SPRING 2005: **COT 5993** Intro. to Algorithms

[Programming Assignment 1; Due Apr 19 at start of class]

Reminder: Add a signed statement that you have adhered to the collaboration policy for this class and that what you are presenting is your own work.

Problem Description

Your program should read in a list of n numbers (from a file `Data.txt`) and store it in an array A.

Next it should interactively prompt a user to type in a number x, and output YES or NO depending on whether or not there are two numbers in A that add up to x. The program should use two different algorithms to figure out the answer and also output the time it takes using both the methods.

The two algorithms are as follows:

Algorithm 1 (Naive) It should try the sum of every pair of numbers in A and check whether it adds up to x. Report YES as soon as a pair is found, else report NO. Also report the time taken by this algorithm.

Algorithm 2 (Smart) It should first sort A, and then for each number in A, it should perform binary search (as described in class) to check whether a pair of numbers in A adds up to x. Report YES as soon as a pair is found, else report NO. Also report the time taken by this algorithm.

Repeat the above process for values of n equal to 32, 64, 128, 256, 512, 1024, and 2048. For each value of n, test it on 10 different values of x. Then output the average times taken by algorithms 1 and 2 on the 10 runs for each value of n.

Notes

Which sorting algorithm should you use? Something to think about! Which one is known to be fastest in practice? If you want, you could use more than one to compare. You could try larger values of n (i.e., higher powers of two) and take the run times for the two algorithms and plot them on a graph. Why is it convenient to use powers of two for n? You could also plot the curves $n \lg n$ and n^2 for each value of n on the same plot. How will that help? Do the run times confirm what you know from your theoretical analysis?

You may use C++, C, Java, or Perl to do your task. Do not count the time it takes to read in the values into the array A. Do not count the time it takes to read in the value of x. If you want you can write two different programs for the two algorithms, although I suggest putting it in one program.

You are given one data file with 2048 numbers in it. For values of n smaller than 2048, simply read that many values and reuse the same file. Document your program well and print out the source code and the output for submission.