1. This problem concerns a one dimensional clustering problem. Let $X = \{x_1, \ldots, x_n\}$ be $n$ distinct real valued points with $x_1 < x_2 < \ldots < x_n$. A $k$-clustering of $X$ is a partitioning of $X$ into $k$ disjoint subsets (clusters) $C_1, \ldots, C_k$. The diameter $\text{diam}(C)$ of a cluster $C$ is defined as $\max\{|x - y| : x, y \in C\}$. We want to find a $k$-clustering $C_1, \ldots, C_k$ so as to minimize $\max_i \text{diam}(C_i)$.

Provide a DP algorithm for computing an optimal $k$-clustering. That is, provide semantic and computational arrays for the problem (including any base cases). What is the time complexity of your algorithm?

Solution: We note that an optimal partitioning of the points into $k$ clusters is equivalent to a set of non overlapping sets of consecutive points. If $C = \{x_k, x_{k+1}, \ldots, x_\ell\}$ is a cluster with $k \leq \ell$, then $\text{diam}(C) = x_\ell - x_k \geq 0$. We define the following semantic array:

$D[j, s] =$ cost of an optimal $j$-clustering containing the points $x_1, \ldots, x_s$. Here the cost of a $j$-clustering into $C_1, \ldots, C_j$ is $\max_{1 \leq i \leq j} \text{diam}(C_i)$.

The desired answer is $D[k, n]$. Here is an equivalent appropriate computational array:

Basis: $D[1, s] = x_s - x_1$ for all $s$

Inductively: for $j > 1$, $D[j, s] = \min_{r : j \leq r \leq s} \max \{D[j - 1, r - 1], x_s - x_r\}$

We reason informally as follows, in the base case(s) where only one cluster is allowed, the diameter of this cluster is the distance from $x_1$ to $x_s$. When $j > r$ (i.e. more clusters than points), the cost is zero. We could When $j > 1$, an optimal clustering ending at point $x_s$ has some rightmost cluster which begins at some $x_r$ with $r \leq s$. The $x_s - x_r$ term represents the diameter of the last cluster and the $D[j - 1, r - 1]$ term represents an optimal clustering (with one less cluster) of the points to the left of $x_r$. Note: When $j > r$ (i.e. more clusters than points), the cost is zero. We could put this into the base case but instead, we are only considering $j \leq r$ in the computational array.
2. Consider the following triangulation problem. The input is a sequence \(< z_0, z_1, \ldots, z_{n-1} >\) of points in the Euclidean plane where this sequence forms a convex polygon \(P\). A triangulation of \(P\) partitions \(P\) into \(n-2\) triangles \(T_0, T_1, \ldots, T_{n-3}\) where \(T_i\) has edges \(< z_i, z_{i+1(mod n)}, z_{v(i)} >\) where \(v(i) \notin \{i, i+1(mod n)\}\). The cost \(c(T_i)\) of a triangle \(T_i\) is its perimeter, the Euclidean length of the sides of the triangle and the cost of a triangulation is the sum of the triangle costs. Show how to use dynamic programming to compute an optimal cost triangulation.

Solution: As the problem definition indicates, for every edge \(z_i, z_{i+1(mod n)}\) we need a third point on the polygon to complete a triangle \(T_i\). Think of the points being numbered so that the polygon is travelled clockwise as \(i\) increases. We can start with any edge, say formed by the vertices \(z_0, z_1\), and consider an optimal triangulation where \(z_{v(0)}\) is the third vertex in the triangle \(T_0\). Now when we remove \(T_0\) from the given polygon we are left with either one polygon or two polygons. The first case happens when \(v(0)\) is either 2 or \(n - 1\), otherwise the latter case happens. (If the polygon had only 4 or 5 vertices, then the first case has to occur.) For simplicity let me consider the case when \(n \geq 6\) and removing \(T_0 =< z_0, z_1, z_{v(0)} >\) results in two polygons, namely \(< z_1, z_2, \ldots, z_{v(0)} >\) and \(< z_0, z_{v(0)}, z_3, \ldots, z_{n-1} >\). Then if \(v(0)\) was chosen to be in an optimal triangulation, we have its perimeter cost plus the cost to triangulate the remaining two polygons. This leads to the following semantic array:

\[ C[i, j] = \text{the optimal triangulation of the polygon determined by } z_i, z_{(i+1) mod n}, \ldots, z_{j mod n} \text{ with } j \geq i + 2. \]

(Dont be distracted by the occurrences of “mod n”. We need something like this to formalize things but it is just used to describe the polygon by a sequence of points moving (say) clockwise around the polygon.

The desired cost is \(C[0, n - 1]\).

We have the following computational array:

\[ C[i, i+2] = d(z_i, z_{(i+1) mod n}) + d(z_{(i+1) mod n}, z_{(i+2) mod n}) + d(z_{(i+2) mod n}, z_i) \text{ where } d \text{ is the Euclidean distance.} \]

For \(j > i+2\),

\[ C[i, j] = \min_{k:i < k < j}[C(i, k) + C(k, j) + d(z_i, z_{(i+1) mod n}) + d(z_{(i+1) mod n}, z_k) + d(z_k, z_i)] \]