Homework 9

Brian Knotten, Brett Schreiber, Brian Falkenstein

September 18, 2018

6

The algorithm will iterate over a $n \times n$ matrix, D, where n is the number of points in the input. Each row and column will represent a point in P, so that $D[i, j]$ represents some relation between points i and j. Each cell will contain a value. This value corresponds to the edge length (P_i, P_j) plus the minimum length of cuts up until that edge is considered.

The algorithm begins by fixing an edge in the polygon, specifically edge P_1 and P_n. This edge corresponds with the bottom left corner of the matrix, $D[0, n]$. Which edge is fixed has no effect on the algorithm, as it will still consider every way of cutting the polygon with $n - 3$ edges (resulting in $n - 2$ triangles).

Cell $D[i, j]$ is equal to the distance (P_i, P_j) plus the minimum of $D[i + 1, j]$ and $D[i, j - 1]$. Because of this, the matrix will be filled left to right, then bottom to top (so that higher indexed rows may be accessed in, IE $D[i+1, j]$, and lower indexed columns may also be accessed, IE $D[i, j - 1]$).

The solution will be the minimum value over the diagonal beginning at $D[0, 2]$ IE $D[x, x+3]$ for $0 < x < n-1$. This diagonal represents all the values with all possible edge combinations, so taking the min over this diagonal will give the optimal solution.

The actual edges in the solution can easily be determined via backtracking. If the min value is found at point $D[i, j]$, then we know that edge (P_i, P_j) is in the solution. We can then get the next edge in the solution by finding the min of $D[i + 1, j]$ and $D[i, j - 1]$, and taking that edge, etc.