1. An interior diagonal of a simple polygon P with $n \geq 4$ vertices is a *balanced separator* if it subdivides P into two smaller polygons, each with at least $\lceil n/3 \rceil + 1$ vertices. Describe and analyze an algorithm to find a balanced separator in a given simple polygon P (with at least four vertices). *[Hint: Prove that a balanced separator always exists!]*

Solution: First I'll prove the following stronger theorem: *Every frugal triangulation of P contains a diagonal that is a balanced separator.*

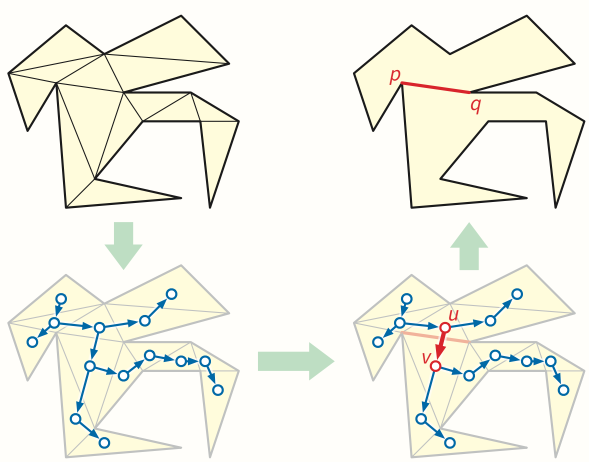
Fix an arbitrary frugal triangulation T of P. Let T^* be the *interior dual* graph of T, which has a vertex for each triangle of T and an edge for each diagonal of T. (The complete dual graph of T has one more vertex, dual to the unbounded outer face, but we're ignoring that vertex.)

It is not hard to show that T^* is a tree. First, T^* is connected because the interior of P is connected. Second, T^* is acyclic because the boundary of P is connected.

Choose an arbitrary leaf of T^* (that is, an ear in T) as its root, so that every vertex in T has at most two children. For any vertex v (that is, any triangle in T), let N_v denote the number of descendants of v in T, including v itself. Finally, let $N = n - 2$ denote the number of vertices in T^*.

Now let u be the deepest vertex in T^* such that $N_u \geq (2N+1)/3$. (This vertex is clearly unique.) If u has only one child, let v be that child; if u has two children, label them v and w so that $N_v \geq N_w$. By construction, we have $N_v \geq \frac{1}{2}((2N+1)/3 - 1) = (N-1)/3$ and $N_v < (2N+1)/3$, which implies $N - N_v > (N-1)/3$. In other words, deleting edge uv splits T^* into two smaller trees, each with at least $(N-1)/3$ vertices.

Let pq be the diagonal that is dual to the tree edge uv. We just showed that cutting along pq splits T into two smaller polygon triangulations, each with at least $(N-1)/3$ triangles, and therefore at least $(N-1)/3 + 2 = n/3 + 1$ vertices. We conclude that pq is a balanced separator. (We get the ceiling for free, because the number of vertices must be an integer!)



Thus, we can find a balanced separator by triangulating P in $O(n \log n)$ time, computing the depth and number of descendants of each vertex in the triangulation's dual tree in $O(n)$ time, and then choosing the tree edge uv as described above. The total running time of the algorithm is $O(n \log n)$. ■

Rubric: 10 points. Partial credit for weaker separation bounds:

- -1 for $n/3 + c$ for any constant $c < 1$.
- -2 for $\alpha n \pm o(n)$ for any constant $0 < \alpha < 1/3$.

2. Let P be an arbitrary generic simple orthogonal polygon with n vertices.

(a) Prove that P has exactly $n/2 - 2$ reflex vertices.

Solution: Orient P counterclockwise (that is, with the interior on the left), so that its total turning angle is 1. Suppose P has r reflex vertices, and therefore $n - r$ convex vertices. Each convex vertex has a turning angle of $+1/4$, and each reflex angle has a turning angle of $-1/4$. It follows that $(n - r)/4 - r/4 = 1$, which immediately implies $r = n/2 - 2$. ■

Rubric: 2 points.

(b) Prove that every proper rectangulation of P has exactly $n/2 - 1$ rectangles.

Solution (Euler): Let R be an arbitrary proper rectangulation R of P. I'll compute the number of rectangles in R by counting its vertices and edges and then applying Euler's formula.

Our general position assumption implies that every vertex of R has degree 2 or 3. A vertex of R has degree 2 if and only if it is a convex boundary vertex. Thus part (a) implies that R has exactly $n/2 + 2$ degree-2 vertices. Counting degree-3 vertices is more complicated.

At each degree-3 vertex v, exactly one incident edge is adjacent to two right angles; call this edge the *stem* of v and the other two edges the *arms* of v. (Think of the vertical stem and horizontal arms of the letter T.) The interior edges of R are exactly covered by orthogonal line segments of three different types:

- Segments that connect an arm of a reflex boundary vertex with the stem of another vertex.
- Segments that connect arms of two reflex vertices. But any such segment must be collinear with two boundary edges, violating our assumption that the polygon P is generic.
- Segments that connect stems of two vertices. But any such segment is a bar that does not contain an edge of P, violating our assumption that the rectangulation R is proper.

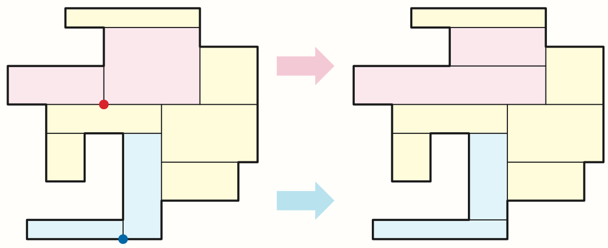
We conclude that the number of degree-3 vertices in R is exactly twice the number of reflex boundary vertices. Part (a) implies that R has exactly $n - 4$ degree-3 vertices.

Altogether, R has $3n/2 - 2$ vertices. The number of edges is $\frac{1}{2}\sum_v \deg(v) = \frac{1}{2}(2 \cdot (n/2 + 2) + 3 \cdot (n - 4)) = 2n - 4$. Thus, **Euler's formula** implies that R has $2 - V + E = 2 - (3n/2 - 2) + 2n - 4 = n/2$ faces, *including the outer face*. We conclude that R has $n/2 - 1$ interior faces, as claimed. ■

Solution (flips): The simplest non-trivial example of an orthogonal polygon is an L-shaped hexagon; this polygon has exactly two proper rectangulations, each with two rectangles.

Let R be any proper rectangulation of any generic orthogonal polygon P. Call a vertex of R *new* if it is not also a vertex of P. Genericity implies that every new vertex v of R has degree 3; moreover the two right angles at v both lie in the interior of P. The union of these two rectangles is orthogonal hexagon.

A *flip* transforms R into another rectangulation R' by finding two adjacent rectangles in R whose union is an orthogonal hexagon L and replacing those two rectangles with the other rectangulation of L. (See the figure below.) Flips do not change the number of vertices, edges, or rectangles. Every new vertex of R corresponds to a possible flip.



Two flips in a rectangulation.

So let R be any proper rectangulation of P with $h > 0$ horizontal edges. Let e be any horizontal edge; let b be the horizontal bar containing R, let v be an endpoint of b that is *not* a vertex of P (which must exist by genericity), and let uv be the edge in b that is incident to v. Then v is a new vertex; the corresponding flip replaces the horizontal edge uv with a vertical edge uw. The resulting rectangulation R' has $h-1$ horizontal edges. It follows by induction that some sequence of h flips transforms R into a rectangulation of P with only vertical edges.

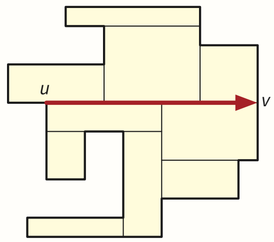
In fact, P has a unique vertical rectangulation $R^|$, which is constructed by the sweepline algorithm in my solution to part (c). Suppose P has at least one reflex vertex, since otherwise the problem is trivial. Let r be a rectangle in $R^|$ whose right wall contains the leftmost reflex vertex of P. (There are at most two such rectangles.) Removing r from $R^|$ yields a smaller rectangulation with $n-2$ boundary vertices and (by the induction hypothesis) $(n-2)/2-1 = n/2-2$ rectangles. It follows that that $R^|$ has $n/2-1$ rectangles. Because flips do not change the number of rectangles, we conclude that every proper rectangulation of P has $n/2-1$ rectangles. ■

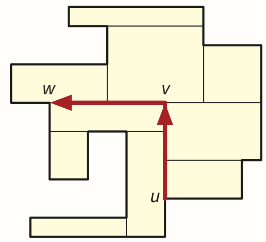
Solution (induction): I'll prove the claim by induction on n. Let R be an arbitrary proper rectangulation of an arbitrary generic orthogonal polygon P, and let n be the number of vertices of P. If P is a rectangle, R has exactly one rectangle and $n = 4$, and therefore $n/2-1 = 4/2-1 = 1$. So assume that $n \geq 6$.

Let b be any bar in R that intersects the interior of P. The definitions of "proper" implies that b contains an edge e of P. Let s be a maximal subsegment of b in the interior of P. (There are at most two such segments.) One endpoint u of s is also an endpoint of e, and therefore a reflex vertex of P. There are two possibilities for the other endpoint v:

- Suppose v lies on the boundary of P. If v were a *vertex* of P, it would lie on an edge collinear with e, violating our assumption that P is generic. Segment uv partitions R into two smaller rectangulations R_1 and R_2, of two smaller orthogonal polygons P_1 and P_2, respectively.

- Suppose v lies in the interior of P. Then v must lie on another bar b' orthogonal to b. This bar must contain another edge e' of P, and therefore must contain a segment vw, where w is a reflex vertex of P. The L-shaped path uvw partitions R into two smaller rectangulations R_1 and R_2, of two smaller orthogonal polygons P_1 and P_2, respectively.



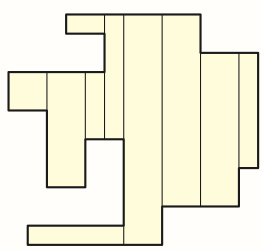


In both cases, v is a vertex of both P_1 and P_2 each vertex of P is a vertex of exactly one of P_1 and P_2, and these are the only vertices of P_1 and P_2. Thus, $n_1 + n_2 = n + 2$, where n_1 and n_2 respectively denote the number of vertices of P_1 and P_2. The induction hypothesis implies that R_1 has $n_1/2 - 1$ rectangles and R_2 has $n_2/2 - 1$ rectangles. We conclude that R has $n_1/2 - 1 + n_2/2 - 1 = (n_1 + n_2)/2 - 1 = (n + 2)/2 - 2 = n/2 - 1$ rectangles. ■

Rubric: 4 points. These are not the only correct solutions!!

(c) Describe an algorithm to construct a proper rectangulation of P. (In particular, this algorithm proves that a proper rectangulation always exists!)

Solution: We construct a degenerate trapezoidal decomposition using a sweep-line algorithm. At each vertical edge, we extend vertical segments through the interior of P, possibly in both directions or neither, until they touch P again.



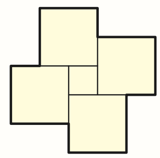
The only differences from the Shamos-Hoey algorithm are (1) the sweep BST can directly use y-coordinates of the horizontal edges as search keys, and (2) at each vertical edge, we perform two update operations on the sweep BST instead of one: either two insertions, two deletions, or one of each. The algorithm still runs in $O(n \log n)$ time.

Alternatively, we can construct the decomposition in $O(n \log n)$ expected time using a randomized incremental algorithm, with similar minor modifications to handle vertical segments. (For example, a single trapezoid in an intermediate decomposition can now have up to six neighbors—three through each vertical wall—because a vertical wall can contain two segment endpoints.) ■

Rubric: 2 points. This is not the only correct solution.

(d) Prove or disprove: In every proper rectangulation R of P, every rectangle in R touches the boundary of P.

Solution: Here is a minimal counterexample:



This example also shows that the second case of the inductive proof of part (b) is actually necessary; not every rectangulation has a *guillotine cut* — an interior segment with both endpoints on the boundary of P. ■

Rubric: 2 points

3. Let $S = \{s_1, s_2, \dots, s_n\}$ be a set of n disjoint line segments, and let $T(S)$ denote the trapezoidal decomposition of S. For each segment $s \in S$, let $\mathbf{deg(s)}$ denote the number of trapezoids incident to s.

   (a) Prove that $\sum_{i=1}^{n} \deg(s_i) \le \alpha n$ for some constant α.

Solution: Let $T(S)$ denote the restriction of the trapezoidal decomposition of S to a large axis-aligned rectangle R that contains S in its interior. (Restricting the decomposition to R lets us avoid sentinel vertices at infinity, which complicate the analysis.) We count the vertices, edges, and faces of $T(S)$ as follows:

- $T(S)$ has exactly $6n+4$ vertices: the endpoints of each segment in S, the top and bottom endpoints of vertical walls through those endpoints, and the four corners of R.
- The four corners of R have degree 2; assuming general position, all other vertices of $T(S)$ have degree 3. Thus, the sum of all vertex degrees is $2 \cdot 4 + 3 \cdot 6n = 18n + 8$. The sum of all vertex degrees is also twice the number of edges. So $T(S)$ has exactly $9n+4$ edges.
- Finally, Euler's formula implies that $T(S)$ has exactly $2 - V + E = 2 - (6n+4) + (9n+4) = 3n+2$ faces.

Now we count incidences between trapezoids in $T(S)$ and segments in S. The unbounded outer face of $T(S)$ is not incident to any segments; the leftmost and rightmost bounded faces are each incident to one segment; and assuming general position, the other faces of $T(S)$ are incident to at most four segments each. We conclude that

$$\sum_{s \in S} \deg(s) \le 2 + 4 \cdot (3n - 1) < 12n.$$

■

Rubric: 3 points. This is not the only correct proof. An $O(n)$ bound without an explicit constant is worth at most 2 points. No penalty for a larger constant α.

   (b) Let α be the constant derived in your solution to part (a). We say that a segment $s \in S$ is *long* if $\deg(s) \ge 2\alpha$ and *short* otherwise. Prove that the number of short segments in S is at least βn, for some constant β (which may depend on α).

Solution: Suppose we choose a segment s uniformly at random from S. Part (a) implies that $\mathrm{E}[\deg(s)] \le \alpha$, which implies

$$\Pr[s \text{ is long}] \;=\; \Pr[\deg(s) \ge 2\alpha] \;\le\; \Pr[\deg(s) \ge 2\,\mathrm{E}[\deg(s)]] \;\le\; 1/2.$$

by Markov's inequality.[a] So at least half the segments in S are short. ■

[a] $\Pr[X \ge x] \le \mathrm{E}[X]/x$ for any random variable X and any positive real number x. This is a weighted version of the pigeonhole principle.

Solution (Let's prove Markov's inequality!): Suppose S contains ℓ long segments and $n-\ell$ short segments. Every segment in S is incident to at least four trapezoids. It follows that

$$\begin{aligned}\sum_{s\in S}\deg(s) &= \sum_{\text{short } s\in S}\deg(s) + \sum_{\text{long } s\in S}\deg(s)\\ &\geq \sum_{\text{short } s\in S} 4 + \sum_{\text{long } s\in S} 2\alpha\\ &= 4(n-\ell)+2\alpha\ell\\ &= 4n-(2\alpha+4)\ell\end{aligned}$$

So part (a) implies that

$$4n-(2\alpha+4)\ell \leq \alpha n \implies \ell \leq \frac{\alpha-4}{2\alpha+4}n \leq \frac{n}{2}.$$

We conclude that S contains at least $n/2$ short segments.

(The inequality $\alpha < 12$ from part (a) actually implies that S contains more than $5n/7$ short segments.) ■

Rubric: 3 points. These are not the only correct proofs. An $\Omega(n)$ bound without an explicit constant is worth at most 2 points. No penalty for a smaller constant β.

(c) An *independent set* of segments is any subset $I \subseteq S$ such that each trapezoid in $T(S)$ is incident to at most one segment in I. Prove that S contains an independent set of at least γn *short* segments, for some constant γ (which may depend on α and β).

Solution: After computing the trapezoidal decomposition of S, we can construct the desired independent set using the following greedy algorithm. First, we initialize S' to the set of all short segments, and initialize I to the empty set. Then, as long as S' is nonempty, we repeatedly choose an arbitrary segment $s \in S'$, add the chosen segment s to I, and remove every segment from S' that shares a trapezoid with s.

```
LowDegreeIndSet(S):
  I ← ∅
  S' ← short segments in S
  while S' ≠ ∅
      s ← any segment in S'
      add s to I
      for all trapezoids Δ incident to s
          for all segments s' incident to Δ
              remove s' from S'
  return I
```

By construction, I is an independent set of segments from S', so every segment in I is short. By definition, each short segment is incident to less than 2α trapezoids, where α is the constant from part (a), and each of those trapezoids is incident to at most three other segments in addition to s. Thus, very crudely, each time we add one short segment to I, we remove at most $6\alpha+1$ segments from S'.[a]

Part (b) implies that S' initially contains at least βn segments. It follows that when the algorithm ends, we have $|I| \geq (\beta n)/(6\alpha+1)$. In particular, the bounds $\alpha < 12$ and $\beta > 5/7$ from our earlier solutions imply $|I| > 5n/441$. ■

[a] More careful analysis reduces this crude upper bound from $6\alpha+1$ to $2\alpha+1$, which implies the stronger lower bound $|I| > n/35$.

Rubric: 4 points. This is not the only correct proof. An $O(n)$ bound without an explicit constant is worth at most 3 points. No penalty for a smaller constant γ.

4. (a) Describe an algorithm to compute the motorcycle graph of n moving points in $O(n^2 \log n)$ time.

Solution: My algorithm computes an array $Death[1..n]$, where $Death[i]$ is the time when the ith motorcycle crashes, or ∞ if it never crashes. It is easy to construct an explicit representation of the motorcycle graph from this array in $O(n)$ time if necessary.

For each pair of indices $i \neq j$, let $t_{i \to j}$ denote the time the ith motorcycle crosses the line containing the track the jth motorcycle. We can compute both $t_{i \to j}$ and $t_{j \to i}$ in constant time by solving the linear system

$$x_i + u_i \cdot t_{i\to j} = x_j + u_j \cdot t_{j \to i}$$
$$y_i + v_i \cdot t_{i\to j} = y_j + v_j \cdot t_{j \to i}$$

To compute all collision times, we consider all $n(n-1)$ possible collisions ***in chronological order***. Motorcycle i actually crashes at time $t_{i\to j}$ if and only if the following conditions hold:

- $t_{i\to j} > 0$ – Motorcycle i initially moves toward the collision point.
- $t_{j\to i} > 0$ – Motorcycle j initially moves toward the collision point.
- $t_{i\to j} < Death[i]$ – Motorcycle i actually reaches the collision point.
- $t_{j\to i} < Death[j]$ – Motorcycle j actually reaches the collision point.
- $t_{i\to j} > t_{j\to i}$ – Motorcycle i reaches the collision point after motorcycle j.

(The second and fifth inequalities make the first inequality redundant.) If we discover that motorcycle i crashes at time $t_{i\to j}$, we set $Death[i] \leftarrow t_{i\to j}$. Because we consider events in chronological order, each collision time $Death[i]$ is set exactly once.

```
MotorcycleDeaths(P[1..n]):
  ⟨⟨Preprocessing⟩⟩
  for i ← 1 to n
      Death[i] ← ∞
  for i ← 1 to n − 1
      for j ← i + 1 to n
          compute Time[i, j] and Time[j, i]
  Events[1..n(n − 1)] ← array containing every index pair (i, j)
  sort Events by Time[i, j]
  ⟨⟨Now for the real work!⟩⟩
  for k ← 1 to n(n − 1)
      (i, j) ← Events[k]
      if Time[i, j] ≤ Time[j, i] ≤ Death[i] and 0 ≤ Time[j, i] ≤ Death[j]
          Death[i] ← Time[i, j]
  return Death[1..n]
```

The algorithm runs in $O(n^2 \log n)$ time; the running time is dominated by sorting the potential collision times. ■

Rubric: 5 points. This is not the only correct solution.

(b) Describe an algorithm to compute the motorcycle graph of n moving points in $O(n \log n)$ time when $x_i = 0$ for every index i; that is, all n points start on the y-axis.

Solution: We separately consider motorcycles moving to the right ($u_i > 0$) and motorcycles moving to the left ($u_i < 0$), since those two sepcies of motorcycle never interact. I'll describe how to handle the rightward bikes below; leftward bikes are handled symmetrically. (General position implies that none of the bikes move vertically ($u_i = 0$), but vertical bikes are easy to handle if necessary, by sorting along the y-axis.)

To find collisions among the rightward motorcycles, we consider possible collisions ***in order from left to right***. In effect, we imagine that each motorcycle has velocity $(1, v_i/u_i)$, so that all motorcycles always have the same x-coordinate, but we use the original velocities (u_i, v_i) to determine which motorcycle survives each collision. The algorithm closely follows the Bentley-Ottmann sweep-line algorithm for counting line-segment intersections.

At the start, we sort the motorcycles by their starting y-coordinates.

We maintain an ordered dictionary storing the indices of all tracks in order along the sweepline. Initially, this sweep dictionary contains the indices $1, 2, \ldots, n$ in order. Whenever we detect a collision, we DELETE the index of one motorcycle from the sweep dictionary; we never INSERT into the sweep dictionary after its initialization.

We also maintain a priority queue containing pairs of motorcycles that are adjacent along the sweep line. The priority of each pair (i, j) is the x-coordinate $x(i, j)$ of the point where the lines containing tracks i and j intersect. (However, if $x(i, j) < 0$, we exclude the pair (i, j) from the event queue.) Thus, initially, the event queue contains (some subset of) the $n-1$ pairs $(1,2), (2,3), \ldots, (n-1, n)$.

At each iteration of the main loop, we find the leftmost event (i, j) in the priority queue. If either motorcycle i or motorcycle j is already dead, we ignore the event. Otherwise, suppose motorcycle i reaches $x(i, j)$ later than motorcycle j. We record the death of motorcycle i, delete i from the sweep dictionary, and insert a new potential collision event between the predecessor of i and the successor of i in the sweep dictionary.

After initialization, the event queue contains at most $n-1$ potential events. In the main loop, after each collision, we insert one new event into the event queue, and there are at most $n-1$ collisions. It follows that our algorithm processes at most $2n-2$ events. Each event requires $O(\log n)$ time for the sweep-dictionary and event-queue operations, so the overall running time is $O(n \log n)$. ■

Rubric: 5 points. This is not the only correct solution.