Project Euler Problems

Michael E. Conlen

May 31, 2013

Preface

Project Euler, http://projecteuler.net/, is a list of programming problems with a mathematical and algorithmic bent. These problems have solutions that vary from the naïve to the sophisticated. While the easiest problems can be effectively solved naïvely the advanced problems require sophisticated solutions to run effectively. Here we compile a set of solutions in various programming languages along with a mathematical treatment of the sophisticated solutions. Where possible the solutions are generalized for various parameters given in the statement of the problem.

Contents

Listings

Chapter 1

Sum of Natural Numbers Divisible by 3 and 5

1.1 Introduction

Problem 1 of Project Euler asks the user to sum the integers less than 1000 which are multiples of 3 or 5. The naïve solutions is to iterate k over the range of integers and if $k \equiv 0 \pmod 3$ or $k \equiv 0 \pmod 5$ then add the integer to the sum. This solutions is given in Listing 1.1; however this solution runs in $O(n)$ time. A direct computation can be found.

Let n be the integer we iterate up through, in this case, 999*. Let $m_q = \left\lfloor \frac{n}{q} \right\rfloor$, the number of natural numbers less than n which are multiples of the natural number q; then notice that the sum of natural numbers less than n and divisible

* The problem asks for numbers up to 1000, thus does not include 1000 where it is a multiple of 5.

Listing 1.1: Problem 1: Naïve Solution

```
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char *argv[])
{
    int j,k;
    j=0;
    for(k=0; k<1000; k++) {
        if(k%3 == 0 || k%5 == 0) j+=k;
    }
    printf("%d\n", j);
    return(0);
}
```

by q is

$$\begin{aligned} q + 2q + 3q + \cdots + m_q q &= q \sum_{k=1}^{m_q} k \\ &= q\frac{(m_q)(m_q+1)}{2} \end{aligned} \tag{1.1}$$

If we are summing over the integers which are multiples of q and r then each natural number which is a multiple of both p and r is counted twice; thus we subtract multiples of qr; and the solution is

$$q\frac{(m_q)(m_q+1)}{2} + r\frac{(m_r)(m_r+1)}{2} - qr\frac{(m_{qr})(m_{qr}+1)}{2} \tag{1.2}$$

A generalized version of this program is given in Listing 1.2. It's runtime is $O(1)$.

Listing 1.2: Problem 1: C Solution

```c
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
    unsigned long long  q=0, r=0, n=0;
    unsigned long long  mq, mr, mqr, sum;
    char                copt;

    while((copt = getopt(argc, argv, "n:q:r:")) != -1) {
        switch(copt) {
            case    'n':
                n = atoll(optarg)-1;
                break;
            case    'q':
                q = atoll(optarg);
                break;
            case    'r':
                r = atoll(optarg);
                break;
            default:
                goto usage;
        }
    }
    if(n == 0 || q == 0 || r == 0) goto usage;

    mq = n/q;
    mr = n/r;
    mqr = n/(q*r);
    sum = q*(mq*(mq+1))/2 + r*(mr*(mr+1))/2 - (q*r)*(mqr*(mqr+1))/2;
    printf("%lld\n", sum);
    exit(0);
usage:
    fprintf(stderr, "%s -n N -q Q -r R\n", argv[0]);
    exit(-1);
}
```