Problem 5: Graph Magic

First, let us look at the case that the eigenvalues are symmetric and see what we can do with this. Let $\lambda_1, ..., \lambda_n$ be our eigenvalues. When we assume that the eigenvalues are symmetric, we see:

$$\sum_{i=1}^{n} \lambda_n = 0.$$

In fact, for k odd integer, we have:

$$\sum_{i=1}^{n} \lambda_n^k = 0.$$

Now we can relate this to the trace of the matrix. We know $Tr(A) = \sum_{i=1}^{n} \lambda_n = 0$ and also $Tr(A^k) = \sum_{i=1}^{n} \lambda_n^k = 0$.

But the trace is also the sum of the diagonal elements of the matrix, $Tr(A) = \sum_{i=1}^{n} a_{ii}$ and $Tr(A^k) = \sum_{i=1}^{n} a'_{ii}$, where a'_{ij} are the elements of the matrix A^k. We can interpret this as the sum of the number of walks from any vertex i to vertex i of length k. As the number of walks is ≥ 0, we see that for all odd k there are no walks from any vertex i to itself of length k. This is the same as the graph being bipartite. Thus if the eigenvalues are symmetric, our graph is bipartite.

Now what we still want to determine if being bipartite also implies that the eigenvalues are symmetric. To prove this, let A be the adjacency matrix of a bipartite graph. Then there exists some matrix B such that:

$$A = \begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix}.$$

Now if $(x, y)^T$ is an eigenvector with eigenvalue λ:

$$\begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \end{pmatrix},$$

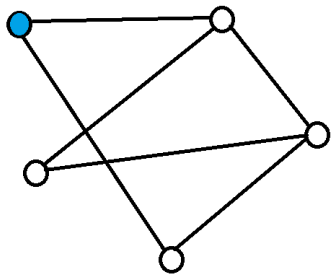
so $B^T x = \lambda y$ and $By = \lambda x$. We see:

$$\begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} x \\ -y \end{pmatrix} = \begin{pmatrix} -By \\ B^T x \end{pmatrix} = -\lambda \begin{pmatrix} x \\ -y \end{pmatrix}.$$

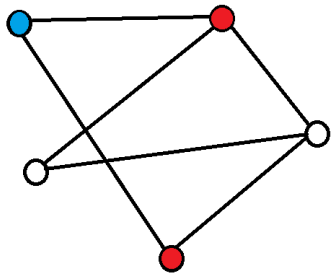
We can repeat this argument for every eigenvector and eigenvalue to conclude that we have symmetric eigenvalues. Thus we conclude that the eigenvalues of A are symmetric if and only if the graph is bipartite.

See next page for explanation of a possible algorithm for this.

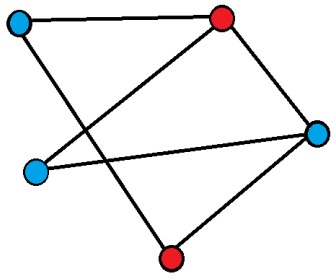
Our algorithm should determine whether a given graph is bipartite. We can do this by using the two-colour principle. If a graph is bipartite, we can color each vertex, such that there is no edge between two vertices of the same colour. One way to work this into an algorithm, is by starting at one vertex and giving that one color:



Then we colour every vertex that has an edge to this first one the other colour:



We continue this till every edge is coloured (or we encounter an edge with two vertices that have the same colour):



In this example case we see that we can thus conclude that the graph is not bipartite, as there is an edge between two blue vertices.