CLOSEST PAIR VIA RANDOMIZATION

By: Alexandra Steau
Randomized Algorithm

• Set of inputs
• In addition to those inputs, the algorithm takes in random numbers while making random choices during execution
Las Vegas Algorithm

• Output:
  • Either gives the correct result or it produces an error
  • Does not gamble with correctness of the result

• Runtime:
  • polynomial in size of their input
  • Always finite

• i.e. Closest Pair via Randomization
Monte Carlo Algorithm

- **Output:**
  - Sometimes incorrect with a certain (usually small) probability

- **Runtime:**
  - Deterministic
  - Independent repetitions of Monte Carlo algorithms drive down the failure probability exponentially.
Is it possible to turn a Las Vegas Algorithm into a Monte Carlo?

• Yes, we can do so by:
  • Stopping the Las Vegas algorithm at a certain time.
  • If the algorithm has already terminated then we have a correct answer.
  • If it has not terminated, its output is correct with some probability.
Closest Pair via Randomization: The Problem

- **The Problem:**
  - **Give:**
    - A set of points, $P = \{p_1, p_2, \ldots, p_n\}$ in a plane
  - **Want:**
    - A pair of points, $\{p_1, p_2\}$, in $P$ which are closest together
    - That is to say: we must find points that minimize $d(p_1, p_2)$
Closest Pair: Brute Force

Algorithm:

\[
\text{minDist} = \infty \\
\text{for } i = 1 \text{ to } \text{length}(P) - 1 \\
\quad \text{for } j = i + 1 \text{ to } \text{length}(P) \\
\quad \quad \text{let } p = P_i, q = P_j \\
\quad \quad \text{if } d(p, q) < \text{minDist:} \\
\quad \quad \quad \text{minDist} = d(p, q) \\
\quad \quad \quad \text{closestPair} = (p, q) \\
\quad \text{return closestPair}
\]

Run time: \(O(n^2)\)
Closest Pair: Divide and Conquer

Algorithm:

1. Sort points according to their x-coordinates.
2. Split the set of points into two equal-sized subsets by a vertical line \( x = x_{\text{mid}} \).
3. Solve the problem recursively in the left and right subsets. This yields the left-side and right-side minimum distances \( d_{L\text{min}} \) and \( d_{R\text{min}} \), respectively.
4. Find the minimal distance \( d_{LR\text{min}} \) among the set of pairs of points in which one point lies on the left of the dividing vertical and the second point lies to the right.
5. The final answer is the minimum among \( d_{L\text{min}} \), \( d_{R\text{min}} \), and \( d_{LR\text{min}} \).

Run time: \( O(n\log n) \)
Closest Pair via Randomization

• Designing the Algorithm
  • Consider the points in random order
  • Keep the current closest pair and the minimal distance \((\delta)\) among the first points
    • Initially \(\delta = d(p_1, p_2)\)
  • Add another point, checking to see if it is closer than \(\delta\) to one of the existing points
  • When all points are added, output the closest pair of points and the distance

** Key: Adding a point must be done in expected constant time
Closest Pair via Randomization cont’d

• Testing a proposed distance
  • We want to assume that all points are within the unit square
  • Let’s suppose that points \( p_1,p_2,\ldots,p_{i-1} \) are already added and the current minimal distance between them is \( \delta \)
  • We will add the point \( p_i \)
  • We want to subdivide the unit square into \( \delta/2 \times \delta/2 \) squares
    \[
    S_{st} = \{ (x,y) \mid s\delta/2 \leq x < (s+1)\delta/2, \quad t\delta/2 \leq y < (t+1)\delta/2 \}\]
  • There are \( N^2 \), \( N = \lfloor 1/2\delta \rfloor \) of them
Closest Pair via Randomization cont’d

• Properties of subdivision
  • **Lemma**
    • If two points $p_1$ and $p_2$ belong to the same subsquare, then $d(p_1, p_2) < \delta$
  
  • Let us call subsquares $S_{st}$ and $S_{s't'}$ **close** if $|s - s'| \leq 2$ and $|t - t'| \leq 2$

  • **Lemma**
    • If for two points $p$ and $q$ we have $d(p_1, p_2) < \delta$, the subsquares containing them are close
    • **Key:** when checking proximity of points, it suffices to check at most 25 points
Closest Pair via Randomization cont’d

• Facts
  • For each point considered so far, we store the number of the subsquare it belongs to
  • Every time $\delta$ is updated, we update the subdivision and the number of subsquares associated with points

• Challenges
  1. Find which points are contained in the subsquares close to the subsquare containing the new point very quickly
  2. The number of times $\delta$ is updated must be in constant time
Closest Pair via Randomization cont’d

• To address challenge 1
  • Use hash tables
    • The universe U is the set of $N^2$ subsquares
    • The set S of elements in the dictionary is the set of subsquares occupied by points $p_1, p_2, \ldots, p_{i-1}$
    • Every subsquare in the dictionary also stores the index of the point it

• To address challenge 2
  • checking if a subsquare close to the subsquare with the new point contains a point and checking the distance to this point can be done in expected constant time
Closest Pair via Randomization cont’d

- Algorithm:
  
  Order points \( p_1, p_2, \ldots, p_n \) in a random order
  
  set \( \delta := d(p_1, p_2) \)
  
  MakeDictionary
  
  for \( i = 1 \) to \( n \) do
    
    find the subsquare \( S_{st} \) containing
    
    look up subsquares close to \( S_{st} \)
    
    find \( d(p_i, p) \) for each \( p \) in those subsquares
    
    if \( \delta' := d(p_i, p_j) < \delta \) for some \( p_j, \) \( j < i \) then do
      
      delete the current dictionary
      
      MakeDictionary
      
      for \( j = 1 \) to \( i \) do
        
        find the smaller subsquare \( S_{uv} \) containing \( p_j \),
        
        Insert( \( S_{uv} \) )
        
      endfor
    
    else Insert \( p_i \) into the current dictionary
    
  endfor
Closest Pair via Randomization Visual

- Points in random order
Closest Pair via Randomization Visual Cont’d

- Initialize the first two random points as Closest Pair
Closest Pair via Randomization Visual

- Consider the next point
Closest Pair via Randomization Visual

- Consider a max of 25 fields for points
Closest Pair via Randomization Visual

- Examine each square
Closest Pair via Randomization Visual

- Look at the points in the 5x5 range
Closest Pair via Randomization Visual

- Select the new Closest Pair in the 5x5 squares
Closest Pair via Randomization Visual

- Check other squares on inserted points
Closest Pair via Randomization Visual

• Still checking
Closest Pair via Randomization Visual

- We have found a new Closest Pair in the 5x5 grid
Closest Pair via Randomization Visual

- Mark the new closest pair
Closest Pair via Randomization Visual

- Still checking
Closest Pair via Randomization Visual

- Generating new dictionary with smaller subsquares
Closest Pair via Randomization Visual

- Add points in the new dictionary
Closest Pair via Randomization Visual

- THE END! We have found the closes pair via randomization
Closest Pair via Randomization cont’d

• Algorithm Analysis:
  • Correctly maintains closest pair at all times, performing at most:
    • $O(n)$ distance computations
    • $O(n)$ lookup operations
    • $O(n)$ MakeDictionary

• What about Insert?
  • Let $X_i$ be a random variable:
    • Equals 1 if when adding a point the distance changes and
    • Equals 0 otherwise
  • Let $x = x_1 + x_2 + \ldots + x_n$ be the number of updates of the dictionary

• Lemma
  • The total number of operations is $n + \sum_{i} iX_i$
Closest Pair via Randomization cont’d

- Algorithm Analysis Cont’d:
  - Lemma
    \[ \Pr[X_i = 1] \leq \frac{2}{i} \]
  - Proof
    - Let us consider the first \( i \) points \( p_1, p_2, \ldots, p_i \) in a random order
    - Suppose the min distance is achieved by \( p_1 \) and \( p_2 \)
    - The point \( p_i \) can cause the min distance to decrease only if:
      - \( p_i = p_1 \) or
      - \( p_i = p_2 \)
    - The probability of this is \( \frac{2}{i} \)
Closest Pair via Randomization cont’d

• Algorithm Analysis Cont’d:
  • **Theorem**
    • In expectation, the randomized closest pair algorithm required:
      • O(n) time + O(n) dictionary operations
  
  • Proof:
    • The number of insert operations:
      \[
      E[X] = n + \sum_i iX_i \leq n + \sum_i i^2 = 3n
      \]
    • We know that there are O(n) distance computation, O(n) lookup operations, O(n) MakeDictionary already.

• Thus, this algorithm runs in O(n) time
Time Complexity

- Brute force
- Divide and conquer
- Randomized
References:

- http://www.cs.arizona.edu/classes/cs545/fall09/closestpair.prn.pdf
- Chapter 13.7 in Kleinberg - Tardos book
- http://www.yovisto.com/video/9660
- http://bigocheatsheet.com/
- http://www.iitg.ac.in/rinkulu/algorithms/slides/lasvegas-closestpp.pdf
THANK YOU!