

Problem Set 1—Due Friday, January 19 4:00

This problem set is intended to get you started on profiling, timing and using the machines.

As we discussed in class 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 $12,000$.

Also time the entire program using the command `"time a.out"` for all three values of n . Do this for both optimized (`gcc -O4`) 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 better program for the following problem: given an integer n count the number of primes in the range $1..n$ (it is easy to modify `prime1` or `prime2` to do this).

Start by creating a program to do this based on `prime2` and profile/time this program for $n = 100,000$.

Using your profile describe what appears to be the main hot spots of the 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{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 highlight the changes you made).

(C) Paper and pencil problems:

i) 5,9 from chapter 2.

- ii) Problem 2 of chapter 4, but find the **last** occurrence of t in the array x .
- iii) Problem 5 from Chapter 5 (page 54)

Please email the program parts to me (martel@cs.ucdavis.edu). B,C can also be emailed, or can be turned in in class or to the HW box.