17 Combinatorics

17.7 Graph planarity (10 units)

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

A *planar* graph is one which can be drawn in the plane without edge crossings. Kuratowski's theorem gives a theoretical characterisation of planar graphs. However, it does not provide a practical algorithm for testing whether a graph is planar, and finding such an algorithm is not easy. This project describes one such algorithm. The focus is on simplicity rather than efficiency, and the process is broken into several separate steps. Nonetheless, the total amount of coding required for this project is still comparatively large.

Planar graphs are sparse, so graphs in the project are specified by means of an *edge list*: this is just a list of edges, each edge being a pair of vertices. We may assume that the vertices are labelled by integers. Note that the edge list does not identify isolated vertices: these don't affect planarity and you might as well assume that the number n of vertices is the largest integer label. It is often convenient to have an *adjacency list*, which is a collection of n lists, the j^{th} list giving the neighbours of vertex j. You will find it useful to write a routine to derive an adjacency list from an edge list.

Even though a planarity testing algorithm will tell whether a graph is planar, it might not supply a way to actually draw the graph in the plane. Finding such a drawing is a separate problem, and we begin with that.

1 Graph drawing

A beautiful theorem of Tutte gives a way to draw a 3-connected planar graph G in the plane. Let C be a cycle with exactly one bridge (as defined in the next section). If C has k vertices, place these at the corners of a convex k-gon. Place the remaining vertices so that each is at the centroid of its neighbours: that is, if u has d neighbours placed at positions $\mathbf{x}_1, \ldots, \mathbf{x}_d$ in the plane, then u has position $\mathbf{x} = (1/d)\sum_{i=1}^{d} \mathbf{x}_i$. Finding these positions requires solving a system of linear equations. Tutte proved that, under the stated conditions, there is a unique solution, and that, by adding straight line segments between adjacent vertices, we obtain a planar drawing in which each face is a convex polygon.

Question 1 Write a program that, given a graph and a cycle C, draws the graph with C at the vertices of a regular polygon. (You need not check that G is 3-connected or that C has one bridge.)

Give your output for each of the five Platonic solids, whose edge lists can be found at `http://www.damtp.cam.ac.uk/user/catam/data/Platonic_x.txt`, where x is one of `4`, `6`, `8`, `12` or `20` (representing the Tetrahedron, Cube, Octahedron, Dodecahedron and Icosahedron respectively). In each case you can take the vertices of the outer face C to be those labelled $1, 2, \ldots, k$ for the appropriate k.

We define the graph K_2 with vertex set $\{1, 2\}$ and single edge 1–2, and the graph P_5 with vertex set $\{3, 4, 5, 6, 7\}$ and edge set $\{3\text{–}4, 4\text{–}5, 5\text{–}6, 6\text{–}7\}$. From these, we define the graph $K_2 + P_5$ with vertex set $\{1, 2, 3, 4, 5, 6, 7\}$, and edge set consisting of all edges of K_2, all edges of P_5, and all possible edges between a vertex of K_2 and a vertex of P_5 (thus $K_2 + P_5$ is a 'complete bipartite union' of K_2 and P_5). Draw the graph $K_2 + P_5$, using $1, 2, 3$ as the outer face.

2 Bridges and components

Nowadays, a bridge is usually defined to be the same as an isthmus, that is, an edge whose removal increases the number of components. Tutte used the word quite differently but we keep his terminology, though with a different meaning.

Given a graph G and a cycle C in it, a *bridge* of C is a non-empty set of edges defined as follows: it is the edges of a component of $G[V(G) \setminus V(C)]$, together with any edges joining that component to C. A chord of C is also defined to be a bridge of C, having a single edge. Therefore the bridges partition $E(G) \setminus E(C)$.

Notice that isolated vertices of G do not feature in any bridge. However, an isolated vertex of $G[V(G) \setminus V(C)]$ which is joined to C by some edges will give rise to a bridge consisting of just those edges. Hence a bridge with one edge is either a chord of C, or it is an edge joining a vertex of C to a vertex of degree one outside C, or it is an edge joining two vertices of degree one both outside C.

The *vertices of attachment* of a bridge are the vertices of C which are end vertices of edges in the bridge. So a bridge might have no vertices of attachment (if it is the edges of a component of $G[V(G) \setminus V(C)]$ not joined to C), or it might have one or more.

Note that bridges can meet each other, but only at vertices of attachment.

> **Question 2** Write a program to find the components of a graph. (For example, pick a vertex, find its neighbours, then their neighbours, and so on.)
>
> Write a further program to find the bridges of a given cycle C in a graph, together with their vertices of attachment.

3 Interleaving

Two bridges B' and B'' of the cycle C are said to *interleave* if there are four distinct vertices $a, b, c, d \in V(C)$, appearing in that order on the cycle (but not necessarily adjacent), such that a and c are vertices of attachment of B', and b and d are vertices of attachment of B''. Additionally, B' and B'' are also said to interleave if they both have exactly three vertices of attachment, these three vertices being the same for B' as for B''. Notice that this entire definition is symmetric in B', B''.

If C has ℓ bridges $B_1, \ldots, B_\ell$, then the *interleave graph* H has ℓ vertices $h_1, \ldots, h_\ell$, with $h_i h_j \in E(H)$ if and only if B_i and B_j interleave.

> **Question 3** Suppose G is a graph with a cycle C that has ℓ bridges $B_1, \ldots, B_\ell$. Explain why G is planar if and only if the following holds: each of the subgraphs with edges $E(C) \cup B_i$, $1 \leqslant i \leqslant \ell$, is planar, and the interleave graph H is bipartite.

> **Question 4** Write a program to construct the interleave graph from a cycle C and its bridges. Write a program to test whether a graph is bipartite.

4 The core of a graph

Suppose G has a vertex v of degree one. We can remove the edge at v without affecting planarity. Likewise if G has a vertex v of degree two, with edges uv and vw, we can remove these two

edges, adding the edge uw if it is not already present. This does not affect planarity either. Repeating such operations as much as possible, we arrive at the *core* G^* of G, in which each vertex has degree zero or at least three. (Possibly G^* has no edges.) The labels of the vertices in G^* can depend on the order in which the operations are done but, other than that, G^* is determined by G. Hence we need not worry about the order of the operations.

Question 5 Write a program to find the core G^* of a graph G.

Question 6 Describe a procedure for finding a cycle in a graph of minimum degree at least two.

Describe a procedure for finding a cycle with a chord in a graph of minimum degree at least three.

Write a program to find a cycle with a chord in a non-empty core G^*.

5 A planarity algorithm

Here is a recursive algorithm for testing whether a graph G is planar.

```
Find the core G* of G.
If G* is empty, G is planar.
Else find a cycle C in G* with a chord e.
Find the bridges of C in G* and the interleave graph H.
If H is not bipartite then G is not planar.
Else G is planar if and only if G* − e is planar.
```

Question 7 Explain why this algorithm works correctly.

Question 8 Write a program to determine whether a graph is planar.

Test your program on various examples. You might use the graphs in Question 1, and the same graphs with one or two edges removed or added.

Question 9 As a further test, write a program to build a random maximal planar graph with n vertices, by starting with the empty graph, and testing each of the $\binom{n}{2}$ possible edges in a random order: if the addition of an edge maintains planarity, keep it in the graph, but if it violates planarity, throw it away. For extra interest, your program should tell you how many edges were added before the first violation.

How many edges should your graph have at the end?

Generate 20 random maximal planar graphs with 40 vertices. In each case, show how many edges were added before the first violation.

Draw one of the graphs, using your program from Question 1.

Question 10 Estimate the complexity of the planarity algorithm from Question 8.

References

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

