CSE 421
Algorithms

Richard Anderson

Lecture 13

Divide and Conquer

Announcements

- HW 5 available
  - Deadline Friday, Nov 3.
- Midterm
  - Friday, Nov 3.
  - Material from lecture/text through end of chapter 5
  - 50 minutes, in class, closed book/notes, short answer, approximately five problems, problems easier than HW problems.

What you really need to know about recurrences

- Work per level changes geometrically with the level
- Geometrically increasing ($x > 1$)
  - The bottom level wins
- Geometrically decreasing ($x < 1$)
  - The top level wins
- Balanced ($x = 1$)
  - Equal contribution

$T(n) = aT(n/b) + n^c$

- Balanced: $a = b^c$
- Increasing: $a > b^c$
- Decreasing: $a < b^c$

Classify the following recurrences (Increasing, Decreasing, Balanced)

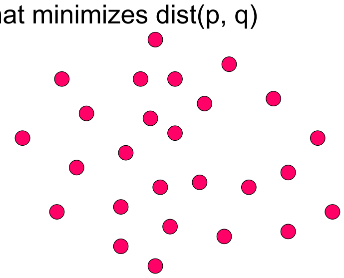
- $T(n) = n + 5T(n/8)$
- $T(n) = n + 9T(n/8)$
- $T(n) = n^2 + 4T(n/2)$
- $T(n) = n^3 + 7T(n/2)$
- $T(n) = n^{1/2} + 3T(n/4)$

Divide and Conquer Algorithms

- Split into sub problems
- Recursively solve the problem
- Combine solutions
- Make progress in the split and combine stages
  - Quicksort – progress made at the split step
  - Mergesort – progress made at the combine step

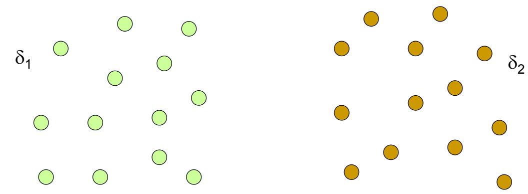
Closest Pair Problem

- Given a set of points find the pair of points p, q that minimizes dist(p, q)



Divide and conquer

- If we solve the problem on two subsets, does it help? (Separate by median x coordinate)



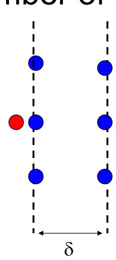
Packing Lemma

Suppose that the minimum distance between points is at least δ, what is the maximum number of points that can be packed in a ball of radius δ?

Combining Solutions

- Suppose the minimum separation from the sub problems is δ
- In looking for cross set closest pairs, we only need to consider points with δ of the boundary
- How many cross border interactions do we need to test?

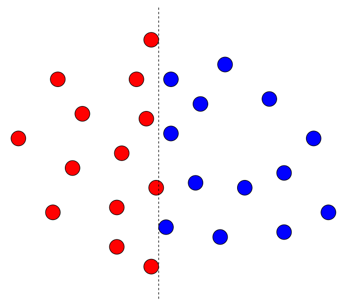
A packing lemma bounds the number of distances to check



Details

- Preprocessing: sort points by y
- Merge step
  - Select points in boundary zone
  - For each point in the boundary
    - Find highest point on the other side that is at most δ above
    - Find lowest point on the other side that is at most δ below
    - Compare with the points in this interval (there are at most 6)

Identify the pairs of points that are compared in the merge step following the recursive calls



Algorithm run time

- After preprocessing:
  - $T(n) = cn + 2\ T(n/2)$

Inversion Problem

- Let $a_1, \dots a_n$ be a permutation of $1 \dots n$
- (a_i, a_j) is an inversion if $i < j$ and $a_i > a_j$

4, 6, 1, 7, 3, 2, 5

- Problem: given a permutation, count the number of inversions
- This can be done easily in $O(n^2)$ time
  - Can we do better?

Application

- Counting inversions can be use to measure how close ranked preferences are
  - People rank 20 movies, based on their rankings you cluster people who like that same type of movie

Counting Inversions

11	12	4	1	7	2	3	15	9	5	16	8	6	13	10	14

Count inversions on lower half

Count inversions on upper half

Count the inversions between the halves

Count the Inversions

11	12	4	1

7	2	3	15

9	5	16	8

6	13	10	14

11	12	4	1	7	2	3	15

9	5	16	8	6	13	10	14

11	12	4	1	7	2	3	15	9	5	16	8	6	13	10	14

Problem – how do we count inversions between sub problems in O(n) time?

- Solution – Count inversions while merging

1	2	3	4	7	11	12	15

5	6	8	9	10	13	14	16

Standard merge algorithms – add to inversion count when an element is moved from the upper array to the solution

Use the merge algorithm to count inversions

1	4	11	12

2	3	7	15

5	8	9	16

6	10	13	14