MA2507 Computing Mathematics Laboratory: Week 9

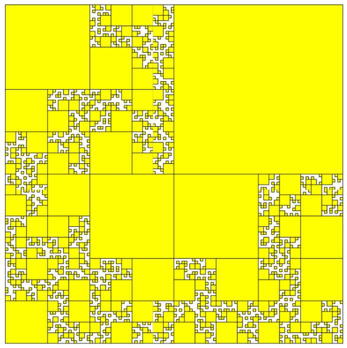
1. Recursive functions. A recursive function is a function that calls itself. Sometimes, you can write elegant programs using recursive functions. Our first example is *random boxes*. We have a recursive function `drawbox(x0,x1,y0,y1,m)`, where $x0$, $x1$, $y0$, $y1$ are the coordinates of a square, m is a control integer. If m is positive, the program divides the square into four smaller ones, colors one small square by yellow at random, and continues `drawbox` on the other three smaller squares with control integer $m-1$. The main program draws a unit square and then carries out `drbox` for some small and positive m.

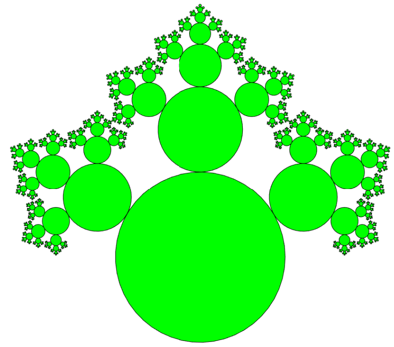
```
% main script
clf
plot([0 1 1 0 0],[0 0 1 1 0],'k')
hold on
axis equal off
drawbox(0,1,0,1,7)
hold off

% recursive function, after the main script
function drawbox(x0,x1,y0,y1,m)
% given a square [x0,x1] x [y0, y1]
if m > 0
    x = [x0, (x0+x1)/2, x1]; % x-coordinates of smaller squares
    y = [y0, (y0+y1)/2, y1]; % y-coordinates of smaller squares
    k = randi(4);
    for i=1:2
        for j=1:2
            kk = 2*i+j-2;   % convert (i,j) to kk in {1, 2, 3, 4}
            if kk == k
                fill([x(i),x(i+1),x(i+1),x(i)],[y(j),y(j),y(j+1),y(j+1)],'y')
            else
                drawbox(x(i),x(i+1),y(j),y(j+1),m-1)
            end
        end
    end
end
end
```

We get the yellow figure on the next page.

As another example, we write a program to draw many disks of different size. The recursive function is `w9exr(c,r,a,m)` where c is the coordinates of the the current disk, r is the radius, m is an integer to control the level, a is an angle with the y axis for one of the three disks at level $m-1$. These three disks are tangent to the current disk, and they have radius $0.5r$, $0.4r$ and $0.4r$, respectively. The straight lines connecting the center of the current disk to the centers of the three smaller disks form angles a, $a+\pi/3$ and $a-\pi/3$ with the y axis, respectively. The main script stats the program with a unit disk and angle $a=0$ for some $m>0$.

```
% main script
```





```
clf
hold on
w9exr([0,0],1,0,6)
axis equal off
hold off

% a function after the main script
function w9exr(c,r,a,m)
% c -- vector for the location of the center
% r -- radius of the current disk
% a -- angle for central disk of the next level
t = linspace(0,2*pi,50);
x=c(1)+r*cos(t);
y=c(2)+r*sin(t);
fill(x,y,'g')
if m > 0
    c1 = c + 1.5*r*[sin(a),cos(a)];
    w9exr(c1,0.5*r,a,m-1);
    a2 = a +pi/3;
    c2 = c + 1.4*r*[sin(a2),cos(a2)];
    w9exr(c2,0.4*r,a2,m-1);
    a3 = a - pi/3;
    c3 = c + 1.4*r*[sin(a3),cos(a3)];
    w9exr(c3,0.4*r,a3,m-1);
end
end
```

The above program gives the green figure above.

2. Fast Fourier Transform: FFT is an efficient method for computing the discrete Fourier transform (DFT). DFT is the following matrix-vector multiplication:

$$\boldsymbol{y} = \mathbf{F}_N \boldsymbol{x}, \tag{1}$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are column vectors of length N, $\mathbf{F}_N$ is an $N \times N$ matrix whose $(j+1, k+1)$ entry equals to ω^{jk} where $\omega = e^{i2\pi/N}$. More explicitly,

$$\mathbf{F}_N = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \dots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \dots & \omega^{(N-1)^2} \end{bmatrix}.$$

The matrix $\mathbf{F}_N$ is symmetric, but not Hermitian. It is not unitary, but it is close enough. We can prove that

$$\mathbf{F}_N \overline{\mathbf{F}}_N = N\mathbf{I} \tag{2}$$

Therefore, $(1/\sqrt{N})\mathbf{F}_N$ is unitary. Usually, when DFT is studied, it is easier to start index from 0 for the vectors $\boldsymbol{y}$ and $\boldsymbol{x}$, that is

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{N-1} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \end{bmatrix}.$$

Now Eq. (1) implies

$$y_j = \sum_{k=0}^{N-1} \omega^{jk} x_k = \sum_{k=0}^{N-1} x_k e^{i2\pi jk/N}, \quad j = 0, 1, ..., N-1. \tag{3}$$

From Eqs. (2) and (1), we get

$$\boldsymbol{x} = \frac{1}{N} \overline{F}_N \mathbf{y} \tag{4}$$

which is equivalent to

$$x_k = \frac{1}{N} \sum_{j=0}^{N-1} y_j e^{-i2\pi jk/N}, \quad k = 0, 1, ..., N-1. \tag{5}$$

Now, we describe FFT assuming N is a power of 2. We split $\boldsymbol{y}$ into two vectors of length $N/2$ by

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \end{bmatrix},$$

and put the even and odd elements of $\boldsymbol{x}$ into two vectors

$$\boldsymbol{x}_1 = \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ \vdots \end{bmatrix}, \quad \boldsymbol{x}_2 = \begin{bmatrix} x_1 \\ x_3 \\ x_5 \\ \vdots \end{bmatrix}.$$

The basic principle of FFT is the following. If $\boldsymbol{a} = \mathbf{F}_{N/2}\boldsymbol{x}_1$ and $\boldsymbol{b} = \mathbf{F}_{N/2}\boldsymbol{x}_2$, then

$$\boldsymbol{y}_1 = \boldsymbol{a} + \mathbf{W}\boldsymbol{b}, \quad \boldsymbol{y}_2 = \boldsymbol{a} - \mathbf{W}\boldsymbol{b},$$

where $\mathbf{W}$ is an $(N/2) \times (N/2)$ diagonal matrix with diagonal entries 1, ω, ω^2, ... You can prove this result starting from Eq. (3). It is all elementary mathematics. Notice that $e^{i2\pi} = 1$.