MA/CSSE 473 Day 15

Divide and Conquer

ClosestPoints

ConvexHull

MA/CSSE 473 Day 15

- HW 6 due now. HW 7 is available.
- Exam 1: Tomorrow, September 30
  - You may start at 7:30 if you wish
  - Chapters 1-3
  - Materials covered in-class through Sept 26
  - HW 1-6
- I asked you to comment on pace of the course …
- **Student Questions**
- Brute force algorithms
- Divide and Conquer

What is the brute force approach to

1. Calculate the n^{th} Fibonacci number?
2. Compute the n^{th} power of an integer?
3. Search for a particular value in a sorted array?
4. Sort an array?
5. Search for a substring of a string?
6. Find the maximum contiguous subsequence? in an array of integers?
7. Find the largest element in a Binary Search Tree?
8. Find the two closest points among N points in the plane?
9. Find the convex hull of a set of points in the plane?
10. Find the shortest path from vertex A to vertex B in a weighted graph?
11. Solve the traveling salesman problem?
12. Solve the knapsack problem?
13. Solve the assignment problem?
14. Solve the $n \times n$ non-attacking chess queens problem?
15. Other problems that you can think of?

Divide-and-conquer algorithms

- Definition
- List some examples seen prior to this course or so far in this course

Closest Points problem

- Given a collection, S, of N points, find the minimum distance between two points in S.
- For simplicity, we assume $N = 2^k$ for some k.
- Sort the points by x-coordinate.
  - If we use merge sort, the worst case is $\Theta(N \log N)$
- If two points have the same x-coordinate, order them by y-coordinate.

Closest Points problem

- Assume that the points of S are sorted by x-coordinate.
- Let c be the median x-value of the points
- Let S_1 be {(x, y): $x \leq c$}, and S_2 be {(x, y): $x \geq c$}
  - adjust so we get exactly N/2 points in each subset
- Let d_1 be the minimum distance between two points in S_1
- Let d_2 be the minimum distance between two points in S_2
- Let d = min(d_1, d_2). Is d the minimum distance for S?
- What else do we have to consider?
- Suppose we needed to compare every point in S_1 to every point in S_2. What would the running time be?
- How can we avoid doing so many comparisons?

After recursive calls on S_1 and S_2

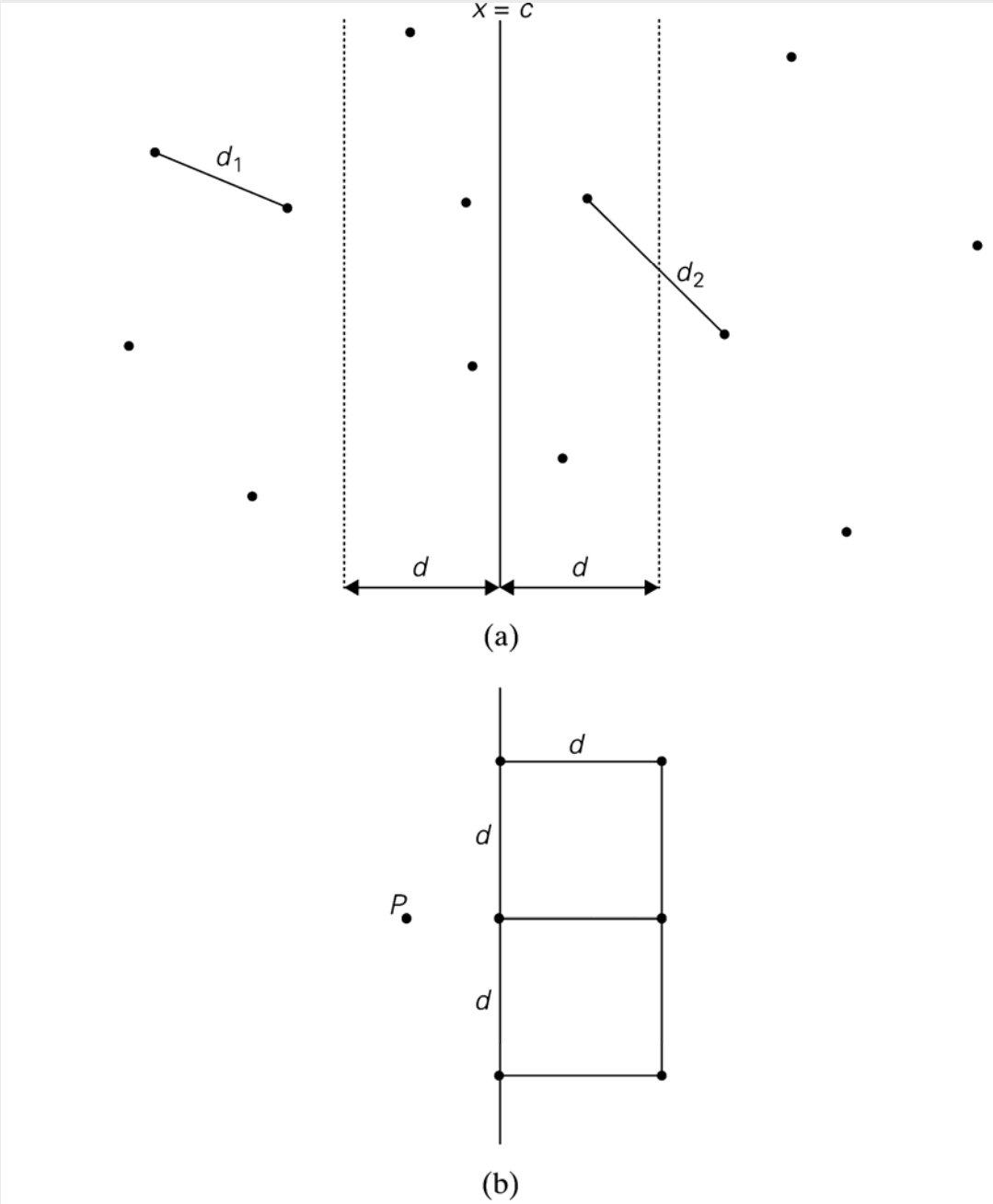

FIGURE 4.7 (a) Idea of the divide-and-conquer algorithm for the closest-pair problem. (b) The six points that may need to be examined for point P.

Convex Hull Problem

- Again, sort by x-coordinate, with tie going to larger y-coordinate.

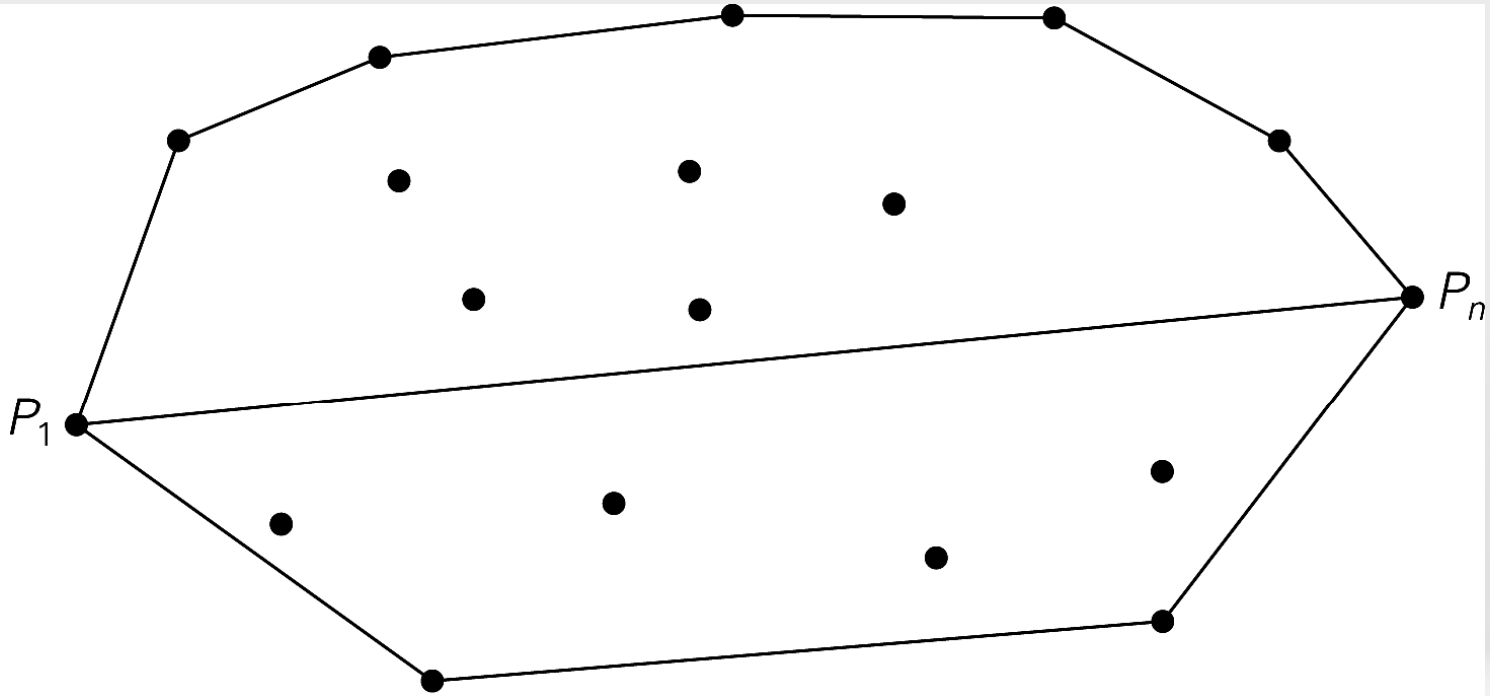

FIGURE 4.8 Upper and lower hulls of a set of points

Recursive calculation of Upper Hull

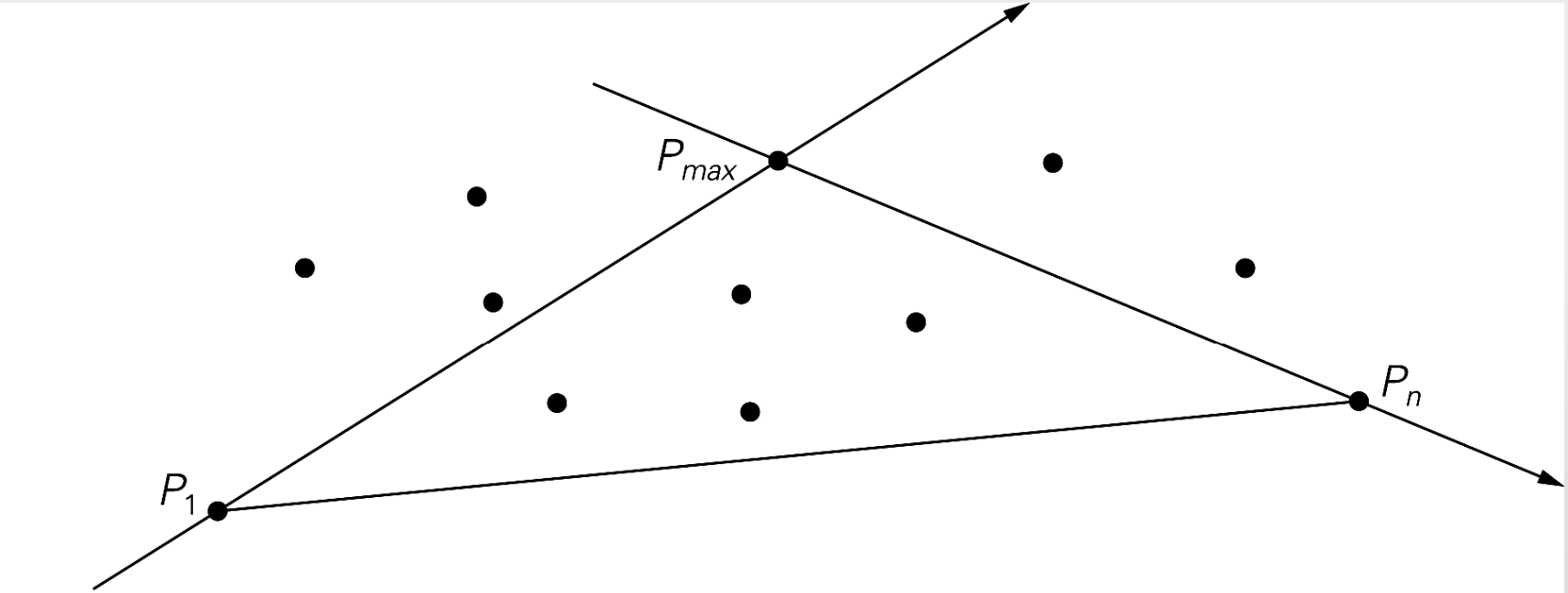

FIGURE 4.9 The idea of quickhull