CS 594: Representations in Algorithm Design **Spring 2022**

Lecture 24: April 7, 2022

Lecturer: Xiaorui Sun *Scribe: Shivali Singh*

1 Last time and today

Previously:

- Introduction to the problem of large matrix multiplication
- Approximation algorithm using tensor rank
- Schonhage's Theorem
- Analyzed the use of algebraic structure for fast matrix multiplication

Today's lecture:

- Geometric data
- Well Separated Pairs Decomposition (WSPD)
- Quad trees

2 Geometric Data

2.1 Introduction

In this course we have seen Graph and Matrix representation and their use in problems such as creating spanners (edge and vertex spanners), partitioning with minimum residue, short cycle decomposition and various others. The data structure used becomes especially important when the data is very large, the input is dynamic and certain property of the data has to be preserved. Data representation can be done more efficiently if additional properties of the data are known. Geometric data representation takes advantage of the information about the location of points in space.

Definition: *Geometric data is a spatial data type that can be represented using a set of points P in space.*

$$P = \{x : x \in \mathbb{R}^d\}$$

The number of points in the set P is $|P| = n$ where n is the total number of data points or nodes in a graph.

Spacial data can also be understood as a vector in a d-dimensional space. If d = 3, the data is geometric data that can be represented in a 3-d space.

2.2 Use of geometric data

2.2.1 Shortest path problem

Geometric data can be used to represent location of places on Earth. A city can be visualized as a set of points, connected by streets that act as edges of a graph. Length of the street determines the distance between two points in the city and two streets intersect at a crossing. As distance is related to location, points that are further apart will have more crossings.

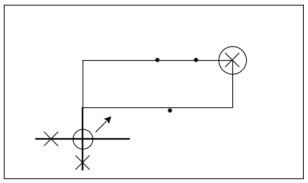

Figure 1: Eliminating paths that will not lead to shortest path

To find the shortest distance between two points we can use standard graph algorithms known to us but by using additional geometric properties we can avoid checking each vertex. If the location of the points are known we can ignore nodes that lead to a path in the wrong direction. Thus, concentrating only on nodes that could possibly be in the shortest path makes the algorithm faster.

2.2.2 Other applications

Clustering problem and Routing problem can also be solved using standard graph algorithm but their runtime can be improved upon by using geometric properties of the data.

2.3 Representation

Geometric data is represented using a graph as $G_p = (P, E, w)$, where E is set of edges, $P \times P$ and w is the weight or distance between points x and y belonging to P. Weight is defined as $w(x, y) = dist(x, y)$ where distance between vectors is calculated using the Frobenius norm as follows

$$||x - y||_2 = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$$

$$||x - y||_1 = \sum_{i=1}^{d} |x_i - y_i|$$

Observation: *There is no redundant information stored as in a graph created from geometric data the pairwise distance for each pair of points is preserved. Therefore, Encoding all n data points will require* $\Theta(n^2)$ *space.*

For a low-dimensional space with small enough value of d, such as in case of 3-d where value of d = 3 we can encode the approximate distance between n points in $\Theta(n)$ using WSPD.

3 $\frac{1}{\epsilon}$ Well Separated Pair Decomposition(WSPD)

For many problems, either NP-hardness or "curse of dimensionality" rules out an efficient algorithm if we insist on an optimal solution. In those cases, a natural approach is to design approximation algorithms, which are guaranteed to run fast and produce solutions provably close to optimal. Encoding the exact distance between each point requires substantial space but for most algorithms knowing the approximate distance is sufficient. This is done using WSPD which is aimed at representing distances efficiently.

Here we can see that points that are closer together belong to one group and points that are far apart belong to different groups. The distance between any two points can be estimated as the distance between the two groups is known. The points in one group are all so close to each other that there distance is considered negligible in comparison to distance between two different groups. So we can thus approximate that the distance between a point in one cluster and all the points in another cluster is almost equal.

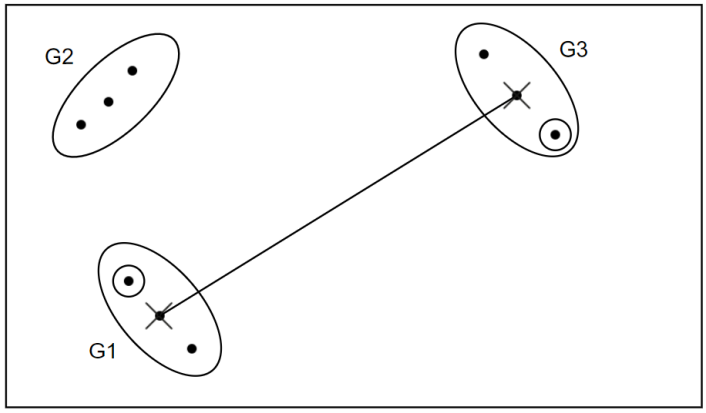

Figure 2: Clusters G1, G2, G3 that are sufficiently far away from each other and each have points relatively close to each other

For example if point x and x' belong to G1 and y and y' belong to G3 then dist(x,y) $\approx dist(x', y')$.

3.1 Conditions for WSPD

Let P be a set of n points in $\mathbb{R}^d$, and $\frac{1}{\epsilon}$ a parameter. One can represent all distances between points of P by explicitly listing the $\binom{n}{x}$ pairwise distances. Of course, the listing of the coordinates of each point gives us an alternative more compact representation (of size dn), but its not a very informative representation. We are interested in a representation that will capture the structure of the distances between the points. What we seek is a structure that allows us to encode the distance information of $\Omega(n^2)$ pairs in a structure of size only O(n).

A Set of pairs $\{(A_i, B_i)\}$ is Well Separated if they satisfy the following:

1. $A_i, B_i \subseteq P$ for all values of i.
2. $A_i \cap B_i \ = \phi$
3. $\bigcup_{i=1} A_i \times B_i \ = P \times P$ and $A \times B = \{(x, y) : x \in A, y \in B\}$
4. A_i, B_i are $\frac{1}{\epsilon}$- separated for i from 1 to n.

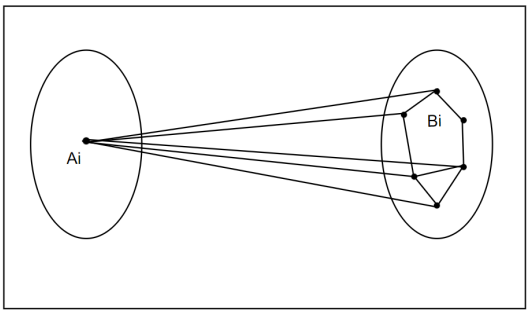

Figure 3: Two clusters A and B that are very far are shown here

Two points are $\frac{1}{\epsilon}$ well separated if :

$$max\{diam(A_i), diam(B_i)\} \leq \epsilon.d(A_i, B_i)$$

and d is the min d(p, q) where p in A_i and q in B_i.

Definition: (WSPD) For a point set P, a well-separated pair decomposition (WSPD) of P with parameter $1_{\frac{}{\epsilon}}$ is a pair decomposition of P with a set of pairs $W = \{\{A_1, B_1\}, ..., \{A_s, B_s\}\}$ such that, for any i, the sets A_i and B_i are $\frac{1}{\epsilon}$-separated.

3.2 Finding the Closest Pair

In this problem we are given a set of points $P \subseteq \mathbb{R}^d$ as input and the goal is to find the pair of points closest to each other. The output is x, y $\in P$ such that d(x,y) is minimized.

3.2.1 In one dimension

The problem of finding the closest pair in 1 dimension can be simplified as finding the two nearest neighbours in a set of points lying on a straight line. It can be solved by simply sorting the points and comparing the distance between a point and its neighbouring point and storing the minimum distance found so far.

3.2.2 In two dimensions

In 2 dimensions the set of points are spread in a 2-d planner surface. This problem is much more complex now that we have to sort the points in the horizontal and vertical direction to implement the above described strategy. There are a few existing algorithms that can find the solution to this problem in $O(nlogn)$ time complexity.

3.2.3 In multi-dimension

As we have seen the problem becomes significantly more difficult as the dimensions are increased, so for a d-dimensional space finding the closest pair of points we need a better strategy to encode the distances of the graph. This is done using WSPD, a structure that captures the approximate distance between all the points efficiently.

3.3 Algorithm for closest pair distance

- Initialise $dist \leftarrow \infty$
- For pairs $(A_i, B_i) \in \frac{1}{\epsilon}$ WSPD where $\frac{1}{\epsilon} > 1$.

if $(|A_i| > 1) or (|B_i| > 1)$ **then**
$pass$
else
$dist \leftarrow min\{dist, dist(p, q)\}$

- Output dist

The output is the minimum distance between two points and thus we can find the pair of closest points.

3.3.1 Number of pairs in WSPD

For a d dimensional space the number of pairs is at most

$$|\frac{1}{\epsilon} - WSPD| = O(n.log(\Phi(P))\frac{1}{\epsilon}^d)$$

where p, q ∈ P and Φ(P) is given as,

$$\Phi(P) = \frac{max\_dist(p, q)}{min\_dist(p, q)}$$

The number of pairs in a WSPD can be simplified to get,

$$|\frac{1}{\epsilon} - WSPD| = O(n.log(n).\frac{1}{\epsilon}^d)$$

In order to be able to construct such a structure we use quad trees which will be described in the section below.

4 Quad Trees

A quadtree is a hierarchical subdivision of space into regions, called cells, which are hyper-cubes. Initially, we have a single large hypercube containing all of P, and for simplicity (by scaling) we assume that this is the unit hypercube $[0, 1]^d$ which corresponds to the root of the tree. The decomposition begins by assuming that the points of P lie within a bounding hypercube. A quadtree is then constructed by a recursive process.

Consider any unprocessed cell and its associated node u in the current tree. If this cell contains either zero or one point of P, then this is declared a leaf node of the quadtree, and the subdivision process terminates for this cell. Otherwise, the cell is subdivided into 2d hyper-cubes whose side lengths are exactly half that of the original hypercube. For each of these 2d cells we create a node of the tree, which is then made a child of u in the quadtree.

4.1 Quad trees for d = 2

In 2 dimensions we can say, quad trees are generated by dividing the square into smaller squares until it is empty or contains a single point. Each node in this tree either has exactly four children or is a leaf.

4.1.1 Construction of Quad Tree

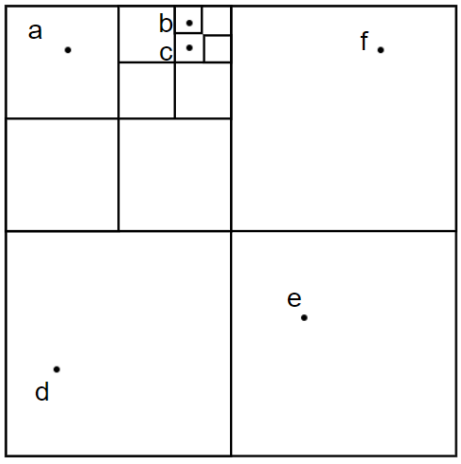

Figure 4: Division of square into subsquares for making the quad tree

We see the construction of quad tree in a 2 dimensional space bounded inside a region marked by a large square. Here, the set {a,b,c,d,e,f} is a set of points in space which have to be represented in the form of a quad tree are shown in the figure below.

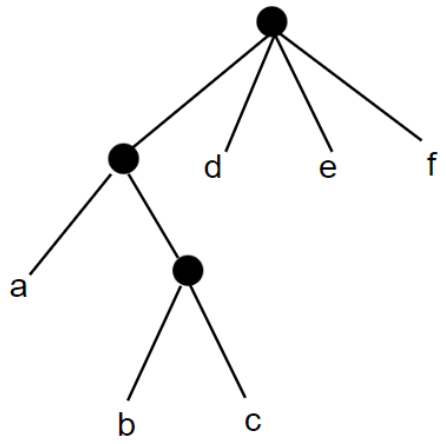

Figure 5: Tree generated

Note: The dot in the tree represents that the square contains least one point and has to be further divided.

4.2 Quad tree for d dimensions

- The upper bound of the depth of the quad tree generated by the algorithm is given as follows:

$$Depth \leq log(\frac{\Phi(P)}{\sqrt{d}})$$

- The time complexity of the algorithm is of the order

$$O(n.log(\frac{\Phi(P)}{\sqrt{d}}).2^d)$$

4.3 WSPD using Quad Tree

We see the algorithm for WSPD below for 2 squares S_1 and S_2.

Algorithm:
WSPD (S_1, S_2) :

if $diam(S_1) < diam(S_2)$ **then**

swap(S_1, S_2)

if $\text{diam}(S_1) \leq \epsilon \ . \ d(S_1, S_2)$ **then**

return(S_1, S_2)

else

WSPD (S_1', S_2) where S_1' is a sub-square of S_1

5 Next Lecture :

- Continuation of WSPD algorithm
- Other Representations of geometric graphs