Chapter 6

Vertical Decomposition and Triangulation of a Simple Polygon

By Sariel Har-Peled, January 31, 2023[1]

All the rocket ships are climbing through the sky,
the holy books are open wide,
the doctors working day and night
– Leonard Cohen.

Version: 0.1

6.1. Polygons

6.1.1. Vertical decomposition

Even a simple non-convex polygon with no holes is not easy to manipulate. A natural approach is to break them into smaller pieces that are easier to manipulate.

A natural way to do so is erect a vertical wall through each vertex of the polygon, where the wall is in the interior of the polygon. This breaks the polygon into ***vertical trapezoids***.

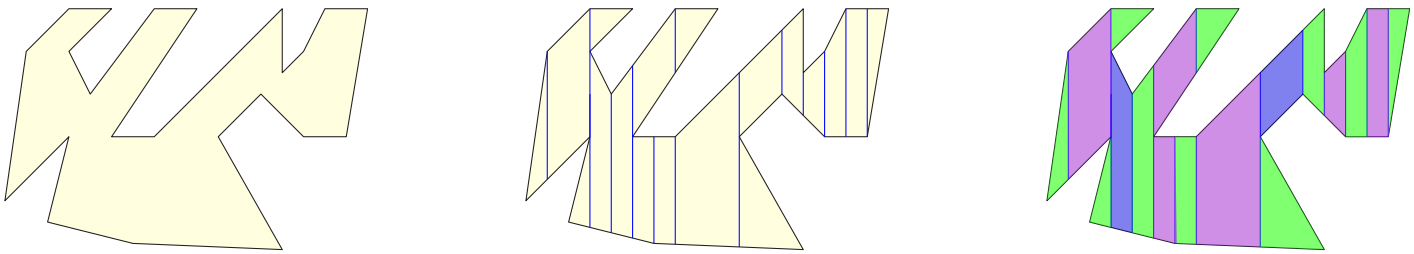

Figure 6.1: A polygon, its vertical walls, and the resulting vertical decomposition.

Computing this decomposition can be done using sweeping. Indeed, we precompute for every edge of the polygon the side of it that the interior of the polygon lies on (how?). Then, every time the sweep-line encounter a vertex, we can compute locally how the polygon looks like, end using the y-structure erect the vertical walls. Now, a careful implementation, enables us to compute explicitly the vertical trapezoids, and even build their dual graph – that is, for each vertical trapezoid we compute its adjacent vertical trapezoids. It is not hard to verify that a vertical trapezoid is adjacent to at most 4 other vertical trapezoids.

Lemma 6.1.1. *Given a simple polygon σ in the plane with n vertices, one can compute its vertical decomposition in $O(n \log n)$ time.*

6.1.2. Triangulation

A ***triangulation*** of a polygon is a partition of it into triangles by adding diagonals – here diagonals are straight segments connecting their endpoints. It is not even immediately clearly that such a partition exists.

Lemma 6.1.2. *Every polygon Π with $n \geq 4$ vertices has a diagonal that connects two non-adjacent vertices p, q of Π, such that the interior of the diagonal pq does not intersect the boundary of Π. That is, one can add pq to Π, and split it into two smaller polygons.*

Proof: Let b be the leftmost vertex of Π, and let u and v be its two adjacent vertices. Connect u and v by a curve γ that lies in the interior of Π. Now, imagine taking γ to be the shortest such curve – formally, we start with ubv, and let it shorten itself. If γ is a single segment, then we are done as it is the desired diagonal. Clearly, γ is a concave polygonal line, and its vertices belong to the vertices of Π. The polygon formed by u, b, v, γ is contained in Π, and it clearly have diagonals with b. Pick any of them to be the desired diagonal. ■

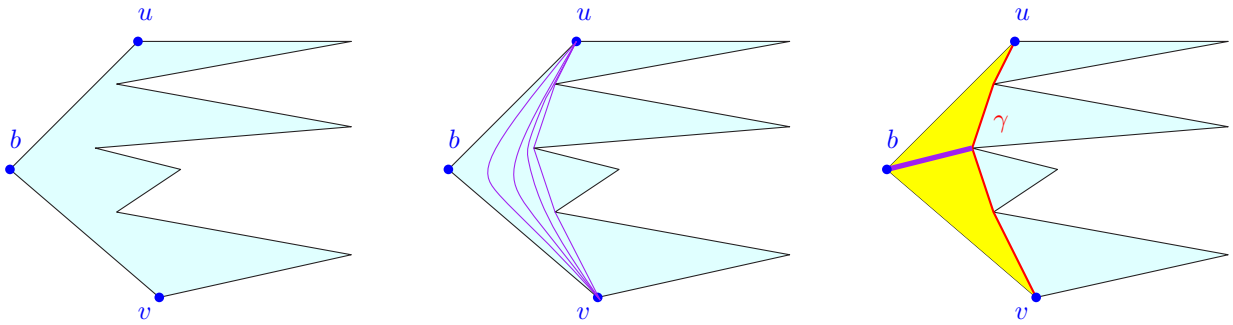

Figure 6.2: In the search for a diagonal.

Clearly, by repeatedly adding diagonals, splitting the polygon into two polygons and repeating the process repeatedly, we end up with a triangulation of the input polygon Π.

Theorem 6.1.3. *Let Π be a simple polygon in the plane. If it has n vertices, then it can be triangulated into $n - 2$ triangles.*

The natural question is how to triangulate a polygon efficiently?

6.2. The triangulation algorithm

6.2.1. First try: Use vertical decomposition

Compute the vertical decomposition of Π, and let the resulting set of vertical trapezoids be denoted by $\Pi_{\|}$. A key observation is that every vertical wall of a vertical trapezoid $\sigma \in \Pi_{\|}$ contains exactly one original vertex of Π. Namely, a vertical trapezoid from $\Pi_{\|}$ contains two original vertices on its two vertical walls. A natural approach is to just add a diagonal if the two such original vertices are not connected by an edge of Π already. This does not necessarily break the polygon into triangles, see Figure 6.3.

A polygon is ***x*-monotone** if any vertical line intersects it in an interval. It is easy to verify that adding the diagonals from the vertical decomposition results in breaking the original polygon into x-monotone polygon.

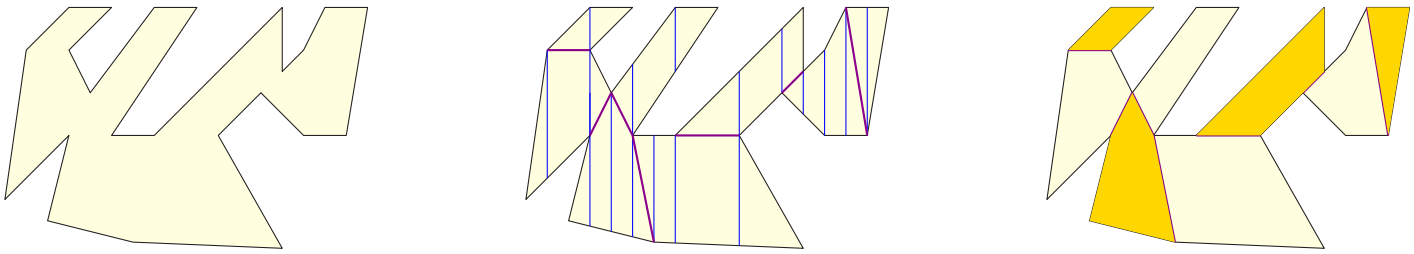

Figure 6.3: Vertical decomposition gets us close to a triangulation. Left: Original polygon. Middle: Vertical decomposition together with added diagonals (purple). Right: Resulting subpolygons of the original polygon.

Figure 6.4: Left: A merge vertex. Right: A split vertex.

Lemma 6.2.1. *After adding the diagonals from the vertical decomposition,* Π *is broken into* x*-monotone polygon.*

Proof: The key observation is that a polygon is not x-monotone, only if it has split or merge vertices. A ***merge*** vertex is a vertex where two portions of the polygon get merged into a single part during the sweep. More specifically, two intervals along the sweep line get merged into a single interval. Similarly, a split vertex is when the sweep "splits" a single interval along the sweep line into two, see Figure 6.4. The key observation is that to the right of a merge vertex there is a vertical trapezoid in the vertical decomposition, that give a rise to a diagonal. This diagonal prevents the two intervals into merging, by splitting the polygon into two polygons. See Figure 6.5. Clearly, the two new polygons are locally x-monotone around this vertex.

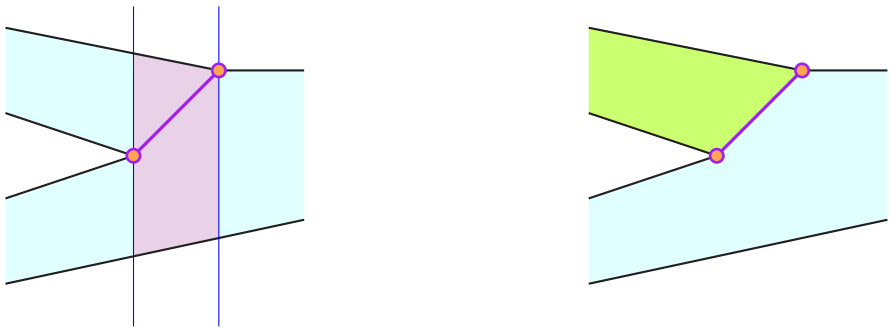

Figure 6.5: Left: A merge vertex and its diagonal from the vertical decomposition. Right: The resulting two polygons.

The same argument applies to split vertices. We conclude that the resulting subpolygons have no merge/split vertices, and are thus x-monotone. ■

We are left with the task of triangulating an x-monotone polygon.

6.2.2. Triangulating x-monotone polygons

An optional step (which we will not do!), somewhat strangely, it is worth running the vertical decomposition diagonal searching algorithm described above on each of these x-monotone polygons. This results in polygons that are x-monotone polygons where either the top part or bottom parts are a single segment. Such polygons are known as *mountains*. One can then triangulate mountains directly.

It turns out that implemented carefully, the algorithm for triangulating mountains can be applied to x-monotone polygons directly, with minor modifications. As such, we are going to describe how to triangulation x-monotone polygons directly. We first start with some "silly" easy cases to gain some intuition.

Triangulating pseudo-triangles. Let us start with the most stupid mountain – a ***pseudo-triangle*** – it is a polygon with a concave chain, and two other edges, one of them, say, forming the bottom of the polygon. See Figure 6.6. A pseudo-triangle is triangle with the inner edge "bend-in".

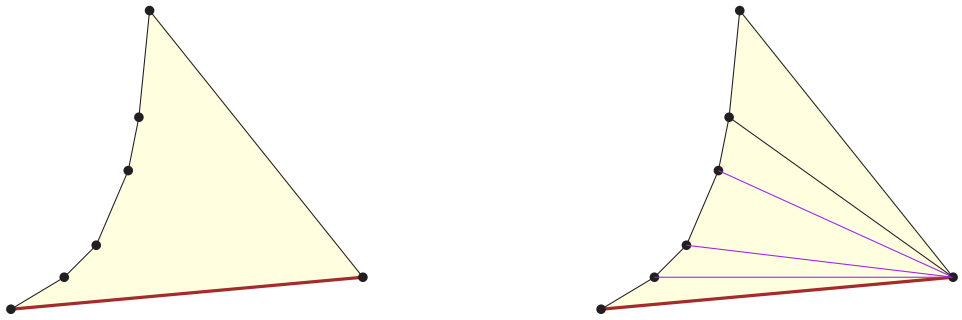

Figure 6.6: Triangulating a pseudo-triangle.

Back to x-monotone polygons. We already seen that shortest paths inside polygons form a concave chain. The idea would be to maintain two such shortest paths for the x-monotone polygon – these two chains would lead from the current sweep line back to some common vertex to both chains. We will maintain a concave chain for the bottom part of the polygon, and a concave chain for the top part of the x-monotone polygon. We will compute these chains using sweeping. The key idea is that we can not discharge/triangulate parts of the polygon as we do the sweeping. So, we sweep the x-monotone polygon from left to right, and we need to handle the various events.

Case I: Top chain is concave, and the sweep line encounters a bottom vertex. This case is depicted in Figure 6.7 – we triangulate the concave chain with the new vertex, and the last diagonal is now the "floor" chain of the sweep, and we continue the sweep to the next vertex.

Case II: Top chain is concave, and the sweep line encounters a top vertex. See Figure 6.8. If the new edge forms a right turn with the current top concave chain, then we create a triangle, and pop an edge from the concave chain. We repeat the process till the new chain is concave together with the new diagonal.

The other option is that the new vertex already form a left turn with the previous vertex in the concave chain – in this case we push the vertex to the chain, and continue the sweeping.

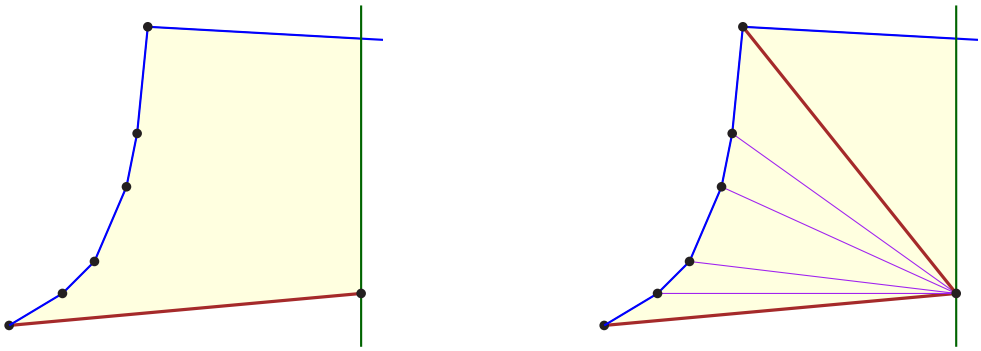

Figure 6.7: Triangulating a chain.

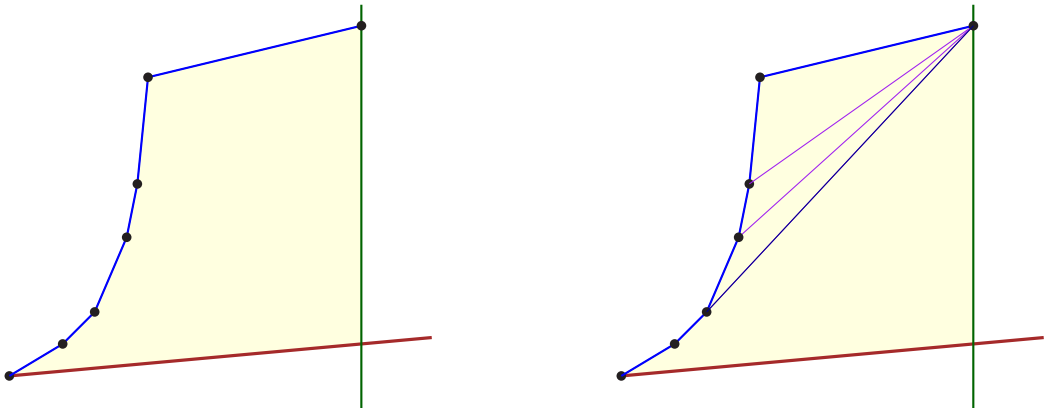

Figure 6.8: Triangulating a chain when a new top vertex arrives.

Other cases. All other cases are essentially symmetric variants of the above cases, as can easily be verified, and they are handled in similar fashion.

The algorithm. The key observation is that only one chain contains more than one vertex, at any give point in time – namely, one of the chains might contain several segments, while the other one is a portion of a segment that the sweep line sweep over. As such, by the end of the sweeping the x-monotone polygon is triangulated. It is not hard to verify that the running time of this algorithm is $O(n)$ time.

Lemma 6.2.2. *An x-monotone polygon with n vertices is triangulated in $O(n)$ time by the above algorithm.*

Theorem 6.2.3. *A simple polygon Π with n vertices can be triangulated in $O(n \log n)$ time.*

Proof: We compute the vertical decomposition of Π in $O(n \log n)$ time using sweeping. We then introduce the diagonals arising from the vertical trapezoids. This partition the polygon into subpolygons that are all x-monotone. Clearly, with careful implementation, one can extract all the subpolygons in linear time. Finally, the algorithm triangulate each subpolygon in linear time using the above algorithm for x-monotone polygons. ■

6.3. A side – separators in trees

Definition 1. *Let T be a tree with n nodes. A vertex* **separator** *in T is a node v, such that if we remove v from T, we remain with a forest, such that every tree in the forest has at most $\lceil n/2 \rceil$ vertices.*

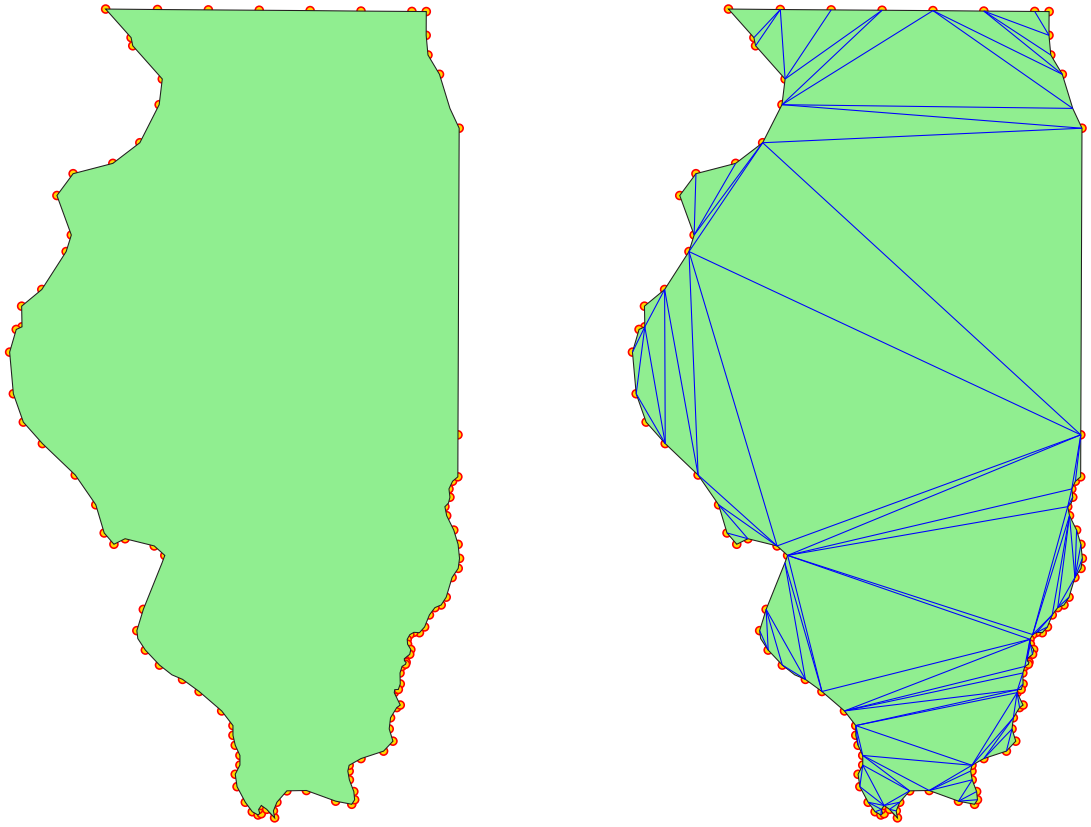

Figure 6.9: A somewhat simplified Illinois, and its triangulation. Note, that there are many long and skinny triangles, and also tiny triangles adjacent to large triangles. These undesirable features. How to get a better triangulation is an interesting topic that we will discuss later on.

Lemma 6.3.1. *Every tree has a separator, and it can be computed in linear time.*

Proof: Consider T to be a rooted tree. We precompute for every node in the tree the number of nodes in its subtree by a bottom-up computation that takes linear time. Set v to be the lowest node in T such that its subtree has $\geq \lceil n/2 \rceil$ nodes in it, where n is the number of nodes of T. This node can be found by performing a walk from the root of T down to the child with a sufficiently large subtree till this walk gets "stuck". Indeed, let v_1 be the root of T, and let v_i be the child of v_{i-1} with the largest number of nodes in its subtree. Let $s(v_i)$ be the number of nodes in the subtree rooted at v_i. Clearly, there exists a k such that $s(v_k) \geq n/2$ and $s(v_{k+1}) < n/2$.

Clearly, all the subtrees of the children of v_k have size at most $n/2$. Similarly, if we remove v_k and its subtree from T, we remain with a tree with at most $n/2$ nodes. As such, v_k is the required separator. ■

Definition 2. *Let T be an unrooted tree with n nodes and maximum degree d. An* **edge separator** *edge separator in T is an $\mathbf{edge}$, such that if we remove $\mathbf{e}$ from T, we remain with two tress, such that each tree has number of vertices in the range $\lfloor \frac{n}{d} \rfloor, \ldots, \lceil \frac{d-1}{d} n \rceil$.*

Lemma 6.3.2. *Consider an unrooted tree T with n vertices and maximum degree has an edge separator e, and it can be computed in linear time. Specifically, after removing e, T breaks into two trees T_1 and T_2, such that $\lfloor \frac{n}{d} \rfloor \leq |\mathsf{V}(\mathsf{T}_i)| \leq \lceil \frac{d-1}{d} n \rceil$.*

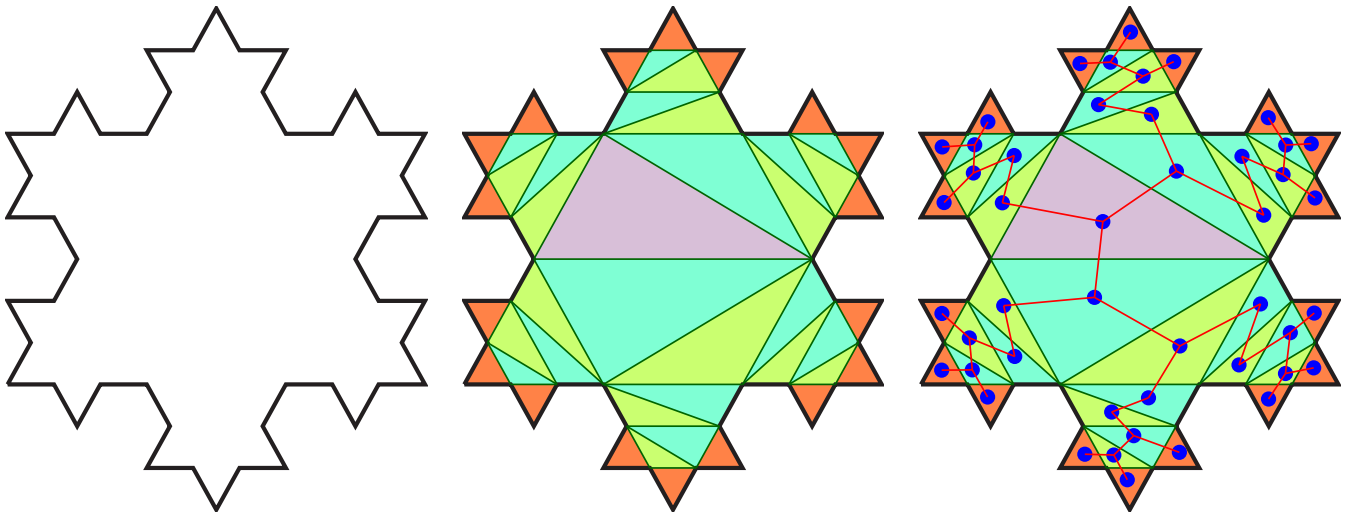

Figure 6.10: The polygon formed by Koch's snowflake of iteration 3, its triangulation, and the corresponding dual graph.

Proof: The proof follows the same argument as above, but the details are tedious – the reader is encouraged to skip it. Pick any leaf v_0 of T to be its root. Let $s(v)$ denote the number of nodes in the subtree of v for a node v of T. Let v_i the child of v_{i-1} such that $s(v_i)$ is maximized. Since T is finite, the path $v_0, v_1, \ldots, v_k$ ends in a leaf v_k. Observe that, for all i, we have

$$s(v_0) = n, \qquad s(v_i) < s(v_{i-1}), \qquad s(v_k) = 1, \qquad \text{and} \qquad s(v_{i-1}) \leq (d-1)s(v_i) + 1.$$

So, let τ be the last index such that $s(v_\tau) > \alpha n$, for $\alpha = \frac{1}{d} - \frac{1}{(d-1)n}$. We have that

$$s(v_\tau) < s(v_{\tau-1}) \leq (d-1)s(v_\tau) + 1 < (d-1)\alpha n + 1 \leq \left\lceil \frac{d-1}{d} n - 1 \right\rceil + 1 = \left\lceil \frac{d-1}{d} n \right\rceil.$$

Thus, removing the edge $v_\tau v_{\tau-1}$, we break the tree into two parts. One of the trees T_1 containing v_τ has $n/d - 1/(d-1) < s(v_\tau) \leq \lceil \frac{d-1}{d} n \rceil$ nodes. If d divides n, this implies that $\frac{n}{d} \leq s(v_\tau) \leq \frac{d-1}{d} n$, and we are done. Otherwise, we have

$$|\mathsf{V}(\mathsf{T}_2)| = n - s(v_\tau) \geq n - \left\lceil \frac{d-1}{d} n \right\rceil \geq n - \frac{d-1}{d} n - 1 = \frac{n}{d} - 1 = \lfloor \tfrac{n}{d} \rfloor. \qquad \blacksquare$$

6.4. Applications of triangulation of simple polygon

6.4.1. Application: Guarding a simple polygon

A ***leaf*** in a tree is a node of degree one.

Lemma 6.4.1. *A tree T with $n > 1$ nodes has at least two leafs.*

Proof: Since a tree with n vertices has $n-1$ edges, its average degree is $2(n-1)/n < 2$. Namely, there must exist a vertex of degree one, which is a leaf. Let u be this leaf, and consider the longest path π in T that starts at u. Let v be the other endpoint of π, and observe that v must be a leaf. If not, either one can make π longer (a contradiction), or there is a cycle in the tree (a contradiction). $\blacksquare$

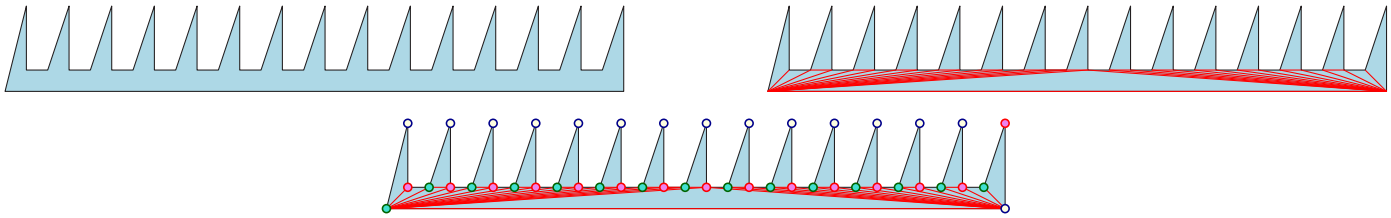

Figure 6.11: Every tooth requires a separate guard. As such, this polygon needs $n/3$ guards. This polygon has 15 "teeth" and 45 vertices.

Consider a simple polygon Π with n vertices. Let Ξ be any triangulation of Π. This triangulation has $n-2$ triangles, and $n-3$ diagonals. Interpret the triangulation as a planar graph G.

Lemma 6.4.2. *A triangulation G of a simple polygon Π is 2-degenerate.*

Proof: Let Π be formed by the vertices $v_1, v_2, \ldots v_n$. If Π is a single triangle, then the claim immediately holds. So assume the number n of vertices of Π is larger than three. The dual graph $\mathsf{T} = \mathsf{G}^\star$ is a tree, and it has two leafs. A leaf corresponds to a triangle, where its edge connect two vertices v_i and v_{i+2} (modulo n in necessary) for some i. Clearly, v_{i+1} has degree 2, as Removing v_{i+1} results in a new triangulation $\mathsf{G} = \mathsf{G} - v_i$ of a simple polygon with $n-1$ vertices. We now use induction. ■

This readily implies that a triangulation of a simple polygon is 3-colorable.

In the ***art gallery*** problem, one needs to plane minimum number of points inside a simple polygon, so that every point in the polygon is visible from one of the guard points.

Lemma 6.4.3. *Any polygon Π with n vertices can be guarded using $\lfloor n/3 \rfloor$ guards, and there are polygons that require at least $n/3$ guards.*

Proof: The lower bound is in Figure 6.11. The upper bound follows by triangulating Π. Let Ξ be the resulting triangulation. This triangulation is 3-colorable, and consider such a coloring using colors $1, 2$ and 3. Assume that color 3 is used the fewest number of times, and let U be this set of vertices. Every triangle has a vertex of U, as such using U for the locations of the guards, we can guard the polygon using $|U| \geq \lfloor n/3 \rfloor$ guards. ■

6.4.2. Application: Sweeping a simple polygon using $O(\log n)$ guards