19BIT0292

Bhaumik Tandan

DIGITAL ASSIGNMENT-1

DATA STRUCTURES AND ALGORITHMS

CSE2011

D1+TD1

Q1) Design an efficient algorithm that achieves the following task: Given an array A[1..n] of floating point numbers, it returns a two-dimensional array, say M, of size n × n in which the entry M[i][j] for i ≤ j contains the average of the array entries A[i] through A[j]. That is: if i ≤ j, then

$$M[i][j] = (A[i] + \cdots + A[j])/(j - i + 1)$$

whereas for i > j we have that M[i][j] = 0

(1) Describe your idea for an algorithm that creates this matrix.

ALGORITHM

- Start
- Take an input for the number of elements from the user in n
- Assign memory to the array that can store n numbers
- Take input in the array
- Allocate memory to a 2d matrix that can store nXn elements
- For the first row of the matrix run a iterative loop for i=0 to i<n
- Save the tentative sum till the current column.
- Using the tentative sum calculate the tentative average
- Store the tentative Average in the first row of matrix
- Run a loop for all subsequent rows in the matrix
- Run another nested loop inside it for subsequent column
- For each element subtract the index of current row number from the tentative average stored in the same column in the previous row.
- Print the matrix

(2) Write down the algorithm in pseudocode

PSEUDOCODE

```
function returning float value avg_subtract(average, element, n)
start
   return ((average*n)-element)/(n-1);
end

function disp(matrix as a pointer to float pointer, n)
start

   For i=0 to i=n-1

      For j=0 to j=n-1
        print matrix[i][j] upto 2 decimal places
      ENDFOR

      change line
    ENDFOR

end

function returning float pointer takeinput(integer pointer to n)
start
   declare a float pointer a
   Take input for the size of the array into n
   Allocate memory of n integer to a

   For i=0 to i=n-1
         Take input into a[i]
   ENDFOR

   return a
end

function cal_avg(pointer to matrix, n, pointer to arr)
   Initialize sum to 0
   For i=0 to i=n-1
```

```
      Add arr[i] to sum and
      store sum/(i+1) to matrix[0][i]
    ENDFOR

set_matrix(pointer to matrix, n, pointer to arr)
start
    call cal_avg by passing matrix, n and arr

    For i=1 to i=n-1

      For j=i to n-1
        Call avg_subtract(matrix[i-1][j],arr[i-1],j-i+2)
        Store the returned value into matrix[i][j]
      ENDFOR

    ENDFOR

end

main
start
    Initialize  n
    Call takeinput(address of n)
    Assign returned value to float pointer  arr
    Call allocate_memory(n)
    Assign returned value to pointer to float pointer  matrix
    Call set_matrix(matrix,n,arr)
    Call disp(matrix,n)
end
```

(3) How many assignments operations will your algorithm perform for an input of size n?

In the algorithm we only give value to upper triangular matrix,
for 1st row we will have n assignments,
2nd row = n-1 assignments
3rd =n-2
assignments……..nth= 1 assignment
1+2+3…...n=n(n+1)/2
That means we ill have n(n+1)/2 assignments for n elements.

(4) Implement the algorithm in the C language and produce results.

```
#include<stdio.h>
#include <stdlib.h>

int count;

float avg_subtract(float avg,float e,int n)
{
   return ((avg*n)-e)/(n-1);
}

void disp(float **matrix,int n)
{
   printf("\nThe matrix is:- \n");

   for(int i=0;i<n;i++)
   {
      printf("\n");

      for(int j=0;j<n;j++)
      printf("%.2f ",matrix[i][j]);

   }
}

float* takeinput(int *n)
{
   float *a;
   printf("Enter the size of array: ");
   scanf("%d",n);
   a=(float*)malloc(sizeof(float)*(*n));
   printf("Enter the array: ");
   for(int i=0;i<*n;i++)
   scanf("%f",a+i);
   return a;
}
```

```
float** allocate_memory(int n)
{
   float **matrix=(float**)malloc(sizeof(float*)*n);
   for(int i=0;i<n;i++)
   matrix[i]=(float*)calloc(sizeof(float),n);
   return matrix;
}

void cal_avg(float **matrix,int n,float* arr)
{
   float sum=0;
   for(int i=0;i<n;i++)
   {
      sum+=arr[i];
      matrix[0][i]=sum/(i+1);
      count++;//matrix allocated
   }
}

void set_matrix(float **matrix,int n,float* arr)
{
   cal_avg(matrix,n,arr);
   for(int i=1;i<n;i++)
      for(int j=i;j<n;j++){
      matrix[i][j]=avg_subtract(matrix[i-1][j],arr[i-1],j-i+2);
      count++;//matrix allocated
      }
}

main()
{
   int n;
   float *arr=takeinput(&n);
   float **matrix=allocate_memory(n);
   set_matrix(matrix,n,arr);
   disp(matrix,n);
   printf("\nAssignments operations: %d",count);
}
```

OUTPUT

```
Enter the size of array: 4
Enter the array: 1 2 3 4

The matrix is:-

1.00 1.50 2.00 2.50
0.00 2.00 2.50 3.00
0.00 0.00 3.00 3.50
0.00 0.00 0.00 4.00
Assignments operations: 10
```

```
Enter the size of array: 3
Enter the array: 1 2 3

The matrix is:-

1.00 1.50 2.00
0.00 2.00 2.50
0.00 0.00 3.00
Assignments operations: 6
```

```
Enter the size of array: 5
Enter the array: 1 2 3 4 5

The matrix is:-

1.00 1.50 2.00 2.50 3.00
0.00 2.00 2.50 3.00 3.50
0.00 0.00 3.00 3.50 4.00
0.00 0.00 0.00 4.00 4.50
0.00 0.00 0.00 0.00 5.00
Assignments operations: 15
```

We can validate it practically, what we derived theoretically in (3)

when n=4 => n(n+1)/2 = (4*5)/2=10
when n=3 => n(n+1)/2 = (3*4)/2=6
when n=5 => n(n+1)/2 = (5*6)/2=15

(5) Calculate the time complexity for running your algorithm with n input.

Since there are assignment n(n+1)/2 and all the preprocessing before each assignment takes linear time we can reach to the conclusion that the time complexity of the code is $n(n+1)/2 = (n^2 + n)/2$

Hence the time complexity is $O(n^2)$

CLICK HERE FOR GITHUB LINK

Q2) . A palindrome is a phrase that reads the same forward and backward (examples: ‘racecar’, ‘radar’, ‘noon’, or ‘rats live on no evil star’). By extension we call every string a palindrome that reads the same from left to right and from right to left. Develop a recursive algorithm that takes as input a string and decides whether the string is a palindrome. Use appropriate date structure to implement the same, without using built in function.

CODE

```
#include<stdio.h>
#include <string.h>

int check(char* str,int s,int e)
{
   if(s>e)
   return 1;
   if(str[s]!=str[e])
   return 0;

   return check(str,s+1,e-1);
}

main()
{
   char str[100];
   int i=0;
   printf("\nEnter the string: ");
   gets(str);

   if(check(str,0,strlen(str)-1))
   printf("Palindrome");

   else
   printf("Not palindrome");
}
```

```
Enter the string: racecar
Palindrome
```

```
Enter the string: radar
Palindrome
```

```
Enter the string: noon
Palindrome
```

```
Enter the string: efee
Not palindrome
```

```
Enter the string: rats live on no evil star
Palindrome
```

```
Enter the string: bhaumik tandan 19BIT0292
Not palindrome
```

CLICK HERE FOR GITHUB LINK