17 Combinatorics

17.1 Graph Colouring (7 units)

This project is based on the material found in the Part II Graph Theory course.

In this project you will need to be able to generate graphs from $\mathcal{G}(n,p)$ and $\mathcal{G}_k(n,p)$. The space $\mathcal{G}(n,p)$ is that of graphs with n labelled vertices, edges appearing independently and at random with probability p. The space $\mathcal{G}_k(n,p)$ differs from $\mathcal{G}(n,p)$ only in that ij is never an edge if $i - j \equiv 0 \pmod{k}$.

1 A simple colouring algorithm

A *colouring* of a graph G is an assignment of a colour to each vertex of G, so that no two adjacent vertices receive the same colour. The *chromatic number* of G, denoted by $\chi(G)$, is the smallest number of colours for which it is possible to produce a colouring. It is believed that finding the chromatic number of a graph G is, in general, very hard. At no stage in this project are you required to implement a procedure for the exact evaluation of $\chi(G)$. You will, however, develop procedures for finding upper and lower bounds for $\chi(G)$.

The *greedy algorithm* colours a graph whose vertex set is *ordered* by colouring vertices one at a time in the order given, using colours from $\{1, 2, 3, \ldots\}$. The colour chosen for a vertex is the least colour from among those not already assigned to any previously coloured neighbours.

Question 1 Write a procedure which applies the greedy algorithm to a graph with a given ordering of the vertices. Test your program on ten members of $\mathcal{G}(70, 0.5)$, and compare the number of colours used when the vertices are ordered in the following ways: (i) by increasing degree, (ii) by decreasing degree, (iii) where v_j has minimum degree in the graph $G - \{v_{j+1}, \ldots, v_n\}$, (iv) at random.

Do the same for $\mathcal{G}_3(70, 0.75)$.

Question 2 What ordering will guarantee that the greedy algorithm uses no more than 3 colours for $\mathcal{G}_3(70, 0.75)$? Why do you think the probability 0.75 was chosen here? For each n give an example of a graph G of order $3n$ such that $\chi(G) = 3$ but on which greedy might need $n + 2$ colours.

2 Cliques

A *clique* in a graph G is a complete subgraph of largest order in G. (This definition differs from some in the literature.) Notice that $\chi(G)$ is at least as large as the order of a clique.

A greedy-type algorithm for finding a complete subgraph in G would start with a subgraph of order one (a vertex) and repeatedly try to find a vertex joined to all vertices of the subgraph selected so far, until no further such vertex could be found.

Question 3 Give an argument to suggest that it is unlikely the greedy-type algorithm will find a complete subgraph of order 14 in a graph from $\mathcal{G}(2000, 0.5)$. How large do you think a clique is likely to be in a graph from $\mathcal{G}(2000, 0.5)$?

Question 4 Write a procedure to find a clique in a graph G. [Note: this procedure may be time-consuming but should not be excessively so on the examples here.] Compare, for several graphs, the resulting lower bound you get on $\chi(G)$ with the upper bounds obtained previously.

3 Colouring

An *independent set* in a graph is a subset of the vertex set which spans no edges. A colouring is thus just a partition of the vertices into independent sets.

Question 5 Convert your clique procedure to find an independent set of maximum order in a graph. Hence write a procedure to colour a graph by the following method. First find a largest independent set I_1. Then find a largest independent set I_2 in $G - I_1$, then I_3 in $G - I_1 - I_2$, and so on until nothing remains. Compare the upper bounds on $\chi(G)$ so obtained with previous bounds. Try your program on ten members of $G_7(70, 0.5)$ also. Is there a change in behaviour as p is increased, say from 0.4 to 0.6? If your program can handle larger graphs in a finite time, obtain further data of interest.

None of the above methods for bounding $\chi(G)$ is guaranteed to find $\chi(G)$ exactly.

Question 6 Estimate (crudely) the theoretical running times of all the algorithms used above as functions of n when the input is a typical member of $\mathcal{G}(n, 0.5)$. Describe in outline, but do not implement, a procedure for colouring a graph with exactly $\chi(G)$ colours, and estimate its running time.

References

[1] Bollobas, B., Modern Graph Theory, Springer 1998.

