Lab 9
April 14, 2017

Percolation
(Based on chapter 7 of "Computational Physics" by Giordano and Nakanishi)

Here we will use the Hoshen-Kopelman method to label the clusters in a square lattice.

In this example, we will define a square lattice of 10x10 called n. Each site of the lattice is occupied at random based on a certain probability p. All lattice sites (matrix elements) are initialized to 0. Once a site becomes occupied, the value of n[i,j] at that site (row i, column j) is changed to 1, resulting in a 10x10 matrix of zeros and ones.

To identify and label clusters in this lattice, we will use the Hoshen-Kopelman algorithm.

- In this algorithm, we will start with a partially filled lattice, pick a site randomly and define it to be occupied.

```
i = int((lattice_size-2)*rng())+1
j = int((lattice_size-2)*rng())+1
while n[i,j] > 0:
    i = int((lattice_size-2)*rng())+1
    j = int((lattice_size-2)*rng())+1
n[i,j]=1
```

- We will define a counter to store cluster labels called x. Each time a new site is added, the counter will increase by 1.

  These labels, associated with each site, are stored in a 10x10 matrix called d.

```
x = x+1
d[i,j] = x
```

  This label is called the proper label of the site and is a positive integer number.

- Now, to keep track of the matrix connectivity, we will define an array with 100 elements called clust, which will be updated based on the connectivity of the clusters.
  Let's assume that the new site that we added, connect two clusters called m1 and m2. We define clust(m1) and clust(m2) to point to the new cluster number x as follows, indicating that m1 and m2 clusters are now part of cluster x.

  clust(m1)=-x, clust(m2)= -x, clust(x)=x

  You can see that the first two clusters now have and additional label, while we have preserved its original labeling. The new cluster number x, will be the "proper" label of the new larger cluster.

  To find out if for a given site, its cluster label is "proper label" or one of the original labels, we will look at the cluster label, d[i,j]. In this case, the cluster label could be e.g. m1. Then we look at clust(m1). If clust(m1) is positive, m1 is a proper cluster label. If clust(m1) is negative, it means that m1 is now part of a larger cluster. To find it, we iteratively look at the other labels, e.g, clust(m1)=-m2, so we look for clust(m2), if this value is positive, it means that we have found the proper cluster label, if not, we will continue, clust(m2)=-m3, and look for clust(m3), until we find the proper label associated with that cluster.

  In the sample code that is provided, a matrix is generated. Once we add a new occupied site to the matrix, we add a new cluster label to the new site, and define the proper cluster label for that site to be x as well,

  ```
  d[i,j] = x
  clust[x]=x
  ```

- We then search among four neighbors of the new site to find the occupied sites. If any of these sites is occupied, e.g. n[i-1,j], we will look for its cluster number:

  ```
  ind = d[i-1,j]
  ```

  and determine if it is a proper cluster number or not.
  If clust[ind] > 0 it means that ind is the proper cluster number for this cluster. Otherwise, we find the cluster number by looking at

  ind=clust[-1*ind]

  and will continue this until we find the ind that is the proper cluster number of this neighbor (for which ind > 0).

Now, we need to update the information about this cluster. Now the proper cluster number for this cluster must change to the new value x.

```
Clust[ind] = -1 * x
```

Where x is a proper label, and we know that from the fact that clust[x]=x a positive number.

Now, run the script to create the matrix n and identify the clusters in your matrix. Print the matrix, as well as the connectivity matrix for your output. The connectivity matrix should show the proper cluster label for each point.

An example is shown below, n is written to the left, and on the right each point is labeled by its proper cluster label, showing that we have four clusters.

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{matrix} \qquad \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 3 \\ 2 & 2 & 0 & 3 \\ 0 & 0 & 4 & 0 \end{matrix}$$

For the second part of the lab, we will calculate the percolation probability of the clusters. You need to run the script for different values of p (occupancy probability) and determine whether you have percolation. To do so, once the matrix n is generated, you will run the script to label the clusters. Then, you will look at the edges, and see whether there are points on the four edges that have the same "proper" cluster number. If they do, you have a percolating network, if not your network is not percolated.

Calculate the percolation probability of the lattice as a function of lattice size (L=5, 10, 15) and plot it.

Please note that in the script, the edges of the cluster at i,j=0,lattice_size-1 are left empty intentionally. This is to simplify the search for nearest neighbors. So, the actual edges of the network are located at i,j=1,lattice_size-2.