CS111 F21 Homework 8

Michael Glushchenko, 9403890
Partners with: Shiv Kapoor

1 Temperature Matrix to Laplacian

Here's the code we use for this question:

```
def nonZeroRowSums(A):
    res = [sum(A[i,:]) for i in range(A.shape[0])]
    res = [i for i in res if i != 0]
    return res
```

```
A = cs111.make_A(100)
rows = nonZeroRowSums(A.toarray())
print("number of rows that need to be changed =", len(rows))
```

```
number of rows that need to be changed = 396
```

In order for the temperature matrix to become a Laplacian, all of its row sums must be 0. We use the function above to see how many rows would have to be altered, minimum, in order to convert the temperature matrix into a Laplacian.

2 Graphs and Cuts

2.1 How Many Cuts?

Given the formula

$$xLx^T = \sum_{(i,j)\in E} (x(i) - x(j))^2$$

we can inspect what happens to the sum for every edge that crosses the cut. Now, the difference

$$x(i) - x(j) = 0 \text{ or } 2.$$

It is 0 if the i and j are on the same side of the cut, and it is 2 if the edge connecting i and j crosses the cut.

Let's say i and j have an edge between them. Then

$$x(i) - x(j) = 2, \text{ and } x(j) - x(i) = 2.$$

This 2 is then squared through the

$$(x(i) - x(j))^2$$

expression. As a result, the edge the crosses the cut produces a value of 8 through the summation instead of the necessary value of 2.
This means that each cut is counted $\frac{8}{2} = 4$ times. Since

$$xLx^T = \sum_{(i,j)\in E} (x(i) - x(j))^2,$$

we can conclude that the number of edges crossing a cut in any graph is

$$\frac{xLx^T}{4},$$

meaning that our value α is

$$\alpha = \frac{1}{4}$$

2.2 Geometric Cut

We use the following code to produce the geometric cut and find the number of edges that cross the cut, and to confirm that we split points evenly into red and blue (note that I use list comprehension to make my code slightly more concise, since we have to take screenshots of it):

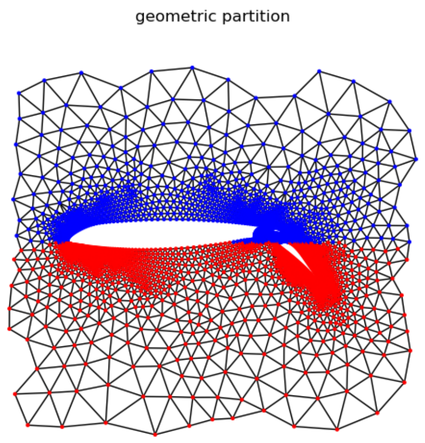
```
# load the airfoil graph
G, xycoords = cs111.read_mesh('airfoil1')
# calculate the median
new_xy = [xycoords[i] for i in range(len(xycoords))]
median_coordinate = np.median(new_xy, axis = 0)[1] # median of the y coordinate
```

```
# plot
first_colors = ['r' if xycoords[v][1] < median_coordinate else 'b' for v in G.nodes()]
plt.ion()
plt.figure(figsize=(6,6))
plt.title('geometric partition')
plt.axis('equal')
nx.draw(G, pos = xycoords, node_size = 4, node_shape = 'o', node_color = first_colors)
```

```
# calculate vector x based on the red/blue coloring
my_list = [1 if first_colors[i] == 'r' else -1 for i in range(len(first_colors))]
x = np.array(my_list)
# make the laplacian of G
L = nx.linalg.laplacian_matrix(G).toarray()
print("num of edges that cross the cut = ", x @ L @ x.T * 1/4)
print("num of red points =" , first_colors.count('r'))
print("num of blue points =" , first_colors.count('b'))
```

```
num of edges that cross the cut =  148.0
num of red points = 2126
num of blue points = 2127
```

As a result, this is the chart that's produced:



2.3 Spectral Cut

Using similar logic as in **2.2**, but now using the Fiedler vector to divide points into red and blue groups; here's the code used for this question:

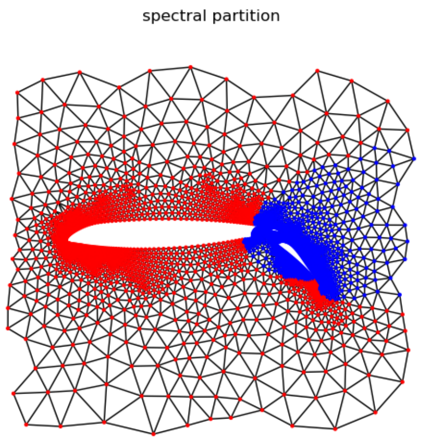
```
# make the laplacian of G
L = nx.linalg.laplacian_matrix(G).toarray()
# eigenvectors of the laplacian L
lam, W = spla.eigh(L)
# finding the fiedler vector of L
fiedler_vector = np.transpose(W)[1]
# calculate the median of fiedler_vector
median_fiedler_vec_value = np.median(fiedler_vector)
```

```
# now we label each element with the corresponding element
# of the Fiedler vector and color it based on whether that
# label is smaller or larger than the median value we just computed
second_colors = ['r' if fiedler_vector[v] < median_fiedler_vec_value else 'b' for v in G.nodes()]
plt.ion()
plt.figure(figsize=(6,6))
plt.title('spectral partition')
plt.axis('equal')
nx.draw(G, pos = xycoords, node_size = 4, node_shape = 'o', node_color = second_colors)
```

```
# calculate vector x based on the red/blue coloring
my_list = [1 if second_colors[i] == 'r' else -1 for i in range(len(second_colors))]
x = np.array(my_list)
# how many edges cross the cut?
print("num of edges that cross the cut =", x @ L @ x.T * 1/4)
print("num of red points =", my_list.count(1))
print("num of blue points =", my_list.count(-1))
```

```
num of edges that cross the cut = 132.0
num of red points = 2126
num of blue points = 2127
```

As a result, here's the drawing produced by **2.3**:


spectral partition