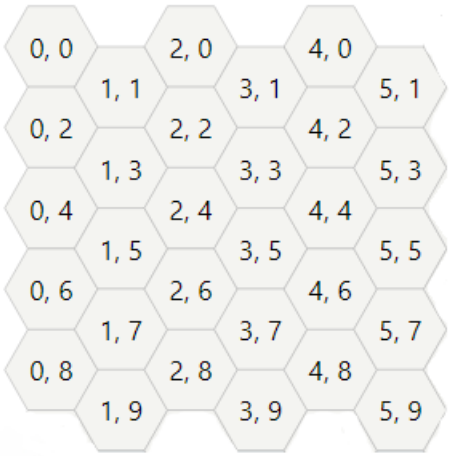
EE431 HOMEWORK 1 - 501504512

1. Our proposed hexagonal tessellation into 2D array strategy is doubled height coordinates. We used x coordinate for left to right change, y coordinate for up to down change.

Going only x coordinate adds x to 2, similarly going only y coordinate adds y to 2. If we move one hexagon crisscross, our grid adds x to 1 and y to 1. So, every hexagon can be defined in 2D array.



a) The distance between two points in the tessellation can be easily calculated with x and y values and one side length of a hexagonal A.

$$Ecludian\ Distance = \sqrt{\left((x_2 - x_1) * w\right)^2 + \left((y_2 - y_1) * h\right)^2}\ where\ w = \frac{3}{2}A,\ h = \frac{\sqrt{3}}{2}A$$

b)

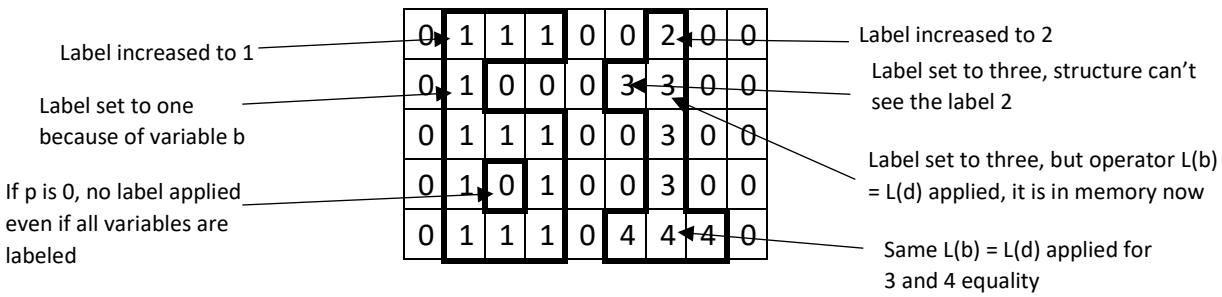
6-adjacency	Coordinate	Up-right	(x+1, y-1)
Center	(x, y)	Up-left	(x-1, y-1)
Up	(x, y-2)	Down-right	(x+1, y+1)
Down	(x, y+2)	Down-left	(x-1, y+1)

2. We have 4 variables so we can use 16 operators for labeling.

a	b	c
d	p	

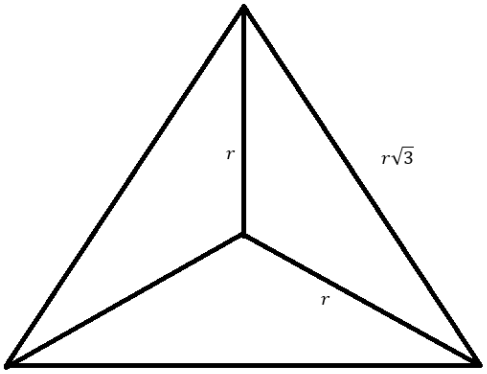
a b c d	Operator	a b c d	Operator
0 0 0 0	L(p) := i*N+j or L(p) := L + 1	0 1 1 0	L(p) := L(c), L(b) = L(c)
0 0 0 1	L(p) := L(d)	1 0 1 0	L(p) := L(c), L(a) = L(c)
0 0 1 0	L(p) := L(c)	1 1 0 0	L(p) := L(b), L(a) = L(b)
0 1 0 0	L(p) := L(b)	0 1 1 1	L(p) := L(d), L(b) = L(c) = L(d)
1 0 0 0	L(p) := L(a)	1 0 1 1	L(p) := L(d), L(a) = L(c) = L(d)
0 0 1 1	L(p) := L(d), L(c) = L(d)	1 1 0 1	L(p) := L(d), L(a) = L(b) = L(d)
0 1 0 1	L(p) := L(d), L(b) = L(d)	1 1 1 0	L(p) := L(c), L(a) = L(b) = L(c)
1 0 0 1	L(p) := L(d), L(a) = L(d)	1 1 1 1	L(p) := L(d), L(a) = L(b) = L(c) = L(d)

Example single pass with operations on an image of number 61 can be visualized like this:



3. 2D shapes needs at least 3 sides, which is a triangle, and maximum we could have infinity sides, which is a circle. Since the compactness of a circle known as the lowest, no need to show all of the shapes and their compactness, but a couple of them would satisfy this theory.

For equilateral triangle:

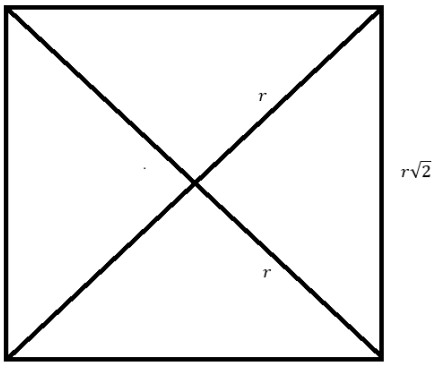


$P_{equilateral\ triangle} = 3\sqrt{3}r$ and $A_{equilateral\ triangle} = \frac{3\sqrt{3}\,r^2}{4}$

By using $P_{equilateral\ triangle}$ and $A_{equilateral\ triangle}$, compactness could be calculated as follows,

$$C_{equilateral\ triangle} = \frac{|P|^2}{A} = \frac{27r^2}{\frac{3\sqrt{3}\,r^2}{4}} = 20.81$$

For square:

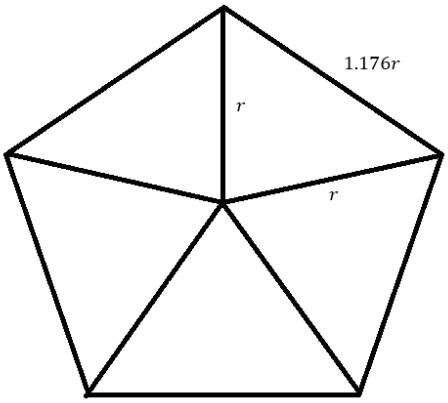


$P_{square} = 4\sqrt{2}r$ and $A_{square} = 2r^2$

Compactness of the square is:

$$C_{square} = \frac{32r^2}{2r^2} = 16$$

For pentagon:

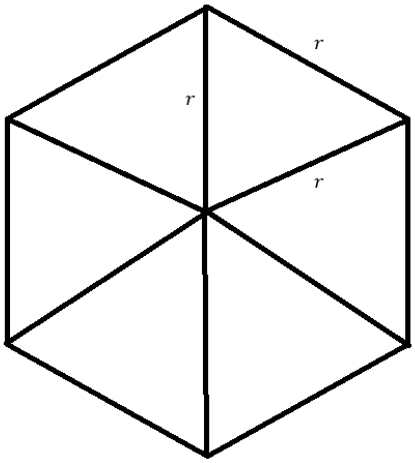


$$P_{pentagon} = 5.88r \qquad \text{and} \quad A_{pentagon} = 2.38r^2$$

Compactness is as follow,

$$C_{pentagon} = \frac{34.57r^2}{2.38r^2} = 14.53$$

For hexagon:



$$P_{hexagon} = 6r \quad \text{and} \quad A_{hexagon} = \frac{3\sqrt{3}\,r^2}{2}$$

Compactness is as follow,

$$C_{hexagon} = \frac{36r^2}{\frac{3\sqrt{3}\,r^2}{2}} = 13.856$$

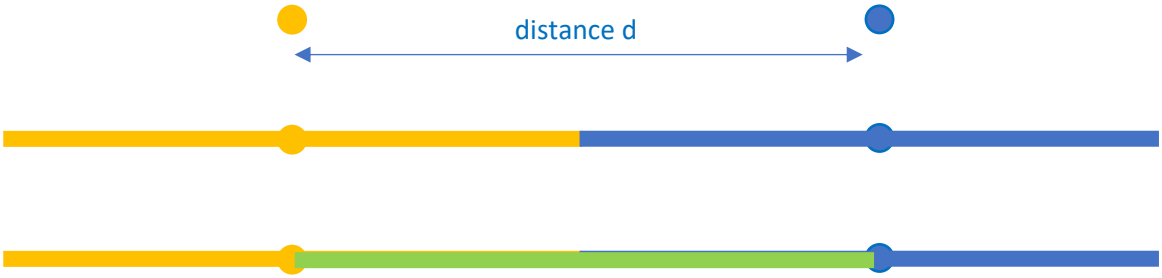
As seen from the results above we can say that when the number of edges increases, compactness tends to decrease. After that, a circle could be seen as a polygon with infinite edges.

When the compactness of the circle calculated, the smallest compactness value have been observed. Calculation as follows,

$$P_{circle} = 2\pi r \quad \text{and} \quad A_{circle} = \pi r^2$$

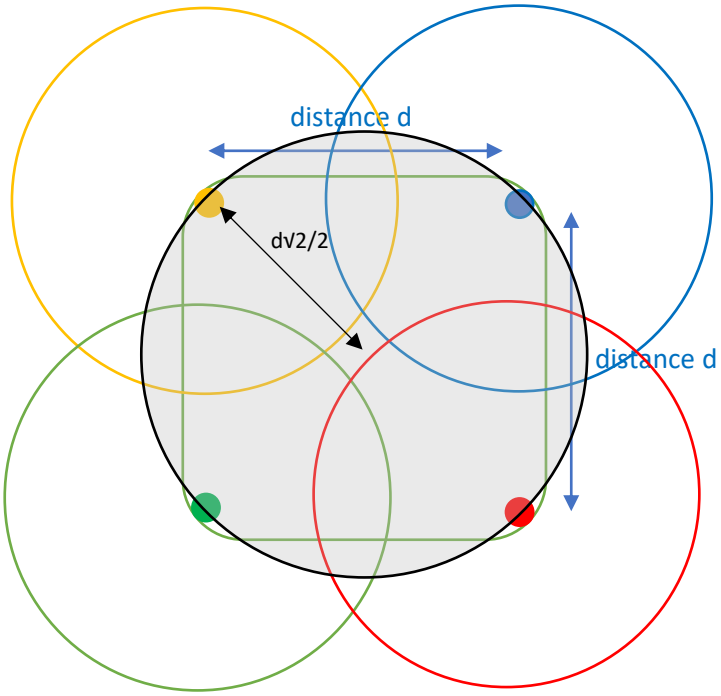
$$C_{circle} = \frac{4\pi^2 r^2}{\pi r^2} = 4\pi = 12.566.$$

4. a) Closing is combination of dilation and erosion respectively, we can visualize these operations for two separated points like the followings.



We used a horizontal line for structure element. After dilation there is an intersection point between them if the line is longer than or equal to d. In erosion operation, every undeleted point should fit a structure element in dilation area, the middle intersection point can fit this green line, so the connection don't lost, and the minimum line length is d.

b) For this question we use the same approach for closing operation, but we should generate an area for fitting a circle, so second dimensions are important unlike part a. We visualize the points and dilation operation like the following.



We can see we can't fit any circle in generated areas even between two points. For fitting a circle in there, we need to use the middle area of the square, so the structure element dilation operation needs to fill in there.

So, when we use a circle with radius $d\frac{\sqrt{2}}{2}$, erosion operation can't delete the connections between points.

5. For rotating image instead of starting point, first a shifting operation should be applied and after that rotation operation should applied to the image. After completing past steps image should be shifted to the starting point. Lastly, the scaling operation could be done.

Step 1:

$$A_{shifted} = A - (\frac{NR}{2}, \frac{NC}{2})$$

Step 2:

$$A_{rotated} = \begin{pmatrix} \cos 45 & -\sin 45 \\ \sin 45 & \cos 45 \end{pmatrix} A_{shifted}$$

Step 3:

$$A_{reshifted} = A_{rotated} + \left(\frac{NR}{2}, \frac{NC}{2}\right)$$

Step 4:

$$A_{scaled} = \begin{pmatrix} 4 & 0 \\ 0 & 3 \end{pmatrix} A_{reshifted}$$

Each step should be applied to every element of the image.

6. We can easily use the example1.c file on the SgimproV1.2b file. We used cygwin64 terminal and irfanview software to execute code and view the images.

The program needs a gamma value and an image file for input arguments, so argc control should be at 3, gamma value comes from argv[1] and pgm_file comes from argv[2]. We need to convert gamma argument to a float, and pgm file to an image. We did these operations with atof and pgm_file_to_img functions which is in img_pro.h

```
if(argc!=3) {printf("\n Usage: question6 gamavalue cathedral.pgm \n"); exit(-1); }

gama = atof(argv[1]);
pgm_file = argv[2];
img = pgm_file_to_img(pgm_file, &NC, &NR);
```

For applying gamma correction, we need to reach every value of the image and convert its value to a new one, calculated with gamma and the original value.

```
for(i=0;i<NR;i++)
  for(j=0;j<NC;j++)
    img[i][j] = 255 * (pow(img[i][j]/255.0,1.0/gama));
```

We can save and show the new file with the following functions.

```
img_to_pgm_file(img,"deneme.pgm", NC, NR);
show_pgm_file("deneme.pgm");
```

Results for gamma values:


gamma = 2


gamma = 1


gamma = 0.81


gamma = 0.36


gamma = 0.09

Full code for question 6:

```c
#include "img_pro.h"
#include "my_header.h"

int main(int argc, char **argv)
{
unsigned char **img, **img2;
char *pgm_file;
int i,j,k,l,NR,NC;
float gama;

/*---------->>> Start assigning variables from command line arguments   */
if(argc!=3) {
  printf("\n Usage: q6 gama [Image file (*.pgm)] \n");
  printf("\n E.g.   q6 20 cathedral.pgm \n");
  exit(-1); }

gama = atof(argv[1]); if(gama<-250 || gama> 250) {printf("\n gama should be
between 0 and 255\n"); exit(0);}
pgm_file = argv[2];
/*----------<<< End assigning valiables from command line arguments  */

img = pgm_file_to_img(pgm_file, &NC, &NR);/*  read img and its size from file */
// show_pgm_file(pgm_file);

for(i=0;i<NR;i++)
  for(j=0;j<NC;j++)
    img[i][j] = 255 * (pow(img[i][j]/255.0,1.0/gama));

img_to_pgm_file(img,"deneme.pgm", NC, NR);
show_pgm_file("deneme.pgm");

// free images
free_img(img);

return(1);
}
```