Final contents of array *Y* will be same in both code segments . True
- S2: Elements of array X accessed inside the for loop shown in code segment 1 are contiguous in main memory. True in case of row major order
- S3: Elements of array X accessed inside the for loop shown in code segment 2 are contiguous in main memory. False
- Ans : C.Only S1 and S2 are correct

GATE 2020,Q22,1 Mark:

Q.No. 22 Consider the following C program.

```
#include <stdio.h>
int main() {
    int a[4][5]={{1, 2, 3, 4, 5},
                 {6, 7, 8, 9, 10},
                 {11, 12, 13, 14, 15},
                 {16, 17, 18 , 19, 20}};
    printf("%d\n", *(*(a+**a+2)+3));
    return(0);
}
```

The output of the program is ________.

**a →

1 [0][0]	2 [0][1]	3 [0][2]	4 [0][3]	5 [0][4]
6 [1][0]	7 [1][1]	8 [1][2]	9 [1][3]	10 [1][4]
11 [2][0]	12 [2][1]	13 [2][2]	14 [2][3]	15 [2][4]
16 [3][0]	17 [3][1]	18 [3][2]	19 [3][3]	20 [3][4]

- Pointer to Array
- **a=a[0][0]
- a[i][j] =*(a[i] +j) =*(*(a+i)+j)

- *(*(a+**a+2)+3)
- = *(*(a+a[0][0]+2)+3)
- = *(*(a+1+2)+3)
- = *(*(a+3)+3)
- =*(a[3]+3)
- =a[3][3]=19
- Ans :19