Assignment 1 – Comp 252

Alexandre St-Aubin and some other guy

January 24, 2024

1. Algorithm Design. You are given n vectors $x_1, ..., x_n$ in $\mathbb{Z}^n$. Design an efficient algorithm in the ram model for computing for each x_i one of its nearest neighbors among the other points, using the standard Euclidean metric to measure distances. You can't use real numbers, and operations like square root are not available. Nevertheless, show how this can be done in $O(n^3)$ worst-case time.

   Solution:
   We first note that minimizing the eucledian distance between 2 vectors of length n is equivalent to minimizing the absolute difference between each component of the vectors. Let $x_i = (x_{i,1}, ..., x_{i,n})$, $x_j = (x_{j,1}, ..., x_{j,n})$ both in $\mathbb{Z}^n$, and define

$$d_1(x_i, x_j) := \sum_{k=1}^{n} |x_{i,k} - x_{j,k}|$$

   It is obvious that $d_1 \sim O(n)$, and that no real numbers, nor square roots were used to compute it. For any given x_i, $1 \leq i, \leq n$, our goal will be to find the closest vector x_j to x_i, denote it

$$f_{min}(x_i) := \left\{ x_j \mid d_1(x_j, x_i) = \min_{1 \leq j \leq n, j \neq i} d_1(x_j, x_i) \right\}$$

   Now, let $G = K_n$ be the complete graph on n vertices, and let each vertex represent a vector x_i. We want to assign a weight to each edge of G, which will represent the distance between its incident vertices. That is, the weight of the edge $x_i x_j$ will be $d_1(x_i, x_j)$. By the handshaking lemma, G has $\frac{n(n-1)}{2}$ edges, so we only need to compute as many weights (distances).

   We associate an ordered list of distances to each vertex in the graph. The distances in the list will represent only those between vertices incident to the one whose list it is. In order to keep track of the closest vector, we also add a reference from each distance, to the vector it represents. Each time we compute one of the $\frac{n(n-1)}{2}$ edge distances, we add it to the lists of both vertices incident to that edge. At the end of the process described above, for each x_i its closest neighbour will be the first one on its list of distances.

   As for the time complexity of the process, d_1 is $O(n)$, and since we need to compute $\frac{n(n-1)}{2}$ distances, it follows that the overall time complexity is

$$T_n = n \cdot \frac{n(n-1)}{2} = \frac{n^3 - n^2}{2} = O(n^3)$$