

CSE 101 Homework 2
Winter, 2005, Due Feb. 15
Divide-and-Conquer
100 points total = 10 %

Binary Conversion: 20 pts . In the calibration homework, we found an $O(n^2)$ algorithm for converting a decimal integer to binary, where the basic operations involved single digits. Present and analyze a divide-and-conquer algorithm that does better. Use the faster integer multiplication algorithm, *Mult*, shown in class, which runs in time $O(n^{\log_2 3})$, as a subroutine.

Median of two sorted lists: 20 pts. The *median* of a set of numbers is an element of that set so that half the elements (round down) are less than that number and half are at least as large as that number. Present and analyze a divide-and-conquer algorithm that, given two sorted arrays of distinct integers, $A[1..n], B[1..n]$, returns the median (n 'th largest) of the set of elements that appear in either list.

Triangles: 20 pts On the calibration homework, we saw an $O(nm)$ algorithm to compute whether a graph G had a triangle, three distinct nodes x, y, z so that any two were connected by an edge in G . For large m , this is $O(n^3)$. Use the Strassen Matrix Multiply algorithm as a subroutine to give a faster algorithm for this problem, assuming the graph G is presented in adjacency matrix form.

All consecutive sums Give an efficient algorithm for the following problem: You are given an array $A[1..n]$ of real numbers, and want to compute the upper half of an $n \times n$ matrix $Sums[1..n, 1..n]$ where for each $1 \leq I \leq J \leq n$, $Sums[I][J] = \sum_{\{I \leq K \leq J\}} A[K]$. (Try to beat the obvious $O(n^3)$ algorithm.)

Implementation: 20 pts Implement the grade-school and clever divide-and-conquer multiplication algorithms, where the input is coded as an array of digits. Plot both performances on a log-log scale, for random integers of length n for n different powers of 2. Then combine them to use a threshold T as follows: IF $n < T$ use Gradeschool Multiply ELSE use the Divide-and-Conquer recurrence (but recursive calls are to the thresholded algorithm). What is the value of T that gives the best performance on random n digit numbers? Is it bigger or smaller than the cross-over point where divide-and-conquer beats gradeschool multiplication? Show the data to support your conclusions.