Solution to problem 3:

If we take under consideration the properties of the columns and rows, we can clearly see that if we split the original matrix in 4 parts we can eliminate at least one if we compare the values of the top left and bottom right corners of each sub-matrix to the key we are searching for. Thus adopting this approach T(N)=3T(N/4) + C, where C is constant and expresses the time needed for comparison with the diagonal elements described above.

Inputs: Key that we are searching for, 2 dimensional array A, and its endpoints as Cartesian couples (initially 1,1,n,m where n=column #, m=row #) within the table.

Outputs: True if the key is in the array and false if it isn't.

Algorithm:

```
matrix_search(key, A, x1, y1, x2, y2)
        if x1=x2 and y1=y2 and A[x1][y1] = key
                return true
         else
                if A[x1][y1]=key or A[x1][y2]=key or A[x2][y1]=key or A[x2][y2]=key
                                        or A[(x1+x2)/2][(y1+y2)/2] = key
                        return true
                else
                        if key > A[(x1+x2)/2][(y1+y2)/2]
                                if matrix_search(A, (x1+x2)/2, (y1+y2)/2, x2, y2) is true
                                        return true
                        if key < A[(x1+x2)/2][(y1+y2)/2]
                                if matrix_search(A, x1, y1, (x1+x2)/2, (y1+y2)/2) is true
                                        return true
                        if matrix_search((x1+x2)/2, y1. x2, (y1+y2)/2) is true
                                return true
                        if matrix_search(x1, (y1+y2)/2. (x1+x2)/2, y2) is true
                                return true
        return false
```

To find T(n) we will sum up all the possible C-time checks performed until we reach T(1).
At the root level we will have C comparisons, next level we will have 3*C, next (3^2)*C, and so on, thus at the last level we will have 3^(log[4](n))*C comparisons (since the depth of the tree is appr. log[4](n)). The number of all comparisons is:

$$(i\in[0..\log_4(N)-1)\sum 3^i*C + 3^{(\log_4(N)}$$
$$=(3^{(\log_4(N)} - 1)*C/2 + 3^{(\log_4(N)}$$
$$=(C*3^{(\log_4(N)} - C + 2*3^{(\log_4(N)}) / 2$$
$$=(C*(N^{.8}) - C + 2*(N^{.8})) / 2$$
$$=O(N^{.8})$$