### 找质数项目报告

### 陈泰然 PB14210070

**基本算法描述**：由一个初始的素数表开始，逐步筛出后面的非素数，并将筛得的新素数加入原素数表，不断扩展素数表，最终即可得出目标素数。如初始的素数表是{2，3，5，7，11}，后续的待测数表为{13，15，17，19，21，23，25，27，29}（由于大于2的偶数不可能为素数，故提前将偶数忽略），则经一轮筛法运算之后，用3筛，待测表变成{13，17，19，23，25，29}，再一轮用5筛，变成{13，17，19，23，29}，由于7\*7=49>29，故可以不用7去筛了，待测数表已经全是素数，将其接在原素数表后，扩展成{2，3，5，7，11，13，15，17，19，23，29}，再用后面的数表来筛，素数表就会越来越大，最终达到需要求出的第n个素数。素数表的存储结构是一个数组：第一项为有效长度，以后直到下标为有效长度为止存储素数。如{5，3，5，7，11，13}，{6，3，5，7，11，13，17，0，0，0}.

**存储介质**：开始时老师给出的题目是求第十亿个素数，考虑到内存不够，我采用了文件读写的方式存储素数表。这样虽然运算速度被文件IO拖了下去，但是只要硬盘空间足够大，求第多少个素数都不是问题。后来老师要求降低了，求第一亿个素数，内存能装下了，于是我采用了直接在内存中存储素数表的方式，速度大大提升。现在我的程序只要在命令行中加一个参数“-f”来运行，就是工作于文件读写模式，什么参数都不加直接运行，就是工作于内存素数表模式。

**代码优化**：算法上，先将偶数排除在外，再用除2以外的素数进行筛法运算，相对于不将偶数排除在外的算法减少了一半的计算量；将前99个素数特殊处理，加快了小型运算的速度；将前面已经运算过的素数表保留，以备下一次运算需要，减少了重复运算，节约了不知多少运算成本。