CS 545

Finding the closest pair of points

Alon Efrat

Samir Khuller, Yossi Matias:

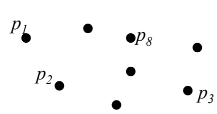
A Simple Randomized Sieve Algorithm for the Closest-Pair Problem

Inf. Comput. 118(1): 34-37 (1995)

Problem definition

Given: A set $S=\{p_1,\ldots p_n\}$ of n points in the plane

Problem: Find the pair $p_i p_j$ that minimizes $d(p_i,p_j)$, where $d(p_i, p_j)$ is the Euclidean distance between p_i and p_j.



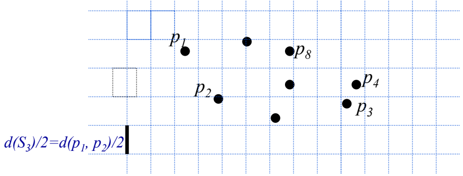
O(n^2) time algorithm – trivial

$\Omega(n \log n)$ bound for any deterministic algorithm.

In this talk – a randomized algorithm whose expected running time is $O(n)$

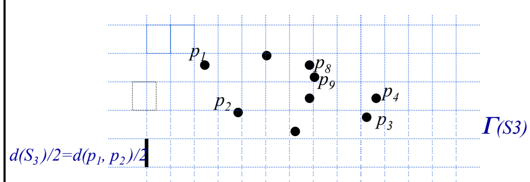
Notation

Let $S_i = \{p_1, p_2, \ldots p_i\}$
Let $d(S_i)$ denote the distance between the closest pair in S_i
Clearly $d(S_2) \geq d(S_3) \geq d(S_4) \geq \ldots \geq d(S_n)$
Idea – incremental algorithm – compute $d(S_{i+1})$ from $d(S_i)$



Let $\Gamma(S_i)$ denote an axis-parallel grid, where the edge-length of each grid-cell is $d(S_i)/2$, and one of its corner is on the point $(0,0)$

Properties of $\Gamma(S_i)$.

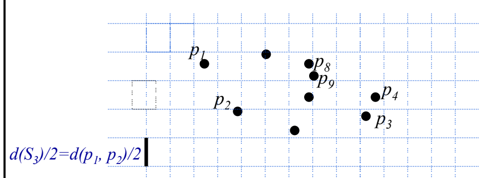


Claim 1: there is at most one point of S_i inside every cell of $\Gamma(S_i)$.

Proof – if there are two, then the distance between them is smaller than the length of the diagonal of the cell, which is

$$(\sqrt{2})d(S_i)/2 = d(S_i)/\sqrt{2} < d(S_i)$$

Locating points.

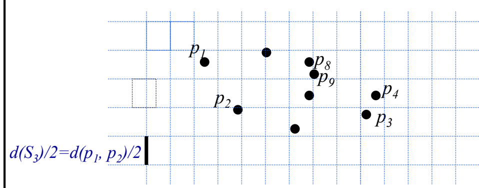


Claim 2: given $d(S_i)$ we can place all points of S_i in a data structure $H(S_i)$, such that we can (in $O(1)$ expected time)

1) insert a new point p_j
2) Given a query point q find if there is a point of S_i in the cell of $\Gamma(S_i)$ containing q.

The structure $H(S_i)$ is described in HW2 Question 7.

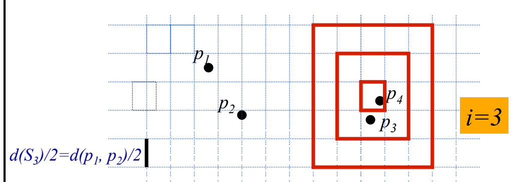
Rehashing with $d(S_i)$



Procedure **Rehashing with** $d(S_i)$:
Construct the hash table $H(S_i)$ (from HW1) with $d(S_i)$, and inserting all points of S_i into the table.

Expected time O(| S_i|).

Inserting p_{i+1} and Deciding if $d(S_i)>d(S_{i+1})$



To decide whether $d(S_i)>d(S_{i+1})$ or $d(S_i)=d(S_{i+1})$ do {

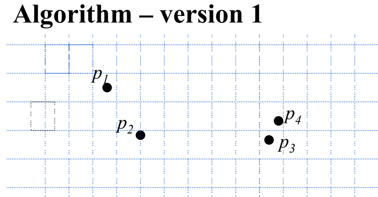
1) find all points of S_i in the cell containing p_{i+1}...
and in all the cells whose distance from this cell < $d(S_i)$.
2) Measure the distance from p_{i+1} to each of these points.

}

Note – only a constant number of cells, and due to Claim 1, only a constant number of points. Altogether: (expected) constant time .

Algorithm – version 1

Input: S
Output: $d(S)$, The closest pair of S



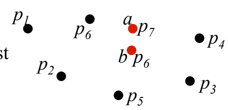
Find $d(S_2)$, and construct $H(S_2)$.
For $i=3,4...n-1$ do {
Use $H(S_i)$ to decide whether $d(S_i)>d(S_{i+1})$ or $d(S_i)=d(S_{i+1})$

If $d(S_i)=d(S_{i+1})$ then $\Gamma(S_{i+1}) = \Gamma(S_i)$ and $H(S_{i+1}) = H(S_i)$.
insert p_{i+1} ; No work is needed. (constant time)
Else /*$d(S_i)>d(S_{i+1})$*/ rehash with $d(S_{i+1})$. ($O(i)$ expected time)
}
Running time: Worst case $1+2+3+...(n-1) = O(n^2)$

Algorithm – version 2

Create random permutation of the points of S before calling the algorithm of version 1.

Assumption: the closest pair is unique.



Claim 3: The probability that $d(S_i) > d(S_{i+1})$ is $2/(i+1)$.
Proof: There are $i+1$ points, two are special (determining the closest pair). All permutations are equally likely, so the probability that one of the special pair appears last in the permutation is $2/(i+1)$.

Finishing the analysis

So in the i'th stage we are spending $O(1)$ time with probability $(i-1)/(i+1)$, and $O(i)$ time with probability $2/(i+1)$, so the expected work in this stage is $O(i)\ 2/(i+1) = O(1)$.

Hence the total expected time is $O(n)$.

Expected time

Let T_i denote the expected time at stage i. Then
$T_i=1$ with probability $(i-2)/i$ and $T_i=i$ with probability $2/i$

$$E(T_i) = \sum_{j=1}^{\infty} j \Pr(T_i = j) = 1 \Pr(T_i = 1) + i \Pr(T_i = i) = 3$$

Hence the expected total time is

$$E(\sum_{i=3}^{n} (T_i)) = \sum_{i=3}^{n} E(T_i) = \sum_{i=3}^{n} 3 = O(n)$$