Exercises on embedded graphs, Lecture 4

Arnaud de Mesmay

October 9, 2025

Exercise 1: Let G be a plane connected graph, where every face is given with a real weight. Let T be a spanning tree of G (a subgraph of G with the same vertex set as G that is a tree). Give a linear-time algorithm that computes for all edges e of G not lying in T, the sum of the weights of all the faces inside the cycle formed by e and the unique path in T joining its endpoints.

Solution: As in the algorithm to compute minimum spanning trees, recall that the co-tree T^* is a subgraph of G^* made of the edges not in T, and it is a tree. Each leaf ℓ of T^* therefore corresponds to a face which is completely enclosed by edges of T plus the edge e dual to the edge incident to ℓ. So for these edges e, the sum that we are looking for is exactly weight of the corresponding leaf in T^*.

For other nodes of T^*, we can compute the sum of the weights inductively: we root T^* in the outer face, set $f(\ell) = w(\ell)$ for each leaf of T^* and define, for any non-leaf node in T^*

$$f(v) = \sum_{c \text{ children of } v} f(c).$$

The entire map f can be computed and stored in an array f in linear time by starting from the leaves and moving up rootwards: instead of a recursion the formula just looks up at previously recorded values[1]

$$f[v] = \sum_{c \text{ children of } v} f[c].$$

Exercise 2: Consider a rectangular grid. In each square in the grid, there is either a diagonal $\diagup$ or an anti-diagonal $\diagdown$. The diagonals connect to the vertices of the rectangular grid, inducing a plane graph, as in (most of) the picture below. Prove that there always exists at least one path from the left to the right side of the grid, or from the top to the bottom side of the grid.

Solution: Let us denote by G the graph where we have put both the $\diagup$ and the $\diagdown$ in each cell of the grid. The graph G is not planar, but it is actually made of two disjoint subgraphs G_1 and G_2: a vertex (i, j) such that $i + j$ is odd is never connected to a vertex (i', j') such that $i' + j'$ is even (since $\diagup$ and $\diagdown$ both preserve that parity). Moreover, both G_1 and G_2 are planar, and G_2 is almost the graph dual to G_1: they are dual except near the boundary, where G_1^* has a single vertex corresponding to the outer face while G_2 has many vertices spread along the boundary.

Now, let us denote by H the plane graph from the exercise, which is naturally split into $H_1 \subseteq G_1$ and $H_2 \subseteq G_2$. We enhance G_1 and H_1 into G_1' and H_1' by adding a vertex s connected

[1] This is the premise of *dynamic programming*, which can arguably be summarized as just replacing parentheses with brackets: we store the values of f in order to not recompute them each time we need them.

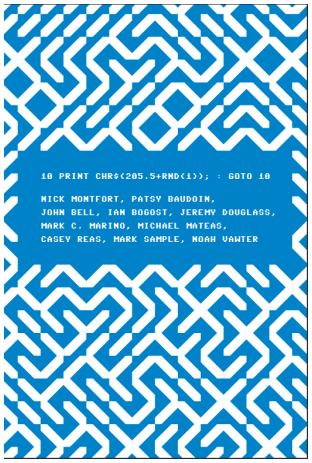

Figure 1: An entire book written about these mazes, available here.

to all the vertices on the left side of G_1 and H_1, and likewise we add a vertex t connected to all the vertices on the right side of G_1. If there is an s-t path in H_1', the goal of the exercise is achieved. If there is not, since such an s-t path exists in G_1, it means that $C := G_1' \setminus H_1'$ is an s-t cut. By cut-cycle duality, there is therefore a cycle C^* separating s^* and t^* in $G_1'^*$ made of the edges dual to those in C. But as we observed earlier, the edges dual to $G_1 \setminus H_1$ are in G_2, and are even in H_2: for any $\diagdown$ not present in H_1, $\diagup$ is present in H_2, and for any $\diagup$ not present in H_1, $\diagdown$ is in H_2. So the cycle C^* intersects the rectangular grid in edges of H.

Now C^* is a closed curve that separates s and t and does not intersect the edges E' of G_1' connecting s and t respectively to the left and the right side of the grid. We conclude by showing that any such closed curve must contain a subpath connecting the top to the bottom on the grid from within. If C^* does not intersect the top side of the grid, then it cannot separate s from t (this is topological: $\mathbb{R}^2 \setminus (E' \cup$ top side of the grid) is homemomorphic to a disk with s and t on its boundary, and no cycle in a disk can separate two points on the boundary). Likewise for the bottom side. So C^* intersects both sides. Now, if there is no path in C^* connecting the top side to the bottom side inside the grid, then any subpath entering the grid from one side, say the bottom one, must exit it from the same side. Such "bigons" can be rerouted so as to not intersect the side at all while still separating s and t, as in the min-cut algorithm. Iteratively, we would obtain a rerouting of C^* not intersecting the bottom side, which is a contradiction[2].

So C^* contains a subpath connecting the top and the bottom side inside the grid, where it consists of edges of H_2 and thus of H. This concludes the proof.

Exercise 3: Let G_1 and G_2 be two connected plane graphs.

1. Give a quadratic-time algorithm deciding whether G_1 and G_2 are equivalent, i.e., whether there exists an oriented homeomorphism of $\mathbb{R}^2$ sending G_1 to G_2.

2. Deduce a quadratic-time algorithm to decide whether two 3-connected planar graphs are isomorphic.

3. (Hard) Can you deduce a quadratic-time algorithm to decide whether two 2-connected planar graphs are isomorphic?

[2]At this stage of the proof, we need not specify whether C^* and its rerouting lives on H_2 or anywhere else: we showed that *any* closed curve C^* not intersecting the top and the bottom sides and E' cannot separate s and t, and obtain a contradiction with that.

4. (Not that hard if you assume you know how to do the 2-connected case) Can you deduce a quadratic-time algorithm deciding whether two planar graphs G_1 and G_2 are isomorphic?

If you are curious, this can actually be done in linear time, as first proved here.

Hint: If you want to tackle this exercise at home, you might want to look up block-cut trees for question 4 (which are quite intuitive), and SPQR trees for question 3 (which are quite a bit less intuitive).