Data Structures and Algorithms (CS F211) – T3

Prof.N.L.Bhanu Murthy

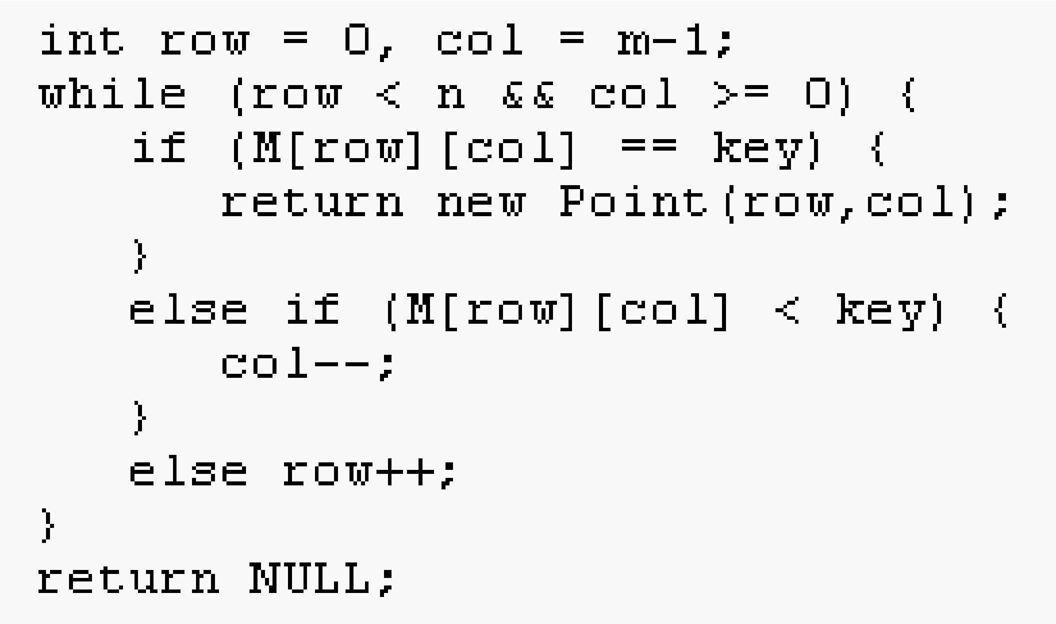
BITS Pilani
Hyderabad Campus

Problem 1

Let *M* be an n X m integer matrix in which the entries of each row are sorted in increasing order (from left to right) and the entries in each column are in increasing order (from top to bottom). Give an efficient algorithm to find the position of an integer *x* in *M*, or to determine that *x* is not there. How many comparisons of *x* with matrix entries does your algorithm use in worst case?

Problem 1

Let *M* be an n X m integer matrix in which the entries of each row are sorted in increasing order (from left to right) and the entries in each column are in increasing order (from top to bottom). Give an efficient algorithm to find the position of an integer *x* in *M*, or to determine that *x* is not there. How many comparisons of *x* with matrix entries does your algorithm use in worst case?



```
int row = 0, col = m-1;
while (row < n && col >= 0) {
    if (M[row][col] == key) {
        return new Point(row,col);
    }
    else if (M[row][col] < key) {
        col--;
    }
    else row++;
}
return NULL;
```

Suppose that each row of an n x n array A consists of 1's and 0's such that, in any row i of A , all the 1's come before any 0's in that row. Suppose further that the number of 1's in row i is at least the number in row i + 1, for i = 0,1,2,....,n-2 Assuming A is already in memory , describe a method running in O(n) time for counting the number of 1's in the array A.

Suppose that each row of an n X n array consists of 1’s and 0’s such that, in any roq ofA, all the 1’s come before any 0’s in that row. Assuming A is already in memory, describe a method running in O(n)time for finding the row of A that contains the most 1’s.

Give a recursive algorithm to compute the product of two positive integers m and n using only addition.

Give a recursive algorithm to compute the product of two positive integers m and n using only addition.

```
int multiply(int m, int n)
{
        int result;

        if (n == 1)
                result =  m;
        else
                result =  m + multiply(m, n-1);
        return(result);
}
```

Problem:

Describe a method for finding both the minimum and maximum of n numbers using fewer than 3n/2 comparisons.

Problem:

Describe a method for finding both the minimum and maximum of n numbers using fewer than 3n/2 comparisons.

Sol:

Sort n/2 pairs. Find min of losers, max of winners.

# comparisons: n/2 + n/2 –1 + n/2-1 = 3n/2 –2.

Thank You!!