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 $\text{dist}(p, q)$

**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 $\delta$, what is the maximum number of points that can be packed in a ball of radius $\delta$?

**Combining Solutions**

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

**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 $\delta$ above
    - Find lowest point on the other side that is at most $\delta$ 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:
  – \( T(n) = cn + 2T(n/2) \)

Inversion Problem
• Let \( a_1, \ldots, a_n \) be a permutation of \( 1 \ldots n \)
• \((a_i, a_j)\) 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 used to measure how close ranked preferences are
  – People rank 20 movies, based on their rankings you cluster people who like the same type of movie

Counting Inversions

Count the Inversions

Count inversions on lower half
Count inversions on upper half
Count the inversions between the halves
Problem – how do we count inversions between sub problems in \( O(n) \) time?

- Solution – Count inversions while merging

Use the merge algorithm to count inversions

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