Name: Changzhi Cai　　　　　　　　　　　　Perm number: 9911579

Algorithm of this program:

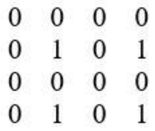
Import a binary image (from pattern1.mat, pattern2.mat or custom.txt (users can change input matrix arbitrarily)

Get the image’s size (m, n)
Create a same size zero matrix ‘transImage’ to prepare for storing the Manhattan distance of each element

```
For i from 1 to m: traverse every row in the matrix
    For j from 1 to n: traverse every element in the row
    Create a matrix T to store the Manhattan distance between this element and each object
        For i1 form 1 to m:
            For j1 from 1 to n: traverse every element in the matrix to find all objects
            If an object is found:
            Calculate the Manhattan distance between these two elements using |i1-i|+|j1-j|
            Store the distance into T matrix
        End searching based on this element: find the minimum value from T and store it to the
        ‘transImage’ matrix
    End of traverse this row
End of traverse this matrix
```

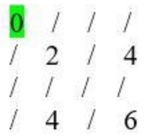
Save the transformed image as .mat file
Output the transformed image

Algorithm illustration using a 4 x 4 example:



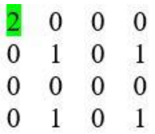
0	0	0	0
0	1	0	1
0	0	0	0
0	1	0	1

Figure 1.1 Input image.



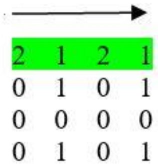
0	/	/	/
/	2	/	4
/	/	/	/
/	4	/	6

Figure 1.2 Consider the Manhattan distance between the first element and all objects.



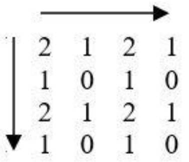
2	0	0	0
0	1	0	1
0	0	0	0
0	1	0	1

Figure 1.3 Take the minimum distance.



2	1	2	1
0	1	0	1
0	0	0	0
0	1	0	1

Figure 1.4 Traverse all elements in this row.

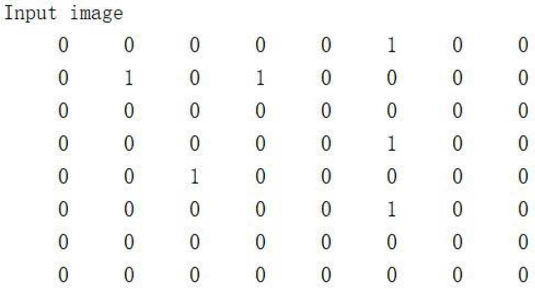


2	1	2	1
1	0	1	0
2	1	2	1
1	0	1	0

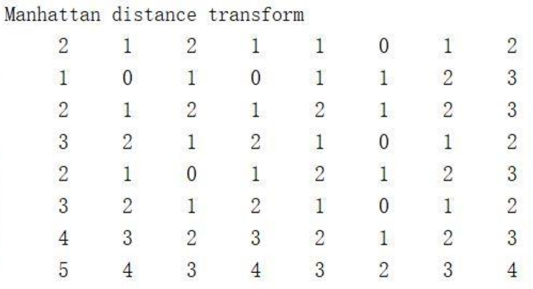
Figure 1.5 Traverse the whole matrix and get the final distance transform.

Take two 8 x 8 examples:

Example #1 (Import pattern1.mat)



```
Input image
     0     0     0     0     0     1     0     0
     0     1     0     1     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     1     0     0
     0     0     1     0     0     0     0     0
     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
```

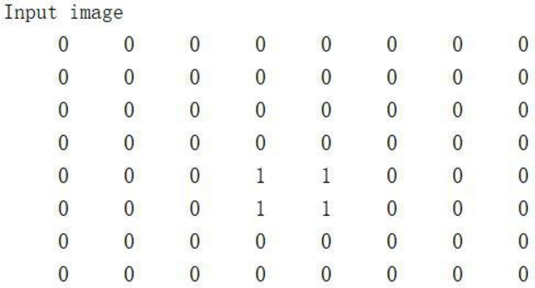


```
Manhattan distance transform
     2     1     2     1     1     0     1     2
     1     0     1     0     1     1     2     3
     2     1     2     1     2     1     2     3
     3     2     1     2     1     0     1     2
     2     1     0     1     2     1     2     3
     3     2     1     2     1     0     1     2
     4     3     2     3     2     1     2     3
     5     4     3     4     3     2     3     4
```

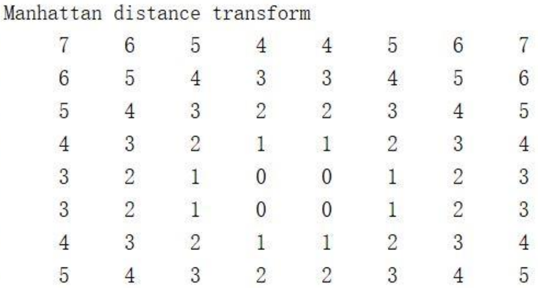
Figure 2.1 Import binary image

Figure 2.2 Output image

Example #2 (Import pattern2.mat)



```
Input image
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     1     1     0     0     0
     0     0     0     1     1     0     0     0
     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0
```



```
Manhattan distance transform
     7     6     5     4     4     5     6     7
     6     5     4     3     3     4     5     6
     5     4     3     2     2     3     4     5
     4     3     2     1     1     2     3     4
     3     2     1     0     0     1     2     3
     3     2     1     0     0     1     2     3
     4     3     2     1     1     2     3     4
     5     4     3     2     2     3     4     5
```

Figure 3.1 Import binary image

Figure 3.2 Output image