Computational Geometry

(csci3250)

Laura Toma

Bowdoin College

Finding collinear points

We'll start with a warmup problem:

Problem: Given a set of n points in 2D, determine if there exist three points that are collinear.

Come up with different solutions to this problem (and analyze/compare them).

Finding collinear points

Brute force:

- for all distinct triplets of points p_i, p_j, p_k
  - check if they are collinear

- Correct?
  - yes because it checks all triplets
- Worst-case running time:
  - n chose 3 = $\Theta(n^3)$ triplets
  - checking if three points are collinear can be done in constant time

  ==> $O(n^3)$ algorithm
- Space: O(1)

Via sorting

Algorithm 2

- initialize array L = empty
- for all distinct pairs of points p_i, p_j
  - compute their line equation (slope, intercept) and add it to an array L
- sort array L by (slope, intercept)
- traverse L and if you find any 3 consecutive identical (s,i) $\rightarrow$ collinear

- Correct?
  - if points a, b, c are collinear ==> (slope, intercept) of (a,b) (b,c) and (a,c) are the same
- Worst-case running time:
  - $\Theta(n^2) + sort(n^2) = \Theta(n^2 \lg n)$
- Space:
  - $\Theta(n^2)$ for L

With a binary search tree

Algorithm 3

- initialize BBST = empty
- for all distinct pairs of points p_i, p_j
  - compute their line equation (s, i)
  - insert (s,i) in BBST; if when inserting you find that (s,i) is already in the tree, you got three collinear points and return true
- (if you ever get here) return false

- Correct?
  - if points a, b, c are collinear ==> (slope, intercept) of (a,b) (b,c) and (a,c) are the same
- Worst-case running time:
  - using a balanced tree (like red-black tree, or AVL-tree, or…)
  - $\Theta(n^2)$ inserts => $\Theta(n^2 \lg n)$
- Space:
  - $\Theta(n^2)$ for BBST

With hashing

A hash table supports find(x), insert(x), delete(x)

Algorithm 4

- initialize HashTable = empty
- for all distinct pairs of points p_i, p_j
  - compute their line equation (s, i)
  - insert (s,i) in HashTable; if when inserting you find that (s,i) is already in the HT, you got three collinear points and return true
- (if you ever get here) return false

- Correct?
  - if points a, b, c are collinear ==> (slope, intercept) of (a,b) (b,c) and (a,c) are the same
- Worst-case running time:
  - $\Theta(n^2)$ searches & inserts => $\Theta(n^2)$ **If we assume O(1) for find(x).**
- Space:
  - $\Theta(n^2)$ for hash table

Hashing

```
List<T>[] t;
int n;
```

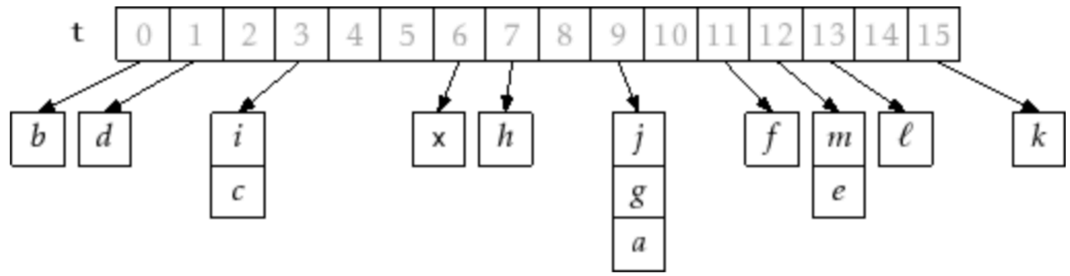


Figure 5.1: An example of a `ChainedHashTable` with $\mathtt{n} = 14$ and $\mathtt{t.length} = 16$. In this example $\mathtt{hash(x)} = 6$

Does find(x) run in O(1) ?

- Run time depends on how many other elements have same hash
- **O(1) on the average** assuming a good hash function (spreads the keys uniformly) and m = O(n). Worst-case is still O(n).
- **O(1) expected worst-case** can be achieved with universal hashing (by choosing the hash function uniformly at random from a set of universal hash functions, i.e. which guarantee no collision with high probability)

  Families of *universal hash functions* are known for integers

  can be extended to primitive types (char, float, string)

- Summary: does find(x) run in O(1) ?

  theory: O(1) expected, could be O(n) worst case

  O(1) approximately true for many real world situations

With hashing

Algorithm 4

- initialize HashTable = empty
- for all distinct pairs of points p_i, p_j
  - compute their line equation (s, i)
  - insert (s,i) in HashTable; if when inserting you find that (s,i) is already in the HT, you got three collinear points and return true
- (if you ever get here) return false

- In conclusion, this runs in $\Theta(n^2)$ **on the average, assuming a good hash function**

A different way to sort

Algorithm 5

- for every point p_i
  - set array L = empty
  - for every point p_j (with $p_j != p_i$)
    - compute slope of p_j wrt to p_i and add it to array L
  - sort L
  - traverse L and if you find two consecutive points that have same slope, they are collinear with p_i so return true
- (if you get here) return false

- Correct?
  - if points a, b, c are collinear ==> slope of b and c wrt a are equal
- Worst-case running time:
  - $n \times sort(n) = \Theta(n^2 \lg n)$
- Space:
  - $\Theta(n)$ for L

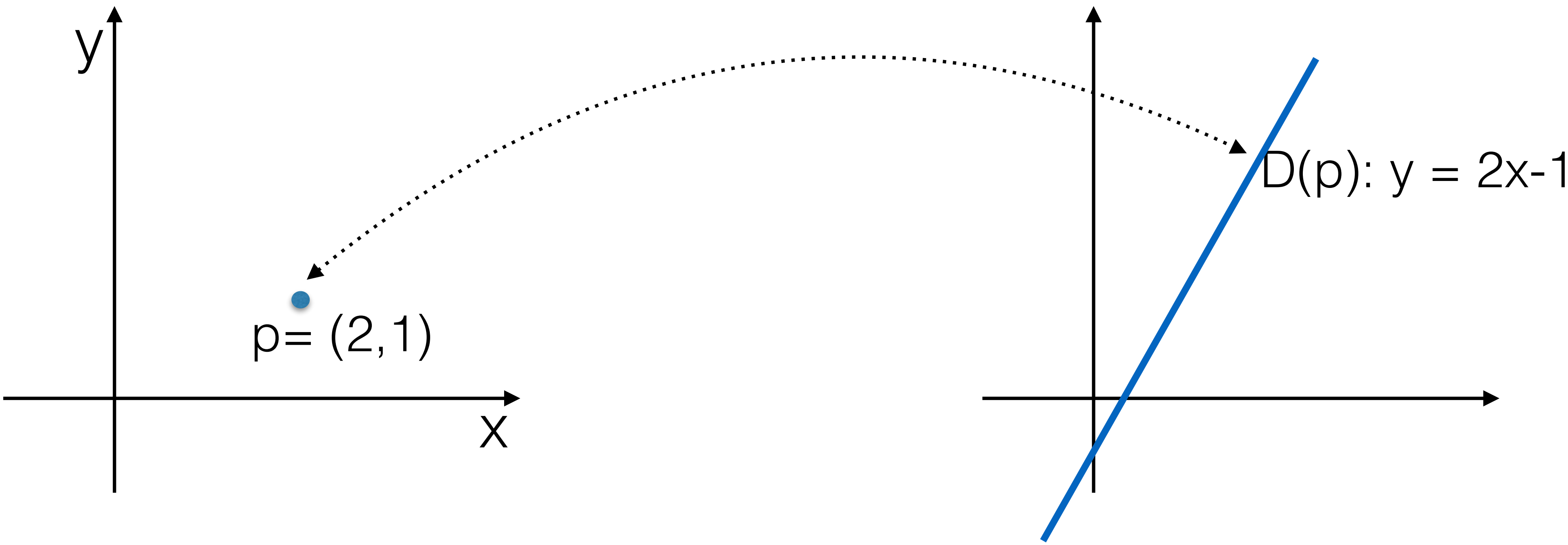
Summary

- Problem: Given a set of n points in the plane, determine if any are collinear.

- Algorithms
  - brute force: $O(n^3)$
  - via sorting: $O(n^2 \lg n)$ with $O(n^2)$ space
  - with BBST: same as above
  - hashing: $O(n^2)$ with $O(n^2)$ space assuming good hash function
  - smart sort: $O(n^2 \lg n)$ with O(n) space

Can we do better?

Duality transforms of points and lines in $\mathbb{R}^2$



Definition: The duality transform is defined as:

$$p = (a, b) \quad \cdots\cdots\blacktriangleright \quad D(p) : y = ax - b$$

$$l : y = ax - b \quad \cdots\cdots\blacktriangleright \quad D(l) : p = (a, b)$$

Write the duals for the following points

$$p = (1,1)$$

$$p = (3,5)$$

$$p = (-4,2)$$

$$p = (0,1)$$

Write the duals for the following lines:

$$y = 3x - 4$$

$$y = x - 1$$

$$y = 2x + 1$$

$$y = x$$

Properties

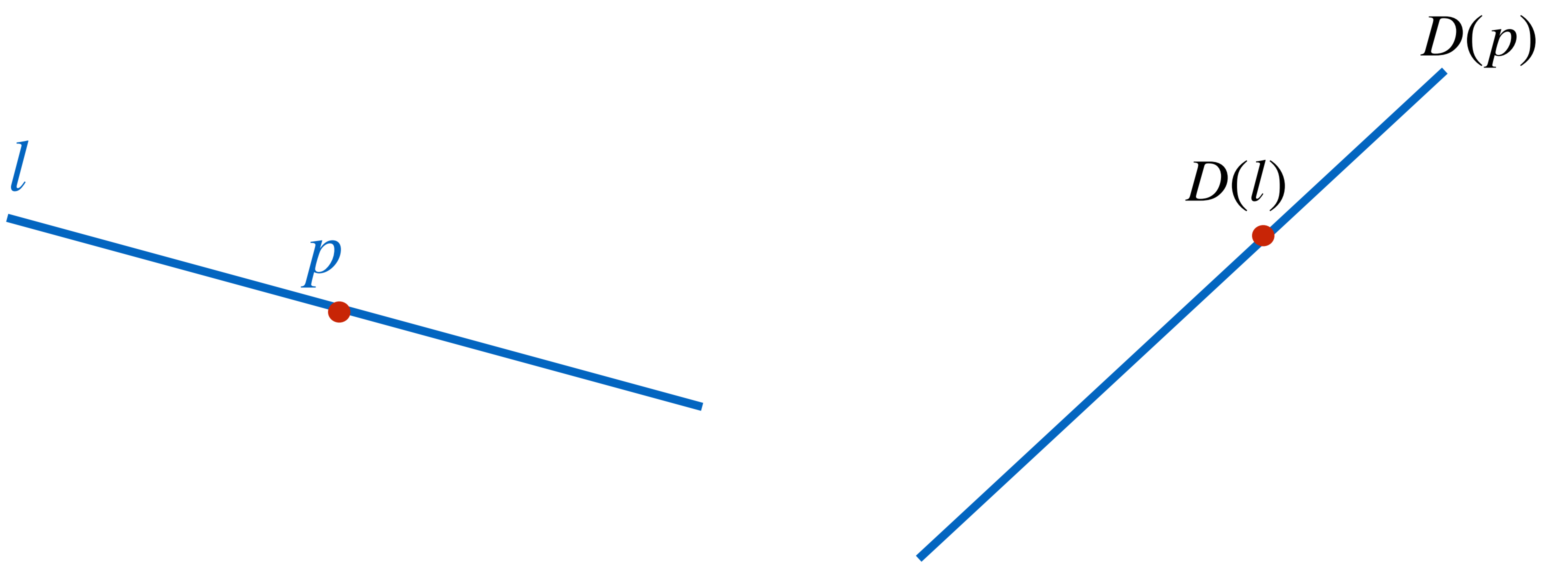
Lemma 1:

- $D(D(p)) = p$ and $D(D(l)) = l$

Properties

Lemma 2 [Incidence preserving]:

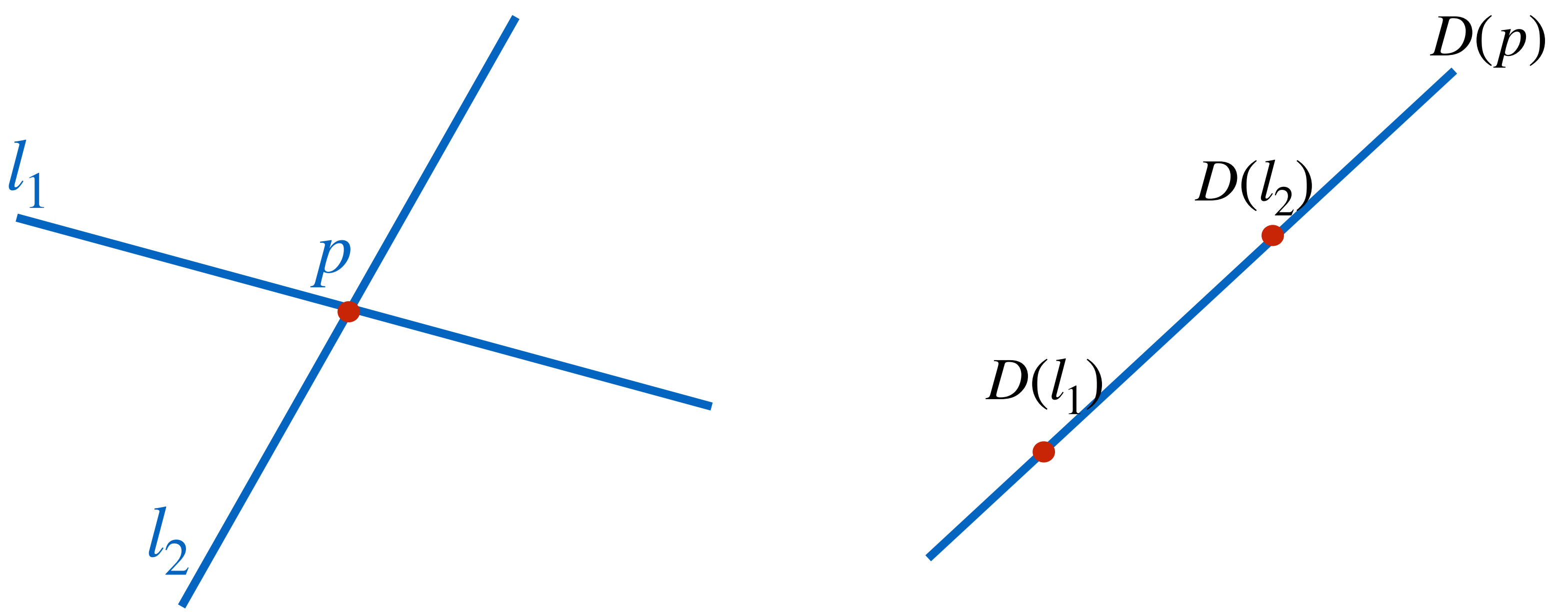
- If p lies on a line l, than $D(l)$ lies on $D(p)$



Properties

Lemma 3:

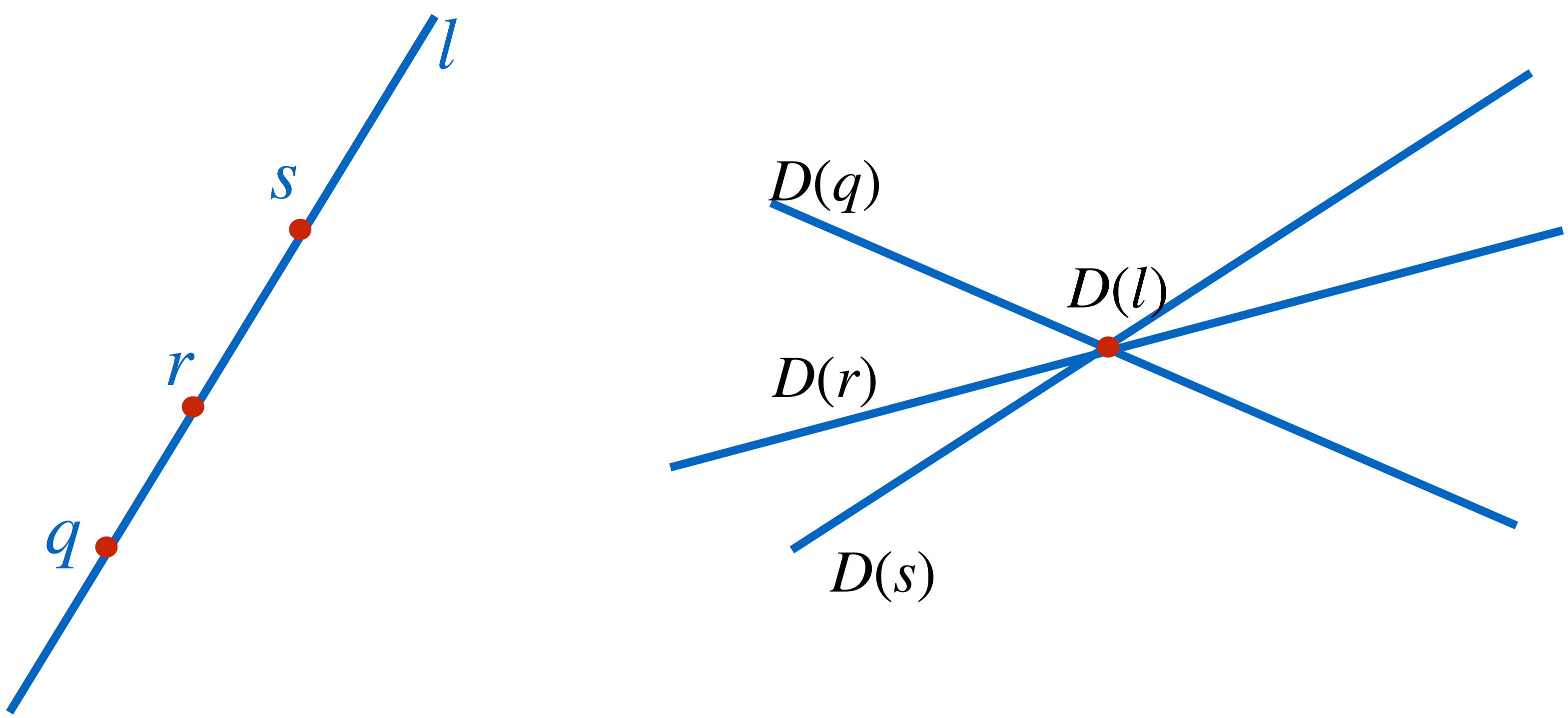
- l_1 and l_2 interect in point $p \iff D(p)$ passses through $D(l_1)$ and $D(l_2)$



Properties

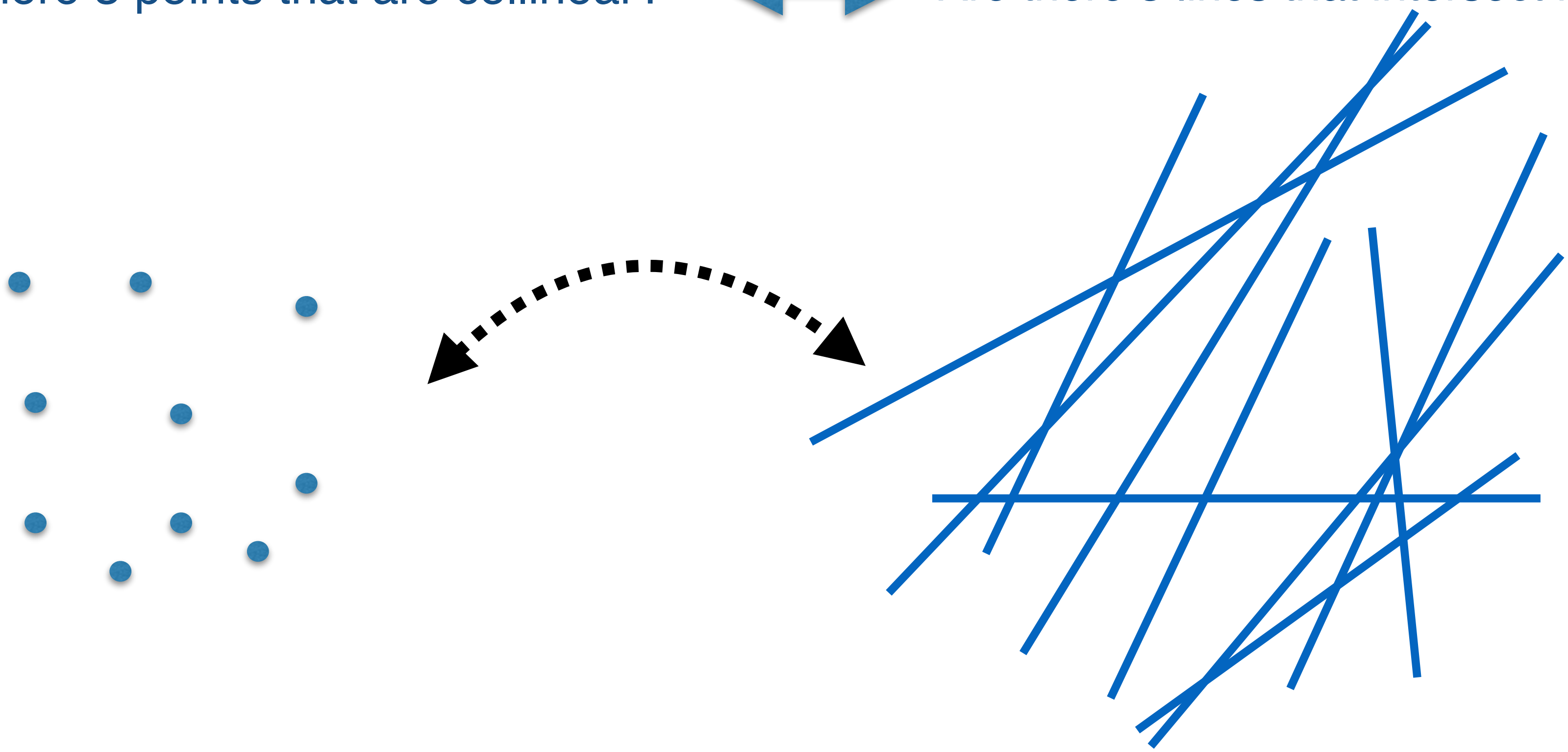
Lemma 4:

q, r, s collinear <==> $D(q), D(r), D(s)$ intersect in a common point



And now back to our problem

Are there 3 points that are collinear? ⟷ Are there 3 lines that intersect in one point?



It is known how to compute the intersections of n lines in

$O(n \lg n) + k = O(n^2)$, where $k = O(n^2)$ is the nb. of intersections

Summary

- Problem: Given a set of n points in the plane, determine if any are collinear.
- Algorithms
  - brute force: $O(n^3)$
  - via sorting: $O(n^2 \lg n)$ with $O(n^2)$space
  - with BBST: same
  - hashing: $O(n^2)$ with $O(n^2)$ space assuming good hash function
  - smart sort: $O(n^2 \lg n)$ with O(n) space
- And fastest solution: compute line intersections in the dual in $O(n^2)$