ECS 122B: Algorithm Design and Analysis Handout ??
UC Davis — Charles Martel Jan. 11, 2011

Problem Set 1—Due Friday, January 21 3:15

This problem set is intended to get you started on profiling, timing and using the machines. It also has some traditional algorithms type problems.

Primes: 50: There are various approaches to generating all primes up to some integer n. Two such approaches are given in the program p1.c in the profiling handout (the first tests all factors up to n, the second is smarter and only tests up to $\sqrt{n}$). You will explore these programs and try to improve them.

1) profile this program using gprof and gcov as described in the profiling handout for $n = 5,000$ and $20,000$.

Also time the entire program using the command "time a.out" for all both values of n and for $n = 50,000$. Do this for both optimized (compile using gcc -lm -O4 p1.c) and unoptimized versions.

2) Add timing commands to p1.c (as in the timing.c example) to time the individual prime routines. Time prime1 and prime2 for all three values of n.

3) The current program checks to see if the two functions give the same output and reports an error if not. Modify the program to use assert statements to do this as is described in column five.

4) Your goal now is to create a faster version of the program that given an integer n counts the number of primes in the range $1..n$ (modify p1.c to read in n as an input) Start by creating a program to do this based on prime2 and profile/time the new program for $n = 100,000$. Make sure your program works properly, but you need not cross check the results in the "production" version you time.

Using your profile describe what appears to be the main hot spots of the improved program. Then modify the program to significantly improve the running time (you should be able to reduce the running time to less than 1/4 of the prime2 approach if you are clever for $n = 100,000$).

Start by using code-tuning approaches (modify the code, but not the high level approach of checking divisors up to sqrt of n). You can then consider higher level changes which alter your basic approach.

5. Extra Credit: explore how the number of primes grows as a function of n.

What to turn in: (A) A copy of your profiles of the original program. for the new values of n.

(B) A description of the hot spots in your prime counting program, what changes you made, and a copy of your modified programs: your best "code-tuning one" and your best one overall (you should comment/highlight the changes you made).

(C) Paper and pencil problems:

i)[15] problem 5 from chapter 2 of Programming Pearls.

ii) [15] Consider the binary search problem when the input array x has duplicates. Describe how to modify the binary search code so you now return two positions i, j that represent the first and **last** occurrence of t in the array x ($i = j$ if there are zero or one items that match t). Your solution should still run in $(\log n)$ time.

iii)[20] In our solution to find the missing integer (where we didn't have enough memory to use a bit vector), a simple implementation repeatedly reads the input file, counting the number of values in the current range we know has a missing integer. Thus if our file has n integers in the range 0 to r, we need $\log r$ reads of the n integers and a total of $\Theta(n \log r)$ reads.

A smarter solution uses some auxiliary disk files to reduce the total amount of data read to $\Theta(n)$ (so we only reread a relevant part of the input data). You may assume there are no duplicates in the input, but note that since you don't have much memory, you must write things out to disk (sequentially) rather than collecting a lot of data before writing (for example, you might read in a block of 1K items from an input file, then collect up to a few K items before writing them out to disk).

Give a high level description of your algorithm and analyze its run time. Your goal is to solve the problem in $\Theta(n + \log r)$ time.

Please email the program parts to Sam. B,C can also be emailed, or can be turned in in class or to the HW box in Kemper Hall, room 2131. When you email him put:
ECS 122B PS1
in the subject line.