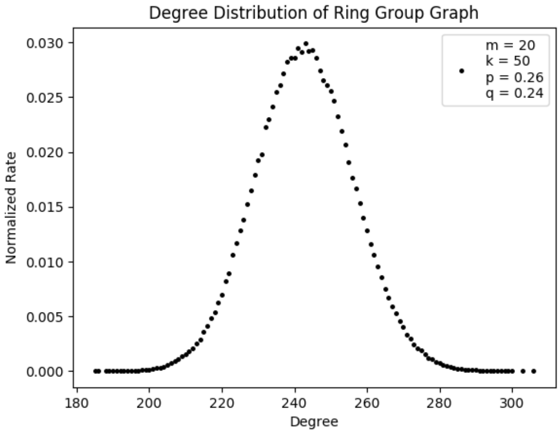
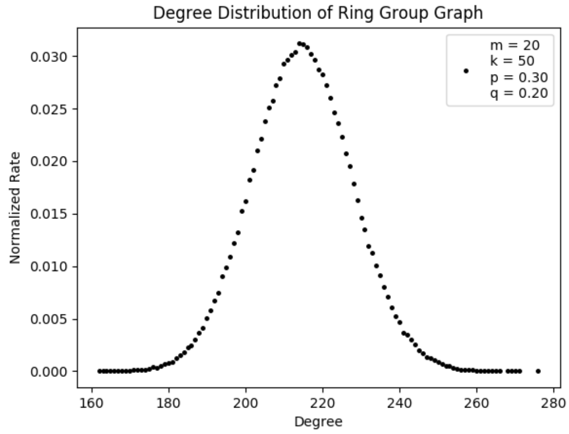
Network Analysis Assignment

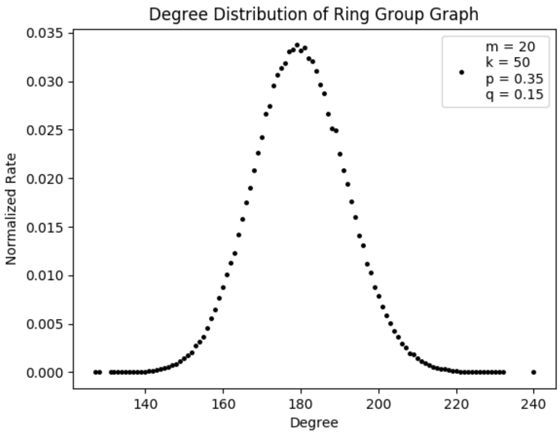
kdkj55

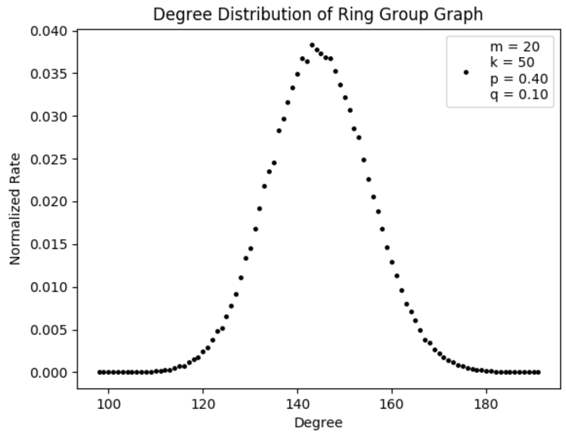
Question 1 [30 marks]

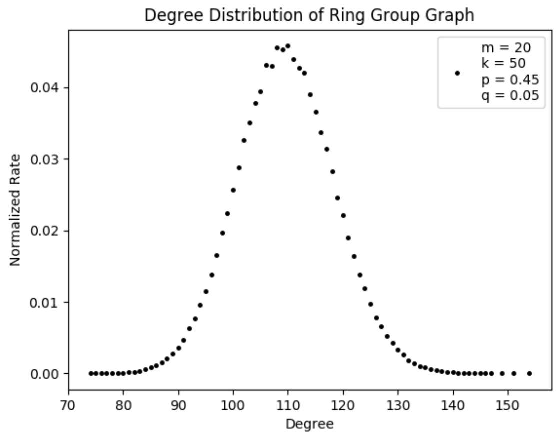
- Investigate the degree distribution of Ring Group Graphs for $p + q = 0.5$, $p > q$.

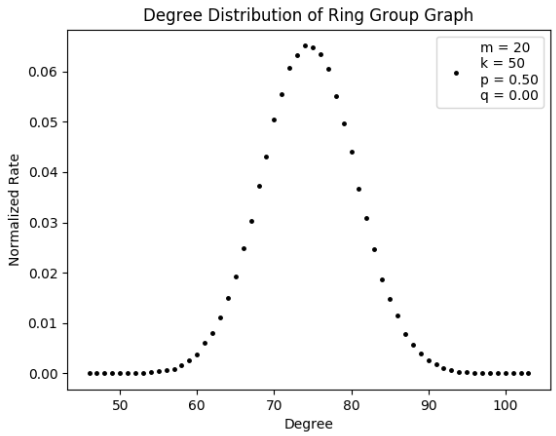








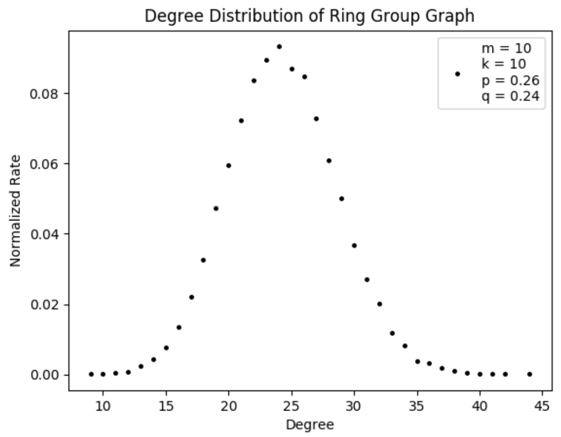


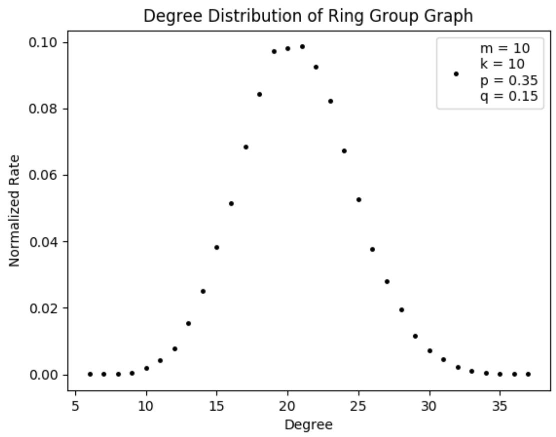


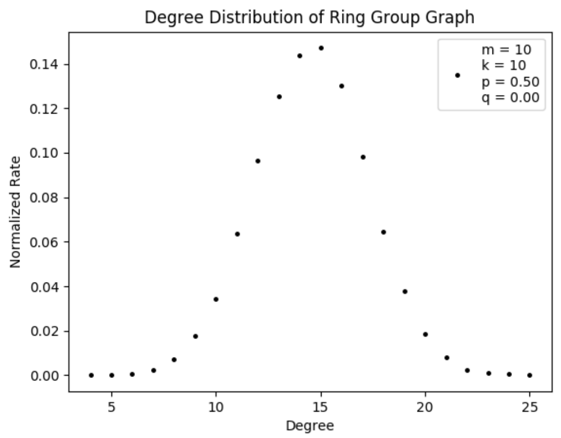
With the values $m=20$ and $k=50$, we get the results shown above. Clearly, the shape of the distribution is consistent for varying values of p and q. However, the mean value and variance of the distributions do change when we alter these parameters; specifically, as p is increased (and q is decreased), we see that the mean degree in the graphs decreases. This is to be expected, as p is the probability of an edge between vertices in adjacent groups, and q is the probability of an edge between vertices in non-adjacent groups. Given that the majority of groups are non-adjacent to each other, it was obvious that decreasing q would decrease the average degree of the vertices. We also see that the standard deviation of the graphs decreases as q is decreased. This too is expected, and it's simple to explain why when we consider the extreme cases. With *p=0.50* and *q=0.00*, edges can only exist between vertices from adjacent groups. This means that the degree of any given vertex must be between 0 and 150 (a degree of 150 in this case meaning that the vertex is connected to every other vertex in its group, as well as the two adjacent groups). When we increase *q*, we introduce the probability of a vertex being connected to any of the 1000 vertices in the graph. Clearly, this will increase the variance in degree distributions.

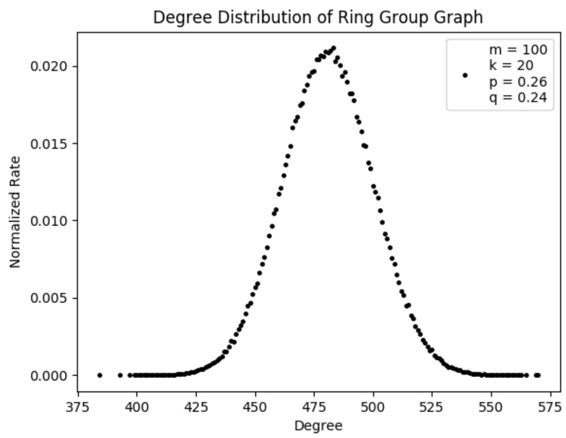
- You should report on [...] whether the same effects are found for different values of *m* and *k*.

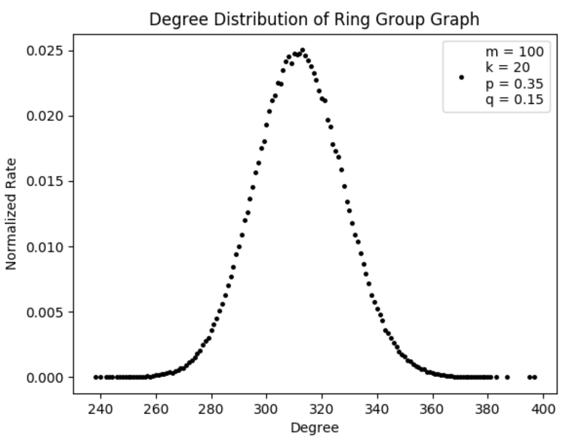
To conserve space, I will only show the plots for *p=0.26*, *p=0.35* and *p=0.50* for each graph. The plots above are for *m=20* and *k=50*, and I will now choose *m=10, k=10* and *m=100, k=20*. These graphs should be diverse enough to test if the relationship identified above holds for different ring group graphs. The results below clearly indicate that the relationships between mean value, variance and the parameters *p* and *q* do in fact hold for different values of *m* and *k*.

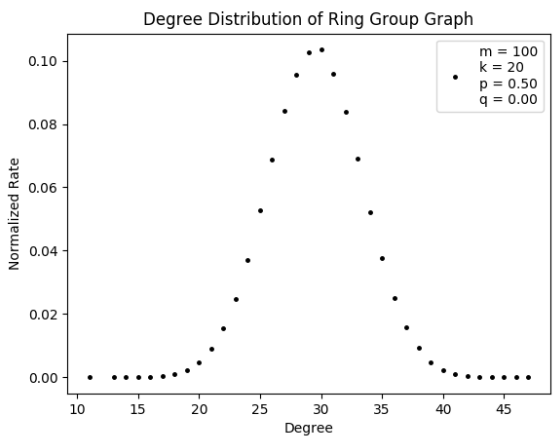






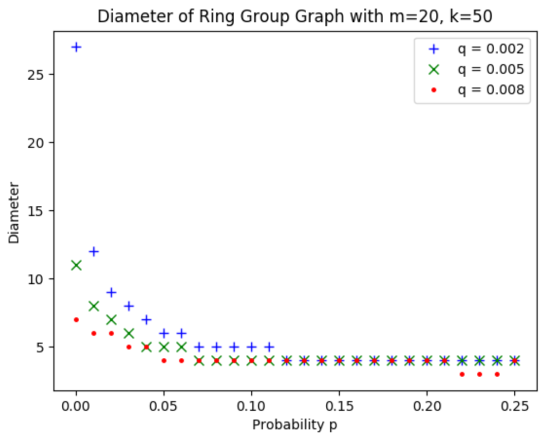


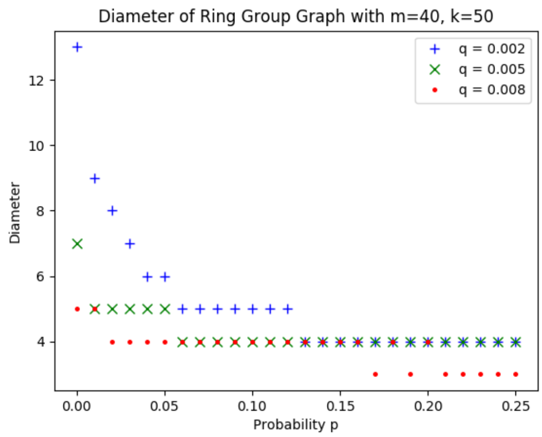




- Investigate the relationship between the diameter of Ring Group Graphs and p (for fixed q, $p > q$)

The relationship here is that the diameter of the graph is inversely proportional to the value of p. I have produced results for different values of q, on two different sizes of graph; the relationship is the same in all cases.





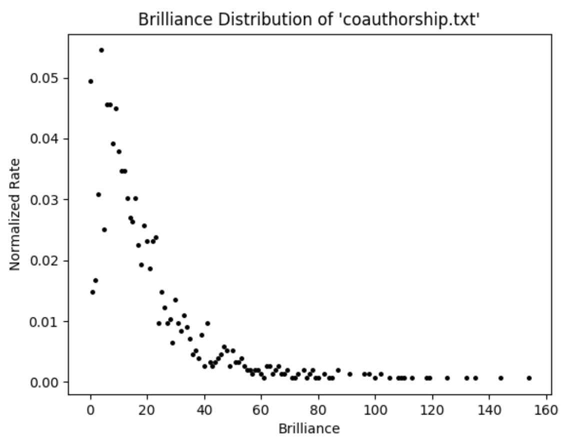
Question 2 [30 marks]

- For the graph from coauthorship.txt, investigate the distribution of vertex brilliance.

To calculate the brilliance of a vertex, I initially tried to calculate it exactly; that is, check every subset of neighbour nodes to find the one that gives the largest independent set. However, when the graph becomes too big, the run-time of this approach becomes exceedingly long. The method I ended up with is as follows. For each vertex:

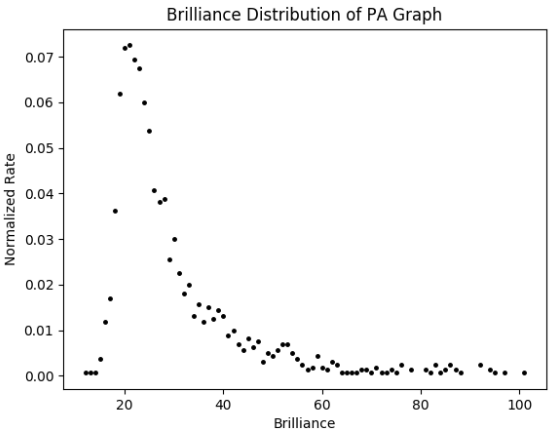
- Create the subgraph consisting of only the neighbours of the vertex in question, with all edges between these neighbours in place. From this, we can tell which neighbours are immediately independent (those with no neighbours in the subgraph).
- If the number of neighbours - the number of ones that are already independent is less than 20, then calculate the exact brilliance as outlined above. Otherwise, approximate it as follows:
- From the subgraph created earlier, repeatedly take the node with the smallest non-zero degree, and find all of its neighbours. Remove these neighbours from the graph, as well as all of their edges. Keep doing this until all nodes have a degree of zero, resulting in an independent set. The number of nodes remaining is an approximation of the brilliance of the original vertex.

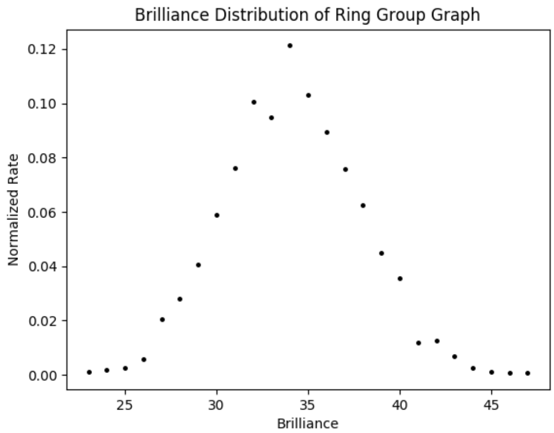
Running the coauthorship graph through this algorithm produces the distribution below.



The distribution of brilliance in this graph displays a wide variety. Whilst the most common brilliance is around 10-15, some vertices have brilliances as high as 155 (although this is not at all common).

- For [PA Graphs and Ring Group Graphs] create examples with approximately the same number of vertices and edges as the graph from coauthorship.txt and investigate the distribution of vertex brilliance. Comment on what you find.





The graph from coauthorship.txt has 1559 nodes and 44661 edges. To get approximately the same number of vertices / edges, I used the following parameters:

- PA graphs - 1600 nodes, and a degree parameter of 30. An example graph constructed in the way had 45533 edges, which is close enough.
- Ring group graphs - the parameters *m=16*, *k=100*, *p=0.06*, *q=0.03*. This also gives 1600 nodes, and an example graph had 45396 edges.

The brilliance distributions produced are shown in the figures above.

For the PA graphs, the distribution is similar in shape to that of the coauthorship graph. However, the PA graphs have the peak in their distribution at a higher brilliance, and yet seem to have a smaller variance. The highest brilliance for PA graphs is only about 100 where as the highest brilliance for the coauthorship graph is around 155.
The brilliance distribution of ring group graphs is completely different to the other two graph types. The shape more closely resembles a binomial or a normal distribution, the most common brilliance appears to be slightly greater at about 34, and the variance is much smaller - no vertex has a brilliance below 23 or above 47.

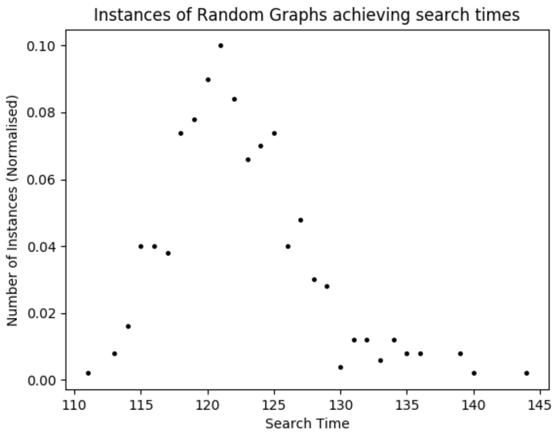
Question 3 [40 marks]

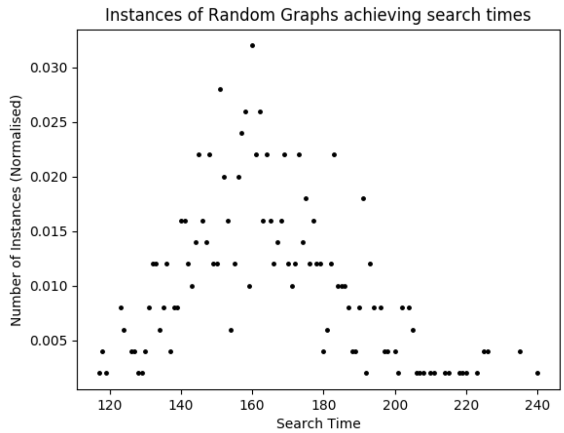
- For [Random Graphs and Ring Group Graphs], describe an algorithm for searching in the graphs. You should explain why you believe your strategy might be effective and implement and test it on many instances. Plot search time against the number of instances that achieve that time. Comment on your plots. Credit will be given for the effectiveness of the algorithm you design, and also, independently, for your explanation of the rationale behind the design.

<u>Random Graphs</u>

For random graphs, the simplest search algorithm is to query a random neighbour of the current vertex and immediately move to it, repeating until the target vertex is found. As we have no additional heuristics to use in this search, we have no way of knowing if the search is moving closer to the target or not. Therefore, it does not make sense to waste 'time' querying the neighbours, and we should move at random until the target is found. I considered an alternative algorithm, in which we check the degree of each vertex visited, and if it exceeds a certain threshold then query all of the neighbours in search of the target. The theory here was that better connected nodes are more likely to be adjacent to the target vertex, so it is worth querying the neighbours. However, this seemed to be increasing the

search time rather than decreasing it. If we reach a vertex that is well connected, but not adjacent to the target, we immediately accrue a large search time from querying the many neighbour nodes. Afterwards, we still have no indication of which move will decrease the distance to the target and are forced to move randomly regardless.
The random search first described produces the figure on the left graph, with the figure on the right a result of the other algorithm discussed. Note that the random graphs used have 100 vertices, with each pair of vertices having an edge with probability 0.1.



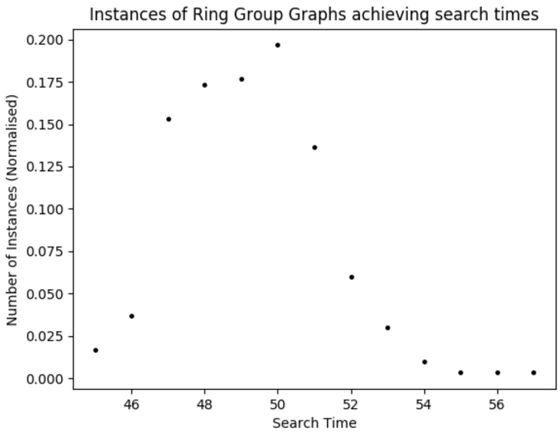


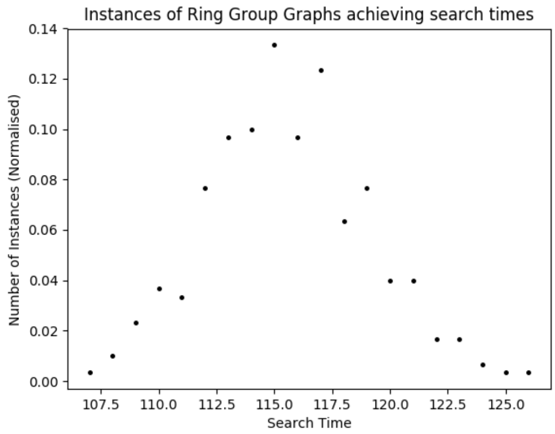
<u>Ring Group Graphs</u>
My search algorithm for these graphs is as follows:
Note that when I discuss groups 'adjacent' to the target group, the target group itself is included.

- At each vertex, query the neighbours until either of the following conditions are met, in which case, move immediately to that vertex, or there are no neighbours left:
  - The target vertex is found.
  - A vertex in a group adjacent the target group is found, if the current vertex is not already in such a group.
- If all of the neighbours have been queried and the above conditions have not been met, then:
  - If the current vertex is already adjacent to the target group, the search moves to a random neighbour that also satisfies these conditions, if possible.
  - Otherwise, the search moves to the neighbour vertex in the group closest to the target group, breaking ties randomly.

This algorithm aims to move the search closer to the target's group. It does so, because the probability of a vertex being connected to the target is greater when in an adjacent group. Once it has reached such a group, it moves randomly until the target is found. Unlike with the random search, we query all of the neighbours before moving, to reduce the risk of moving further away from the target. This

The results achieved in my Ring Group Graph search are displayed in the left figure below. The right hand figure is for comparison, and shows the results attained when using the random search outlined above. The results attained by a random search serve to highlight the efficiency of the algorithm I produced. Note that these search times were attained on a graph with the parameters *m=k=10*, *p=0.3*, *q=0.05*.


Instances of Ring Group Graphs achieving search times
Number of Instances (Normalised)
Search Time
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
46
48
50
52
54
56


Instances of Ring Group Graphs achieving search times
Number of Instances (Normalised)
Search Time
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
107.5
110.0
112.5
115.0
117.5
120.0
122.5
125.0