Min-Weight Triangulation

input: A sequence $p[1..n]$ of points in the plane, forming (in clockwise order) the vertices $p[1], p[2], \ldots, p[n]$ of a convex polygon.

output: The minimum weight of any *triangulation* of p. A triangulation of p is a set of *chords* (diagonals — straight-line segments connecting pairs of vertices) that subdivides the convex polygon into triangular faces. The weight of the triangulation is the sum of the lengths of the chords. We include the boundary segments (with endpoints p_i, p_{i+1}) as chords.

Here is the algorithm. Fix any input instance $p[1..n]$.

subproblems. For each pair (i, j) with $1 \leq i < j \leq n$, and $j - i \geq 1$, define $T[i, j]$ to be the optimal cost for the subproblem $p[i..j]$ formed by the $j - i + 1 \geq 2$ points $p[i], p[i+1], \ldots, p[j]$. The answer to the given problem is then $T[1, n]$.

recurrence. Given two points $p[i] = (x_i, y_i)$ and $p[j] = (x_j, y_j)$, their Euclidean distance is $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. Denote this distance by $d(p[i], p[j])$.

Here is the recurrence. In the boundary case $j = i+1$, the subproblem has two points $p[i], p[i+1]$, The (degenerate) triangulation is the chord with these two points as endpoints, of weight $T[i, j] = d(p[i], p[j])$. Then, for subproblems with $j - i + 1 \geq 3$ points,

$$T[i, j] = \min \big\{ T[i, k] + T[k, j] + d(p[i], p[j]) : k \in \{i + 1, i + 2, \ldots, j - 1\} \big\}.$$

The intuition is as follows. Each triangulation of $p[i..j]$ can be obtained as follows for some $k \in \{i + 1, \ldots, j - 1\}$: take any triangulation of $p[i..k]$, and any triangulation of $p[k..j]$, and add the chord with endpoints $p[i]$ and $p[j]$. (That chord must be in the triangulation, and point $p[k]$ is the third vertex on the triangle that has that chord as one side.) Draw a picture and consider some examples for more intuition. Make sure to consider the boundary cases when $k \in \{i + 1, j - 1\}$.

There are other ways of setting up a recurrence. This recurrence splits the triangulation around the triangle with vertices $p[i], p[j], p[k]$, then recurses on each side. One could split the triangulation around other triangles, or split the triangulation around a single chord. However, these approaches may lead to exponentially many subproblems.

time. There are $O(n^2)$ subproblems. For each, the right-hand side of the recurrence takes $O(n)$ time to evaluate (assuming the smaller subproblems are already solved). So the total time is $O(n^3)$.

correctness. To show correctness, we would use the ideas sketched above to carefully prove that the recurrence relation is correct. Correctness of the algorithm would follow by induction (on the size of the subproblem $T[i, j]$).

Is this algorithm equivalent to running the Shortest Path algorithm on an appropriate graph? The standard trick of creating a graph with one vertex per subproblem $T[i, j]$ doesn't work, because, in the right-hand side of the recurrence for $T[i, j]$, for each k, the term includes *two* subproblems $T[i, k]$ and $T[k, j]$.