Algorithms and Complexity
UCLA-CS180-S18

Quentin Truong
Taught by Professor Meka

Spring 2018

Contents

1 Algorithm Design: Ch5: Divide and Conquer

1.1 Introduction

- Divide and conquer
  - Class of algorithmic techniques in which one breaks input into several parts, solves subproblems recursively, and recombines into overall solution
- Analyze running time
  - Generally will use recurrence relations
  - For divide and conquer problems, the brute force solution is typically polynomial; we are trying for a lower polynomial

1.2 A First Recurrence: The Mergesort Algorithm

- Mergesort
  - Sort list of numbers by dividing into two equal halves, solving each half recursively, and recombining solutions
  - $T(n) \leq 2T(n/2) + O(n)$
  - Ignore ceiling and floor issues for odd numbers, b/c not actually impactful
- Approaches to Solving Recurrences
  - Unrolling the recurrence into a graph allows us to see how many operations are performed at each level
    - Identify the pattern, and sum over all levels
  - Substituting a solution into the recurrence
    - Requires a guess
  - Partial substitution can determine the constants
    - Useful for determining exact constants if we know the general form of the solution

1.3 Further Recurrence Relations

- 5.3 $T(n) \leq qT(n/2) + cn$
- 5.4 Any function T() satisfiyng 5.3 with q >2 is bounded by $O(n^{\log_2 q})$
- 5.5 Any function T() satisfying 5.3 with q=1 is bounded by O(n)
- 5.6 $T(n) \leq 2T(n/2) + cn^2$

1.4 Counting Inversions

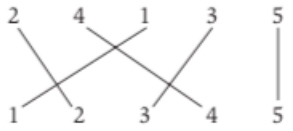

Figure 5.4 Counting the number of inversions in the sequence 2, 4, 1, 3, 5. Each crossing pair of line segments corresponds to one pair that is in the opposite order in the input list and the ascending list—in other words, an inversion.

- Application of Counting Inversions
  - Analysis of rankings
    - Collaborative filtering, to match preferences to those of other people on the Internet

    - Recommend things according to what other similar people like
  - Meta-search tools
    - Execute same query on different search engines
    - Measure how "out of order" the different orderings are
  - Generally
    - Given a sequence of distinct numbers, measure how far the list is from being in ascending order
    - Number should increase as more scrambled
- More formal definition of Counting Inversions
  - Count the number of indices i <j which form an inversion, $a_i > a_j$
- Algorithm
  - Divide list in two halves, recursively sort and count inversions for each
  - To recombine two halves, take min of each sorted half
  - If take min from first half, no inversions are added
  - If take min from the second half, then add the number of remaining elements from the first half to the inversion count

1.5 Finding the Closest Pair of Points

- 1D Algorithm
  - Sort; min must be adjacent to each other, so traverse
- 2D Algorithm
  - P_x are the points sorted with respect to x
  - P_y are the points sorted with respect to y
  - L is the line dividing P_x in half
  - Q are first half of points in P_x
  - R are second half of points in P_x
  - Recursively determine closest pair of points in Q; then in R
  - $\delta = \min(d(q_0^*, q_1^*), d(r_0^*, r_1^*))$
  - If there exists q ∈ Q and r ∈ R for which d(q, r) $<\delta$, then each of q and r lies within a distance of δ of L
  - If s, s′ have d(s, s′), then s and s′ are within 15 positions of each other in the sorted S_y
    - Each box has sidelength (δ/2), and at most one point (b/c if had more than one point, *delta* would be different)
    - 16 boxes leads to 16 possible positions
    - Can reduce down to 5 possible positions using packing arguments
    - Linear time to try/fail find a position in S
  - Bruteforce for $P \leq 3$
  - Satisfies recurence found in 5.1 to achieve O(nlogn)

1.6 Integer Multiplication

- Multiplication
  - Adding up partial products works the same in base-10 as in base-2
- Karatsuba's Algorithm
  - Based on breaking up partial sums
  - $xy = (x_1 * 2^{n/2} + x_0)(y_1 * 2^{n/2} + y_0$
  - $= x_1y_1 * 2^n + (x_1y_0 + x_0y_1) * 2^{n/2} + x_0y_0$
  - This achieves $T(n) \leq 4T(n/2) + cn = O(n^2)$
  - $(x_1 + x_0)(y_1 + y_0) = x_1y_1 + x_1y_0 + x_0y_1 + x_0y_0$
  - Subtract away x_1y_1 and x_0y_0
- Analysis
  - $T(N) \leq 3T(n/2) + O(n)$
  - $O(n^{log_2 3})$

1.7 Convolutions and the Fast Fourier Transform

- Convolution
  - Take two vectors of length n and produce a vector of 2n - 1 coordinates
  - $\sum_{(i,j):i+j=k;i,j<n} a_i b_j$
  - Many applications, especially in signal processing
- Fast Fourier Transform
  - Can calculate convoution in nlogn
  - Product of polynomials is equivalent to convolution
  - Reconstruct C from values on the $(2n)^{th}$ roots of complex roots of unity
  - Coefficients of C are coordinates of convolution vector

2 Ch1:

2.1 Introduction

–