*April 6th, 2019*

**Name:** Amanda Forde
**Student ID No.:** 16308191
**Email:** *a.forde21@nuigalway.ie*

# CS319 SCIENTIFIC COMPUTING PROJECT REPORT

**Introduction:**

This project consisted of creating an algorithm to simulate a particular network, accompanied by the implementation of an algorithm for extracting information from the network. The network which I was required to reproduce was one referred to as a cycle with shortcuts. It is a cycle on `N` vertices in which `K` extra edges have been added at random. I then tested the 'Friendship Paradox' on this network. I have outlined below a brief synopsis of the steps that were taken in order to achieve this.

**Cyclic Network with Shortcuts:**

In order to simulate the network, a sparse matrix class was created, referred to as '`Duplet`', which can in fact be used to store the edges of any undirected graph of a reasonable size. `Duplet` is designed in a very similar fashion to `Triplet` class [i]. It has two arrays, `I[k]` and `J[k]`, which are associated with the row and column values of the network's adjacency matrix. However, it does not require an array storing the non-zero values, as when applied to a network, these are merely assumed to be 1. For example, if `[1] [2]` is stored in `Duplet`, it indicates that there is an edge between vertex 1 and vertex 2 and hence, in the adjacency matrix of the network the value of $A_{1,2}$ and $A_{2,1}$ is 1.

In order to save on space in the `Duplet` class and as the graph was undirected, it was decided to only store each edge once. For all `k`, `I[k]` < `J[k]`, corresponding directly to the upper half of the adjacency matrix. This allows for the manufacture of larger networks with more vertices.

Indeed, the construction of the cyclic network was made significantly easier by using the `Duplet` class. For an `N`-node cyclic graph with `K` extra edges, `Duplet` is then considered to be of size `N+k`, that is, arrays `I[k]` and `J[k]` are both of length `N+k`. `N` and `K` are both specified by the user. The inclusion of input checking was vital as `N` must be greater than `3` to ensure a cyclic network and `K` is clearly required to be less than the maximum number of extra edges that could possibly be added.

The algorithm first creates what can be thought of as the mere template of the cyclic graph. It inserts the outer edges, i.e. all edges between `i` and `i-1` for `i` from 1 to `N`, using the function `setij(i,j)` which is a member of `Duplet` class. `0` and `N-1` is also included to guarantee a closed cyclic graph.

Then, `K` edges are added at random. Existence of an edge is checked using `whereij(i,j)`, another member function of `Duplet`. It returns `-1` if the edge is not already stored. It is also ensured that no node loops to itself. An extract of the algorithm demonstrating this is included.

The result of a call to srand(time(NULL)) is used as the seed of the random number generator so that the value of the seed changes with time. Thus, every time the programme is used, a new set of random numbers are generated. In other words, if the same values for N and k are entered at independent times when the programme is run, a different set of edges should still be obtained.

The network is deemed to be completely formed when K edges have been added. This is equivalent to both arrays I[k] and J[k], belonging to the Duplet class, being full.

```
Duplet D(N, N+k);

D.setij(0,N-1);
for (unsigned int i=1; i<N; i++)
   D.setij(i,i-1);

unsigned int edges = 0;
while (edges < k)
  {
     unsigned int i = rand()%N;
     unsigned int j = rand()%N;
     if ( i != j && D.whereij(i,j)== -1 )
       {
          D.setij(i,j);
          edges++;
       }
   }
```

*Cyclic_Friendship.cpp*

**The Friendship Paradox:**

In 1991, the sociologist Scott L. Feld  The idea of the 'Friendship Paradox' was when he found that on average, most people have fewer friends than their friends have.[ii]

Representing friendship through a network using graphs allows us to explore this idea in a mathematical and computational manner. In this project, an algorithm was implemented which tested the 'Friendship Paradox' on a cyclic network, one which was stored in a class referred to as 'Duplet'. A function, named HowMany(Duplet D, vertex v), was defined. This function returns the number of friends a particular vertex, v, has when provided with the vertex and corresponding network, stored as a Duplet. It performed this easily by counting the number of instances that 'v' occurred in Duplet. In graph theory, this is equivalent to determining the degree of the vertex 'v'. The definition of this function proved extremely useful as the number of friends and the number of friends of friends of each individual vertex had to be established.

```
unsigned int HowMany(Duplet D, unsigned int v)
{
  unsigned int friends = 0;
   for (unsigned int counter = 0; counter < D.nnz_max(); counter++)
   {
       if (D.getI(counter) == v)
         {friends++;}
       else if (D.getJ(counter) == v)
         {friends++;}
   }

   return(friends);
}
```

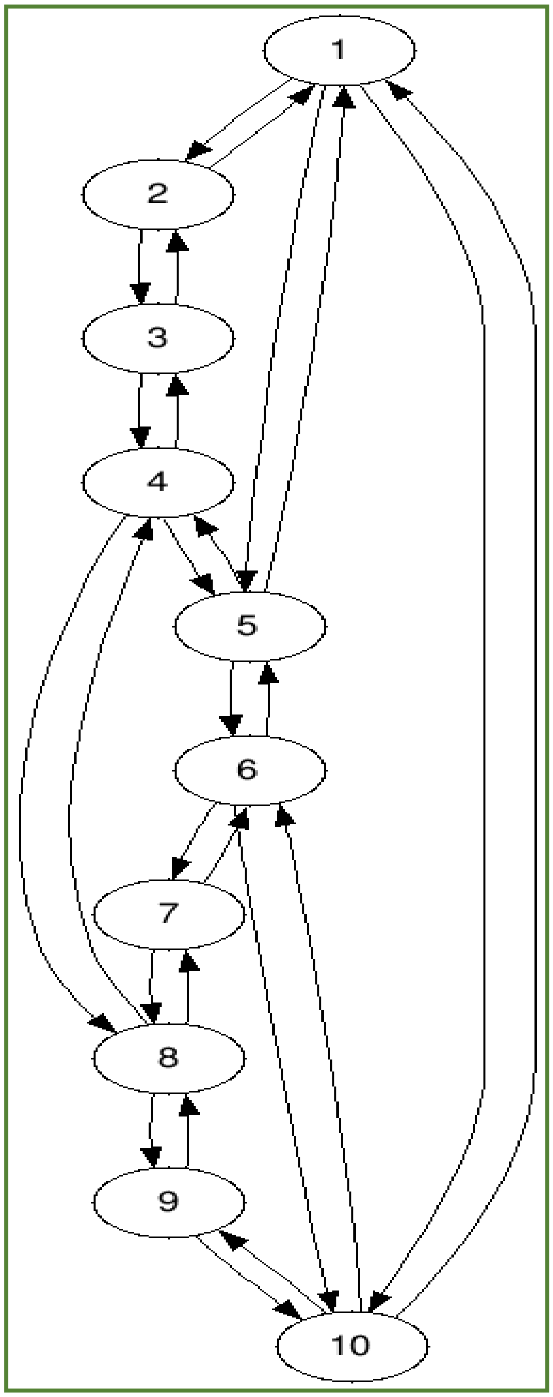
*Cyclic_Friendship.cpp*

In order to accomplish the former, a Vector F [1] was created for each vertex which stored all of 'v''s friends. This Vector was simply iterated through and HowMany(D,v) function was implemented on each entry in the Vector to find the total number of friends of friends of 'v'.

This resulted in the computation of the average number of friends of an average person in the network as: $\frac{total\ no.\ of\ friends}{N}$

and the average number of friends of average friends of an average person as: $\frac{total\ no.\ of\ friends\ of\ friends}{total\ no.\ of\ friends}$ .

Implementing the algorithm many times provided a strong indication that the Friendship Paradox is in fact always true. No matter how many vertices or edges were added to the cyclic network, the second value was consistently greater or equal to the first. Equality was achieved when all vertices had the same number of edges. For example, if no new edges were added then both values were found to be equal to `2` or in the case of a complete graph, when the maximum number of edges were added, both values were equal to `N-1`.



*Obtained using `http://graphs.grevian.org/graph`*

**Algorithms in Practice:**

This network shown has been created using the algorithm detailed above. For simplicity of demonstration, N = 10 and k = 3 were chosen. As seen from the diagram, it is evidently a cycle in which the 3 extra edges were randomly chosen to be: [4][8], [6][10] and [1][5]. The nodes have been indexed from 1 to 10.

As the programme allows for the potential creation of a CSV file containing the networks adjacency matrix, this option was chosen. It was then possible to use `Adj2Dot.cpp` [iii] to convert the CSV file to a dot(GraphViz) file. This could then be copied to `http://graphs.grevian.org/graph` in order to view the network. The result obtained can be viewed right.

When the 'Friendship Paradox' was implemented it computed the average number of friends of an average person to be `2.6` and the average number of friends of average friends of an average person to be `2.69231`. The first value is less than the second, as expected.

**Limitations and Additional Comments:**

- For `N=k=10,000`, the programme takes just under a minute to run. As N increases and k increases, it does tend to slow down as one would imagine.

- Furthermore, the `Duplet` class created specifically relates to an undirected graph. This is particularly due to how the functions; `whereij(i,j)` and `setij(i,j)`, have been defined. It is worth noting that alteration of these definitions isn't particularly difficult if one wishes to utilize this class to store a directed graph.
- A `C++ set` could also have been used for containing the friends of an individual vertex. However, as the `Duplet` class requires the `Vector` class in order to work effectively and the fact that I was well acquainted with this class, it was decided that using this `Vector` class would be the best option.
- `Cyclic_Friendship.cpp` includes the demonstration of how the adjacency matrix of a network can be created from a `Duplet` storing that network. The user has been given the option to create a CSV file with this adjacency matrix. An advantage of this is that one can view a cyclic network created using `Adj2Dot.cpp`, which converts an adjacency matrix stored in a CSV file into a dot (GraphViz) file. This can then be uploaded to to `http://graphs.grevian.org/graph` to obtain an image similar to that above.
- In order for `Cyclic_Friendship.cpp` to compile, one requires the following files along with `Duplet.cpp and Duplet.h: Vector09.h, Vector09.cpp, Matrix09.h, Matrix09.cpp.` [iv]

**Conclusion:**

This project successfully exhibits a method of creating and storing a cyclic network with shortcuts in `C++`, using the creation of a new class. Member functions belonging to the class, assisted by another function, were employed to extract information from the network. This information was then used to test the Friendship Paradox efficiently.

[i] See *http://www.maths.nuigalway.ie/~niall/CS319/Week10/* for information on `Triplet.cpp`.

[ii] Feld, Scott L. (1991), *"Why your friends have more friends than you do", American Journal of Sociology*, ***96*** *(6): 1464–1477, doi:10.1086/229693, JSTOR 2781907*.

[iii] See *http://www.maths.nuigalway.ie/~niall/CS319/lab7/* for more details regarding `Adj2Dot.cpp`.

[iv] See *http://www.maths.nuigalway.ie/~niall/CS319/Week09/* for the following files: `Vector09.h, Vector09.cpp, Matrix09.h, Matrix09.cpp.`