In-Class Work: DLA

I will start class with reminding us of the rules of the DLA model and will develop the flow chart for the DLA model.

1. Fractal Growth: Random Walk in Two Dimensions

1a. I will guide us through the solution of the random walk in two dimensions. You find the solution in
`~kvollmay/share.dir/inclass2025.dir/classfractal1a.py`
I will run this program and look at $x(t)$ and $y(t)$ with
`./classfractal1a.py > j; xmgrace -block j -bxy 1:2 -bxy 1:3`

1b. Movie

I will also guide us through making a movie of the random walker of `NSTEPS=50` steps on a 30x30 lattice, starting on the site $x = 15$ and $y = 15$. The program makes snapshots of the lattice, putting a lattice value 2 for the current random walker site and putting a lattice value 1 for all previously visited lattice sites. The soluton is in
`~kvollmay/share.dir/inclass2025.dir/classfractal1b.py`
When you run this `classfractal1b.py` it makes the snapshot pictures `frame*.png`.
You can watch a movie with
`animate -delay 30 -pause 5 frame*png`

3. Start Random Walker on Circle

Initialize the lattice with all lattice sites being zero and only in the middle of the lattice is a seed with value one. Use a lattice size `LATSIZE=100`. We want next to implement that a new random walker starts randomly somewhere on a circle (uniformly distributed) with midpoint in the middle of the lattice and with radius `RMAX+2`. For the DLA-program we will use `RMAX=2` but for now use `RMAX=20`. To check that you draw your random walker indeed equally likely on a circle, add to your program that you put 50 initial walkers on their starting point (not yet with random walk steps) and mark for each of these 50 starting points the lattice with the value 2. Make only one image of your lattice after these 50 markings on the lattice. To get a pdf-file of your frame use (similar to above) in the header of program

```
import matplotlib.pyplot as plt
```

and for example

```
plt.imshow(lattice,interpolation='nearest')
plt.savefig('frame3.pdf')
```

Think about how you would program this task. When you have planned your strategy, instead of programming it, you may copy the solution program
`~kvollmay/share.dir/inclass2025.dir/classfractal3.py`
into your working directory. Read the program, to convince yourself what it does and run the program. Look at the resulting `frame3.pdf` by using the command `evince frame3.pdf`.

4. DLA: Random Walk Copy `classfractal3.py` into another file, e.g. into `classfractal4.py`. Modify `classfractal4.py`. Take out of this program the loop over 50 starting points, instead start the random walker at **only one** point on the circle and use `RMAX=2`. Add to your program a loop over `NSTEP=50` random walk steps. To check your program, assign to each site, which is visited by the random walker, the value 2. Make a picture of the lattice after this random walk, so make `frame4.pdf` and look at it.

5. DLA: Distance

5a. We work next on step IV of the DLA rules, for which you need to keep track of the distance r of your walker from the midpoint of the lattice. Add to your program of 4. the determination of r . (Hint: For $\sqrt{}$ in python use in the header, as already in the sample file, `import numpy as np` and then for example $\sqrt{5}$ would be `np.sqrt(5)`). To check your program print for every random walk step `x,y,LATMID,r` and check by hand that you get the right distance from the midpoint of the lattice. Do `NSTEP=200` random walk steps.

5b. Now replace the loop of 200 random steps with a while loop `while walkstop == 0:`. Set `walkstop = 0` before this while loop and set `walkstop = 1`, as soon as $r \geq 2\,R_{\max}$. Initialize $R_{\max}$ to $R_{\max} = 3.0$ and `np.random.seed(11)`. Run your program and check that your stopping of the random walk works as intended.