2.5 Compact straight-line drawings

As a next step we consider plane embeddings in the geometric setting, where every edge is drawn as a straight-line segment. A classical theorem of Wagner and Fáry states that this is not a restriction in terms of plane embeddability.

Theorem 2.31 (Fáry [6], Wagner [21]) *Every planar graph has a plane straight-line embedding (that is, it is isomorphic to a plane straight-line graph).*

Although this theorem has a nice inductive proof, we will not prove it here. Instead we will prove a stronger statement that implies Theorem 2.31.

A very nice property of straight-line embeddings is that they are easy to represent: We need to store points/coordinates for the vertices only. From an algorithmic and complexity point of view the space needed by such a representation is important, because it appears in the input and output size of algorithms that work on embedded graphs. While the Fáry-Wagner Theorem guarantees the existence of a plane straight-line embedding for every planar graph, it does not provide bounds on the size of the coordinates used in the representation. But the following strengthening provides such bounds, by describing an algorithm that embeds (without crossings) a given planar graph on a linear size integer grid.

Theorem 2.32 (de Fraysseix, Pach, Pollack [8]) *Every planar graph on $n \geqslant 3$ vertices has a plane straight-line drawing on the $(2n-3) \times (n-1)$ integer grid.*

Canonical orderings. The key concept behind the algorithm is the notion of a canonical ordering, which is a vertex order that allows to construct a plane drawing in a natural (hence canonical) way. Reading it backwards one may think of a shelling or peeling order that destructs the graph vertex by vertex from the outside. A canonical ordering also provides a succinct representation for the combinatorial embedding.

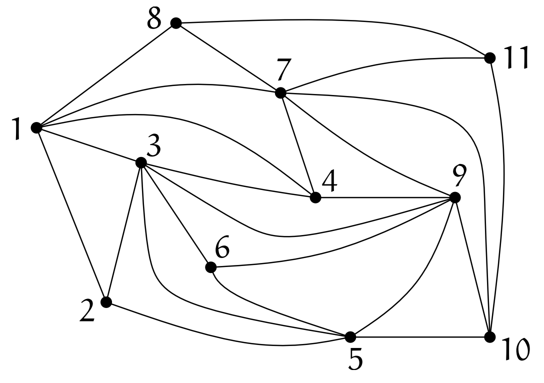
Definition 2.33 *A plane graph is* **internally triangulated** *if it is biconnected and every bounded face is a (topological) triangle. Let G be an internally triangulated plane graph and $C_\circ(G)$ its outer cycle. A permutation $\pi = (v_1, v_2, \ldots, v_n)$ of $V(G)$ is a* **canonical ordering** *for G, if*

(1) G_k is internally triangulated, for all $k \in \{3, \ldots, n\}$;

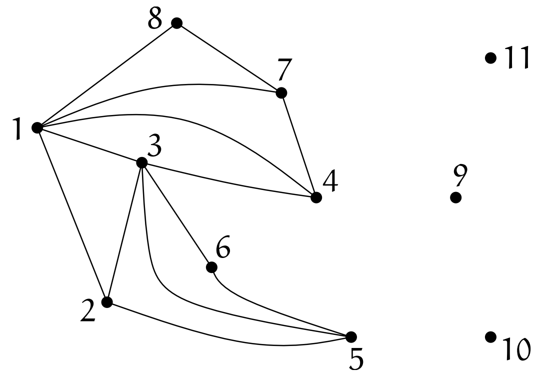
(2) v_1v_2 is on the outer cycle $C_\circ(G_k)$ of G_k, for all $k \in \{3, \ldots, n\}$;

(3) v_{k+1} is located in the outer face of G_k and its neighbors appear consecutively along $C_\circ(G_k)$, for all $k \in \{3, \ldots, n-1\}$;

where G_k is the subgraph of G induced by $v_1, \ldots, v_k$.



(a) G.

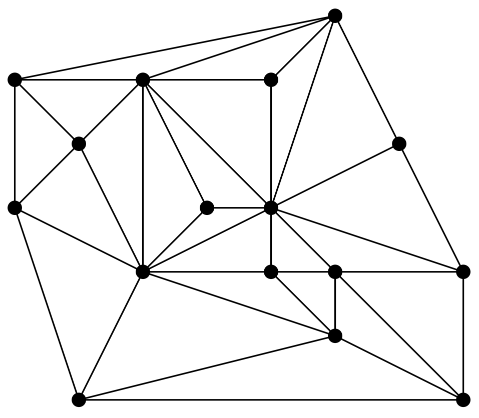


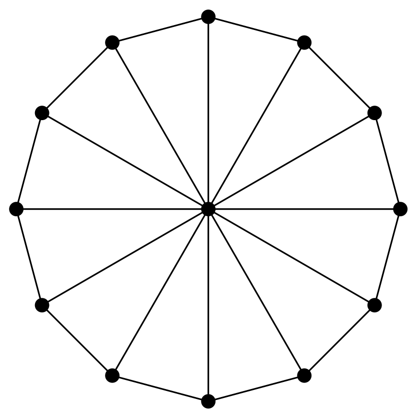
(b) G_8.

Figure 2.18: *An internally triangulated plane graph and a canonical order for it.*

Figure 2.18 shows an example. Note that there are permutations that do not correspond to a canonical order: for instance, when choosing the vertex 4 as the eighth vertex instead of 8 in Figure 2.18b the graph G_8 is not biconnected (1 is a cut-vertex).

Exercise 2.34 *(a) Compute a canonical ordering for the following internally triangulated plane graphs:*





(b) Give a family of internally triangulated plane graphs G_n on $n = 2k$ vertices with at least $(n/2)!$ canonical orderings.

Exercise 2.35 *(a) Describe a plane graph G with n vertices that can be embedded (while preserving the outer face) on a grid of size $(2n/3 - 1) \times (2n/3 - 1)$ but not on a smaller grid.*

(b) Can you draw G on a smaller grid if you are allowed to change the embedding?

Theorem 2.36 *For every internally triangulated plane graph* G *and every edge* $\{v_1, v_2\}$ *on its outer face, there exists a canonical ordering for* G *that starts with* v_1, v_2. *Moreover, such an ordering can be computed in linear time.*

Proof. Induction on n, the number of vertices. For a triangle, any order suffices and so the statement holds. Hence consider an internally triangulated plane graph $G = (V, E)$ on $n \geqslant 4$ vertices. We claim that it is enough to select a vertex $v_n \notin \{v_1, v_2\}$ on $C_\circ(G)$ that is not incident to a chord of $C_\circ(G)$.

First observe that G is plane and $v_n \in C_\circ(G)$ and so all neighbors of v_n in G must appear on the outer face of $G_{n-1} = G \setminus \{v_n\}$. Consider the circular sequence of neighbors around v_n in G and break it into a linear sequence $u_1, \dots, u_m$, for some $m \geqslant 2$, that starts and ends with the neighbors of v_n in $C_\circ(G)$. As G is internally triangulated, each of the bounded faces spanned by v_n, u_i, u_{i+1}, for $i \in \{i, \dots, m-1\}$, is a triangle and hence $\{u_i, u_{i+1}\} \in E$. This implies *(3)* for $k = n$. Properties *(1)* and *(2)* hold trivially (by assumption) in that case. In order to complete the ordering inductively we need to show that G_{n-1} is also internally triangulated.

As G_{n-1} is a subgraph of G, which is internally triangulated, it suffices to show that G_{n-1} is biconnected. The outer cycle $C_\circ(G_{n-1})$ of G_{n-1} is obtained from $C_\circ(G)$ by removing v_n and replacing it with the (possibly empty) sequence $u_2, \dots, u_{m-1}$. As v_n is not incident to a chord of $C_\circ(G)$ (and so neither of $u_2, \dots, u_{m-1}$ appeared along $C_\circ(G)$ already), the resulting sequence forms a cycle, indeed. Add a new vertex v in the outer face of G_{n-1} and connect v to every vertex of $C_\circ(G_{n-1})$ to obtain a maximal planar graph $H \supset G_{n-1}$. By Theorem 2.26 H is 3-connected and so G_{n-1} is biconnected, as desired. This also completes the proof of the initial claim.

It remains to show that we can always find such a vertex $v_n \notin \{v_1, v_2\}$ on $C_\circ(G)$ that is not incident to a chord of $C_\circ(G)$. If $C_\circ(G)$ does not have any chord, this is obvious, because every cycle has at least three vertices, one of which is neither v_1 nor v_2. So suppose that $C_\circ(G)$ has a chord c. The endpoints of c split $C_\circ(G)$ into two paths, one of which does not have v_1 nor v_2 as an internal vertex. Among all possible chords of $C_\circ(G)$ select c such that this path has minimal length. (It has always at least two edges, because there is always at least one vertex "behind" a chord.) Then by definition of c this path is an induced path in G and none of its (at least one) interior vertices is incident to a chord of $C_\circ(G)$, because such a chord would cross c. So we can select v_n from these vertices. By the way the path is selected with respect to c, this procedure does not select v_1 nor v_2.

Regarding the runtime bound, we maintain the following information for each vertex v: whether it has been chosen already, whether it is on the outer face of the current graph, and the number of incident chords with respect to the current outer cycle. Given a combinatorial embedding of G, it is straighforward to initialize this information in linear time. (Every edge is considered at most twice, once for each endpoint on the outer face.)

When removing a vertex, there are two cases: Either v_n has two neighbors u_1 and u_2 only (Figure 2.19a), in which case the edge u_1u_2 ceases to be a chord. Thus, the chord count for u_1 and u_2 has to be decremented by one. Otherwise, there are $m \geqslant 3$ neighbors $u_1, \ldots, n_m$ (Figure 2.19b) and (1) all vertices $u_2, \ldots, u_{m-1}$ are new on the outer cycle, and (2) every edge incident to u_i, for $i \in \{2, \ldots, k-1\}$, and some other vertex on the outer cycle other than u_{i-1} or u_{i+1} is a new chord (and the corresponding counters at the endpoints have to by incremented by one).

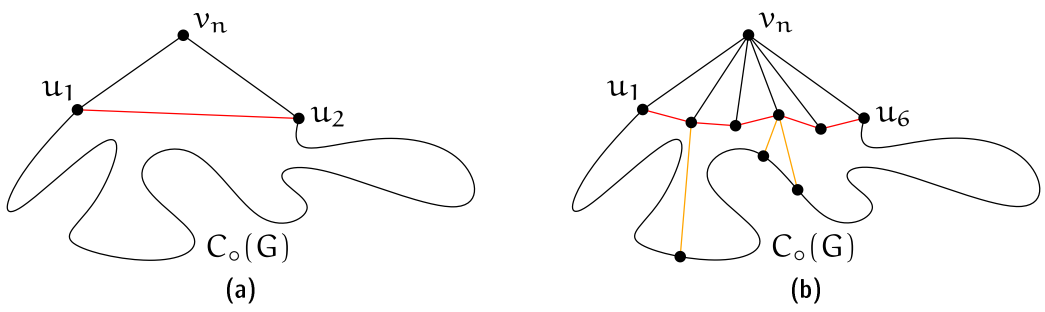


Figure 2.19: *Processing a vertex when computing a canonical ordering.*

During the course of the algorithm every vertex appears once as a new vertex on the outer face. At this point all incident edges are examined. Overall, every edge is inspected at most twice—once for each endpoint—which takes linear time by Corollary 2.5. □

Using one of the linear time planarity testing algorithms, we can obtain a combinatorial embedding for a given maximal planar graph G. As every maximal plane graph is internally triangulated, we can then use Theorem 2.36 to provide us with a canonical ordering for G, in overall linear time.

Corollary 2.37 *Every maximal planar graph admits a canonical ordering. Moreover, such an ordering can be computed in linear time.* □

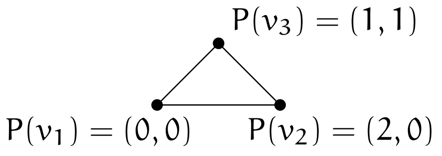
As simple as they may appear, canonical orderings are a powerful and versatile tool to work with plane graphs. As an example, consider the following partitioning theorem.

Theorem 2.38 (Schnyder [18]) *For every maximal planar graph G on at least three vertices and every face f of G, the multigraph obtained from G by doubling the (three) edges of f can be partitioned into three spanning trees.*

Exercise 2.39 *Prove Theorem 2.38.* Hint: *Take a canonical ordering and build one tree by taking for every vertex v_k the edge to its first neighbor on the outer cycle $C_\circ(G_{k-1})$.*

Of a similar flavor is the following open problem, for which only partial answers for specific types of point sets are known [2].

Problem 2.40 (In memoriam Ferran Hurtado (1951–2014))
Can every complete geometric graph on $n = 2k$ vertices (the complete straight line graph on a set of n points in general position) be partitioned into k plane spanning trees?



The shift-algorithm. Let $(v_1, \dots, v_n)$ be a canonical ordering. The general plan is to construct an embedding by inserting vertices in this order, starting from the triangle $P(v_1) = (0, 0)$, $P(v_3) = (1, 1)$, $P(v_2) = (2, 0)$. At each step, some vertices will be shifted to the right, making room for the edges to the freshly inserted vertex. For each vertex v_i already embedded, maintain a set $L(v_i)$ of vertices that move rigidly together with v_i. Initially, $L(v_i) = \{v_i\}$, for $1 \leqslant i \leqslant 3$.

Ensure that the following invariants hold after Step k (that is, after v_k has been inserted):

(i) $P(v_1) = (0, 0)$, $P(v_2) = (2k - 4, 0)$;

(ii) The x-coordinates of the points on $C_\circ(G_k) = (w_1, \dots, w_t)$ are strictly increasing (in this order)[4];

(iii) each edge of $C_\circ(G_k)$ is drawn as a straight-line segment with slope ± 1.

Clearly these invariants hold for G_3, embedded as described above. Invariant (i) implies that after Step n we have $P(v_2) = (2n - 4, 0)$, while (iii) implies that the Manhattan distance[5] between any two points on $C_\circ(G_k)$ is even.

Idea: put v_{k+1} at $\mu(w_p, w_q)$, where $w_p, \dots, w_q$ are its neighbors on $C_\circ(G_k)$ (recall that they appear consecutively along $C_\circ(G_k)$ by definition of a canonical ordering), where

$$\mu((x_p, y_p), (x_q, y_q)) = \frac{1}{2}(x_p - y_p + x_q + y_q, -x_p + y_p + x_q + y_q)$$

is the point of intersection between the line $\ell_1 : y = x - x_p + y_p$ of slope 1 through $w_p = (x_p, y_p)$ and the line $\ell_2 : y = x_q - x + y_q$ of slope -1 through $w_q = (x_q, y_q)$.

Proposition 2.41 *If the Manhattan distance between w_p and w_q is even, then $\mu(w_p, w_q)$ is on the integer grid.*

Proof. By Invariant (ii) we know that $x_p < x_q$. Suppose without loss of generality that $y_p \leqslant y_q$. The Manhattan distance d of w_p and w_q is $x_q - x_p + y_q - y_p$, which by assumption is an even number. Adding the even number $2x_p$ to d yields the even number $x_q + x_p + y_q - y_p$, half of which is the x-coordinate of $\mu((x_p, y_p), (x_q, y_q))$. Adding the

[4]The notation is a bit sloppy here because both t and the w_i in general depend on k. So in principle we should write w_i^k instead of w_i. But as the k would just make a constant appearance throughout, we omit it to avoid index clutter.

[5]The *Manhattan distance* of two points (x_1, y_1) and (x_2, y_2) is $|x_2 - x_1| + |y_2 - y_1|$.

even number $2y_p$ to d yields the even number $x_q - x_p + y_q + y_p$, half of which is the y-coordinate of $\mu((x_p, y_p), (x_q, y_q))$. □

After Step n we have $P(v_n) = (n-2, n-2)$, because v_n is a neighbor of both v_1 and v_2. However, $P(v_{k+1})$ may not "see" all of $w_p, \dots, w_q$, in case that the slope of $w_p w_{p+1}$ is 1 and/or the slope of $w_{q-1} w_q$ is -1 (Figure 2.20).

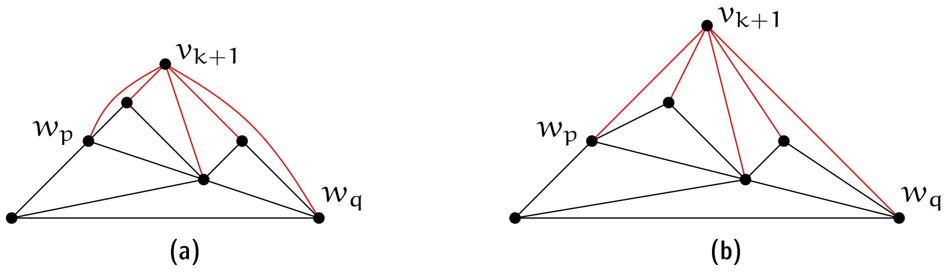


Figure 2.20: *(a) The new vertex v_{k+1} is adjacent to all of $w_p, \dots, w_q$. If we place v_{k+1} at $\mu(w_p, w_q)$, then some edges may overlap, in case that w_{p+1} lies on the line of slope 1 through w_p or w_{q-1} lies on the line of slope -1 through w_q; (b) shifting $w_{p+1}, \dots, w_{q-1}$ by one and $w_q, \dots, w_t$ by two units to the right solves the problem.*

In order to resolve these problems we shift some points around so that after the shift w_{p+1} does not lie on the line of slope 1 through w_p and w_{q-1} does not lie on the line of slope -1 through w_q. The process of inserting v_{k+1} then looks as follows.

1. Shift $\bigcup_{i=p+1}^{q-1} L(w_i)$ to the right by one unit.
2. Shift $\bigcup_{i=q}^{t} L(w_i)$ to the right by two units.
3. $P(v_{k+1}) := \mu(w_p, w_q)$.
4. $L(v_{k+1}) := \{v_k\} \cup \bigcup_{i=p+1}^{q-1} L(w_i)$.

Observe that the Manhattan distance between w_p and w_q remains even, because the shift increases their x-difference by two and leaves the y-coordinates unchanged. Therefore by Proposition 2.41 the vertex v_{k+1} is embedded on the integer grid.

The slopes of the edges $w_p w_{p+1}$ and $w_{q-1} w_q$ (might be just a single edge, in case that $p+1 = q$) become < 1 in absolute value, whereas the slopes of all other edges along the outer cycle remain ± 1. As all edges from v_{k+1} to $w_{p+1}, \dots, w_{q-1}$ have slope > 1 in absolute value, and the edges $v_{k+1} w_p$ and $v_{k+1} w_q$ have slope ± 1, each edge $v_{k+1} w_i$, for $i \in \{p, \dots, q\}$ intersects the outer cycle in exactly one point, which is w_i. In other words, adding all edges from v_{k+1} to its neighbors in G_k as straight-line segments results in a plane drawing.

Next we argue that the invariants (i)–(iii) are maintained. For (i) note that we start shifting with w_{p+1} only so that even in case that v_1 is a neighbor of v_{k+1}, it is never shifted. On the other hand, v_2 is always shifted by two, because we shift every vertex starting from (and including) w_q. Clearly both the shifts and the insertion of v_{k+1} maintain the strict order along the outer cycle, and so (ii) continues to hold. Finally, regarding (iii) note that the edges $w_p w_{p+1}$ and $w_{q-1} w_q$ (possibly this is just a single edge) are the only edges on the outer cycle whose slope is changed by the shift. But these edges do not appear on $C_\circ(G_{k+1})$ anymore. The two edges $v_{k+1}w_p$ and $v_{k+1}w_q$ incident to the new vertex v_{k+1} that appear on $C_\circ(G_{k+1})$ have slope 1 and -1, respectively. So all of (i)–(iii) are invariants of the algorithm, indeed.

So far we have argued about the shift with respect to vertices on the outer cycle of G_k only. To complete the proof of Theorem 2.32 it remains to show that the drawing remains plane under shifts also in its interior part.

Lemma 2.42 *Let G_k, $3 \leqslant k \leqslant n$, be straight-line grid embedded as described, $C_\circ(G_k) = (w_1, \dots, w_t)$, and let $\delta_1 \leqslant \dots \leqslant \delta_t$ be non-negative integers. If for each i, we shift $L(w_i)$ by δ_i to the right, then the resulting straight-line drawing is plane.*

Proof. Induction on k. For G_3 this is obvious. Let $v_k = w_\ell$, for some $1 < \ell < t$. Construct a delta sequence Δ for G_{k-1} as follows. If v_k has only two neighbors in G_k, then $C_\circ(G_{k-1}) = (w_1, \dots, w_{\ell-1}, w_{\ell+1}, \dots, w_t)$ and we set $\Delta = \delta_1, \dots, \delta_{\ell-1}, \delta_{\ell+1}, \dots, \delta_t$. Otherwise, $C_\circ(G_{k-1}) = (w_1, \dots, w_{\ell-1} = u_1, \dots, u_m = w_{\ell+1}, \dots, w_t)$, where $u_1, \dots, u_m$ are the $m \geqslant 3$ neighbors of v_k in G_k. In this case we set

$$\Delta = \delta_1, \dots, \delta_{\ell-1}, \underbrace{\delta_\ell, \dots, \delta_\ell}_{m \text{ times}}, \delta_{\ell+1}, \dots, \delta_t \,.$$

Clearly, Δ is monotonely increasing and by the inductive assumption a correspondingly shifted drawing of G_{k-1} is plane. When adding v_k and its incident edges back, the drawing remains plane: All vertices $u_2, \dots, u_{m-1}$ (possibly none) move rigidly with (by exactly the same amount as) v_k by construction. Stretching the edges of the chain $w_{\ell-1}, w_\ell, w_{\ell+1}$ by moving $w_{\ell-1}$ to the left and/or $w_{\ell+1}$ to the right cannot create any crossings. □

Linear time. *(This part was not covered in the lecture.)* The challenge in implementing the shift algorithm efficiently lies in the eponymous shift operations, which modify the x-coordinates of potentially many vertices. In fact, it is not hard to see that a naive implementation—which keeps track of all coordinates explicitly—may use quadratic time. De Fraysseix et al. described an implmentation of the shift algorithm that uses $O(n \log n)$ time. Then Chrobak and Payne [5] observed how to improve the runtime to linear, using the following ideas.

Recall that $P(v_{k+1}) = (x_{k+1}, y_{k+1})$, where

$$\begin{aligned} x_{k+1} &= \frac{1}{2}(x_p - y_p + x_q + y_q) \text{ and} \\ y_{k+1} &= \frac{1}{2}(-x_p + y_p + x_q + y_q) = \frac{1}{2}((x_q - x_p) + y_p + y_q)\,. \end{aligned} \tag{2.43}$$

Thus,

$$x_{k+1} - x_p = \frac{1}{2}((x_q - x_p) + y_q - y_p). \tag{2.44}$$

$\Rightarrow$ We need the y-coordinates of w_p and w_q together with the relative x-position (offset) of w_p and w_q only in to determine the y-coordinate of v_{k+1} and its offset to w_p.

Maintain the outer cycle as a rooted binary tree T, with root v_1. For each node v of T, the *left child* is the first vertex covered by insertion of v (if any), that is, w_{p+1} in the terminology from above (if $p+1 \neq q$), whereas the *right child* of v is the next node along the outer cycle (if any; either along the current outer cycle or along the one at the point where both points were covered together). See Figure 2.21 for an example.

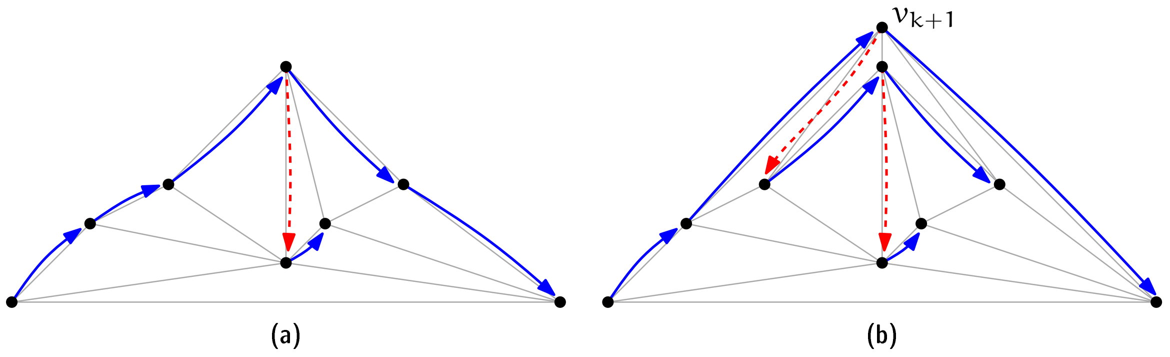


Figure 2.21: *Maintaining the binary tree representation when inserting a new vertex v_{k+1}. Red (dashed) arrows point to left children, blue (solid) arrows point to right children.*

At each node v of T we also store its x-offset $dx(v)$ with respect to the parent node. For the root v_1 of the tree set $dx(v_1) = 0$. In this way, a whole subtree (and, thus, a whole set $L(\cdot)$) can be shifted by changing a single offset entry at its root.

Initially, $dx(v_1) = 0$, $dx(v_2) = dx(v_3) = 1$, $y(v_1) = y(v_2) = 0$, $y(v_3) = 1$, $\mathrm{left}(v_1) = \mathrm{left}(v_2) = \mathrm{left}(v_3) = 0$, $\mathrm{right}(v_1) = v_3$, $\mathrm{right}(v_2) = 0$, and $\mathrm{right}(v_3) = v_2$.

Inserting a vertex v_{k+1} works as follows. As before, let $w_1, \ldots, w_t$ denote the vertices on the outer cycle $C_\circ(G_k)$ and $w_p, \ldots, w_q$ be the neighbors of v_{k+1}.

1. Increment $dx(w_{p+1})$ and $dx(w_q)$ by one. *(This implements the shift.)*

2. Compute $\Delta_{pq} = \sum_{i=p+1}^{q} dx(w_i)$. *(This is the total offset between w_p and w_q.)*

3. Set $dx(v_k) \leftarrow \frac{1}{2}(\Delta_{pq} + y(w_q) - y(w_p))$ and $y(v_k) \leftarrow \frac{1}{2}(\Delta_{pq} + y(w_q) + y(w_p))$. *(This is exactly what we derived in (2.43) and (2.44).)*

4. Set $\mathrm{right}(w_p) \leftarrow v_k$ and $\mathrm{right}(v_k) \leftarrow w_q$. *(Update the current outer cycle.)*

5. If $p+1 = q$, then set $\mathrm{left}(v_k) \leftarrow 0$; else set $\mathrm{left}(v_k) \leftarrow w_{p+1}$ and $\mathrm{right}(w_{q-1}) \leftarrow 0$. *(Update the part that is covered by insertion of v_{k+1}.)*

6. Set $dx(w_q) \leftarrow \Delta_{pq} - dx(v_k)$ and—unless $p+1 = q$—set $dx(w_{p+1}) \leftarrow dx(w_{p+1}) - dx(v_k)$. *(Update the offsets according to the changes in the previous two steps.)*

Observe that the only step that possibly cannot be executed in constant time is Step 2. But all vertices but the last vertex w_q for which we sum the offsets are covered by the insertion of v_{k+1}. As every vertex can be covered at most once, the overall complexity of this step during the algorithm is linear. Therefore, this first phase of the algorithm can be completed in linear time.

In a second phase, the final x-coordinates can be computed from the offsets by a single recursive pre-order traversal of the tree. The (pseudo–)code given below is to be called with the root vertex v_1 and an offset of zero. Clearly this yields a linear time algorithm overall.

```
compute_coordinate(Vertex v, Offset d) {
  if (v == 0) return;
  x(v) = dx(v) + d;
  compute_coordinate(left(v), x(v));
  compute_coordinate(right(v), x(v));
}
```

Remarks. From a geometric complexity point of view, Theorem 2.32 provides very good news for planar graphs in a similar way that the Euler Formula does from a combinatorial complexity point of view. Euler's Formular tells us that we can obtain a combinatorial representation (for instance, as a DCEL) of any plane graph using $O(n)$ space, where n is the number of vertices.

Now the shift algorithm tells us that for any planar graph we can even find a geometric plane (straight-line) representation using $O(n)$ space. In addition to the combinatorial information, we only have to store $2n$ numbers from the range $\{0, 1, \ldots, 2n - 4\}$.

When we make such claims regarding space complexity we implicitly assume the so-called *word RAM model*. In this model each address in memory contains a *word* of b bits, which means that it can be used to represent any integer from $\{0, \ldots, 2^b - 1\}$. One also assumes that b is sufficiently large, for instance, in our case $b \geqslant \log n$.

There are also different models such as the *bit complexity model*, where one is charged for every bit used to store information. In our case that would already incur an additional factor of $\log n$ for the combinatorial representation: for instance, for each halfedge we store its endpoint, which is an index from $\{1, \ldots, n\}$.

Questions

1. *What is an embedding? What is a planar/plane graph?* Give the definitions and explain the difference between planar and plane.

2. *How many edges can a planar graph have? What is the average vertex degree in a planar graph?* Explain Euler's formula and derive your answers from it.

3. *How can plane graphs be represented on a computer?* Explain the DCEL data structure and how to work with it.

4. *How can a given plane graph be (topologically) triangulated efficiently?* Explain what it is, including the difference between topological and geometric triangulation. Give a linear time algorithm, for instance, as in Theorem 2.25.

5. *What is a combinatorial embedding? When are two combinatorial embeddings equivalent? Which graphs have a unique combinatorial embedding?* Give the definitions, explain and prove Whitney's Theorem.

6. *What is a canonical ordering and which graphs admit such an ordering? For a given graph, how can one find a canonical ordering efficiently?* Give the definition. State and prove Theorem 2.36.

7. *Which graphs admit a plane embedding using straight line edges? Can one bound the size of the coordinates in such a representation?* State and prove Theorem 2.32.

References

[1] Bruce G. Baumgart, A polyhedron representation for computer vision. In *Proc. AFIPS Natl. Comput. Conf.*, vol. 44, pp. 589–596, AFIPS Press, Alrington, Va., 1975, URL http://dx.doi.org/10.1145/1499949.1500071.

[2] Prosenjit Bose, Ferran Hurtado, Eduardo Rivera-Campo, and David R. Wood, Partitions of complete geometric graphs into plane trees. *Comput. Geom. Theory Appl.*, **34**, 2, (2006), 116–125, URL http://dx.doi.org/10.1016/j.comgeo.2005.08.006.

[3] John M. Boyer and Wendy J. Myrvold, On the cutting edge: simplified O(n) planarity by edge addition. *J. Graph Algorithms Appl.*, **8**, 3, (2004), 241–273, URL http://jgaa.info/accepted/2004/BoyerMyrvold2004.8.3.pdf.

[4] Bernard Chazelle, Triangulating a simple polygon in linear time. *Discrete Comput. Geom.*, **6**, 5, (1991), 485–524, URL http://dx.doi.org/10.1007/BF02574703.

[5] Marek Chrobak and Thomas H. Payne, A linear-time algorithm for drawing planar graphs. *Inform. Process. Lett.*, **54**, (1995), 241–246, URL http://dx.doi.org/10.1016/0020-0190(95)00020-D.

[6] István Fáry, On straight lines representation of planar graphs. *Acta Sci. Math. Szeged*, **11**, (1948), 229–233.

[7] Hubert de Fraysseix, Patrice Ossona de Mendez, and Pierre Rosenstiehl, Trémaux Trees and Planarity. *Internat. J. Found. Comput. Sci.*, **17**, 5, (2006), 1017—-1030, URL http://dx.doi.org/10.1142/S0129054106004248.

[8] Hubert de Fraysseix, János Pach, and Richard Pollack, How to Draw a Planar Graph on a Grid. *Combinatorica*, **10**, 1, (1990), 41–51, URL `http://dx.doi.org/10.1007/BF02122694`.

[9] Leonidas J. Guibas and Jorge Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graph.*, **4**, 2, (1985), 74–123, URL `http://dx.doi.org/10.1145/282918.282923`.

[10] John Hopcroft and Robert E. Tarjan, Efficient planarity testing. *J. ACM*, **21**, 4, (1974), 549–568, URL `http://dx.doi.org/10.1145/321850.321852`.

[11] Ken ichi Kawarabayashi, Yusuke Kobayashi, and Bruce Reed, The disjoint paths problem in quadratic time. *J. Combin. Theory Ser. B*, **102**, 2, (2012), 424–435, URL `http://dx.doi.org/10.1016/j.jctb.2011.07.004`.

[12] Lutz Kettner, *Software design in computational geometry and contour-edge based polyhedron visualization.* Ph.D. thesis, ETH Zürich, Zürich, Switzerland, 1999, URL `http://dx.doi.org/10.3929/ethz-a-003861002`.

[13] Kazimierz Kuratowski, Sur le problème des courbes gauches en topologie. *Fund. Math.*, **15**, 1, (1930), 271–283, URL `http://matwbn.icm.edu.pl/ksiazki/fm/fm15/fm15126.pdf`.

[14] László Lovász, Graph Minor Theory. *Bull. Amer. Math. Soc.*, **43**, 1, (2005), 75–86, URL `http://dx.doi.org/10.1090/S0273-0979-05-01088-8`.

[15] Bojan Mohar and Carsten Thomassen, *Graphs on surfaces.* Johns Hopkins University Press, Baltimore, 2001.

[16] David E. Muller and Franco P. Preparata, Finding the intersection of two convex polyhedra. *Theoret. Comput. Sci.*, **7**, (1978), 217–236, URL `http://dx.doi.org/10.1016/0304-3975(78)90051-8`.

[17] Neil Robertson and Paul Seymour, Graph Minors. XX. Wagner's Conjecture. *J. Combin. Theory Ser. B*, **92**, 2, (2004), 325–357, URL `http://dx.doi.org/10.1016/j.jctb.2004.08.001`.

[18] Walter Schnyder, Planar Graphs and Poset Dimension. *Order*, **5**, (1989), 323–343, URL `http://dx.doi.org/10.1007/BF00353652`.

[19] Carsten Thomassen, Kuratowski's Theorem. *J. Graph Theory*, **5**, 3, (1981), 225–241, URL `http://dx.doi.org/10.1002/jgt.3190050304`.

[20] William T. Tutte, A theorem on planar graphs. *Trans. Amer. Math. Soc.*, **82**, 1, (1956), 99–116, URL `http://dx.doi.org/10.1090/S0002-9947-1956-0081471-8`.

[21] Klaus Wagner, Bemerkungen zum Vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, **46**, (1936), 26–32, URL http://eudml.org/doc/146109.

[22] Klaus Wagner, Über eine Eigenschaft der ebenen Komplexe. *Math. Ann.*, **114**, 1, (1937), 570–590, URL http://dx.doi.org/10.1007/BF01594196.

[23] Kevin Weiler, Edge-based data structures for solid modeling in a curved surface environment. *IEEE Comput. Graph. Appl.*, **5**, 1, (1985), 21–40, URL http://dx.doi.org/10.1109/MCG.1985.276271.

[24] Hassler Whitney, Congruent Graphs and the Connectivity of Graphs. *Amer. J. Math.*, **54**, 1, (1932), 150–168, URL http://www.jstor.org/stable/2371086.

[25] Wikipedia, Planarity testing — Wikipedia, The Free Encyclopedia. 2013, URL http://en.wikipedia.org/wiki/Planarity_testing, [Online; accessed 19-Feb-2013].