PSSQ Algorithm
Expressing Primes 1 Mod 4
As a Sum of Squares

Jim Cullen, March 13, 2021, Ver 1.0

The problem of expressing a prime number as the sum of two perfect squares has a long history but it's a subject that is not often discussed. Several good algorithms have been published over the years by Hermite, Shanks, and others. Perhaps the better of these were published by Stan Wagon, with improvements due to the work of John Brillhart. The algorithm presented here is a slight variant of those but is closest to that of Brillhart.

Just a short introduction to the problem is given here. Any odd prime **p** must be congruent to **1** or **3 mod 4**. All perfect squares are congruent to **0** or **1 mod 4**. So, given the problem of

$$p = x^2 + y^2$$

where **p** is a given prime number congruent to **1 mod 4**, it's clear that one of the two squares must be congruent to **0 mod 4** and the other congruent to **1 mod 4**, in order to give the sum of **1 mod 4** that **p** requires. In other words, one must be even and the other odd. It has been proven that there is only one possible value for x^2 and one for y^2 that will satisfy the equality, when **p** is a prime congruent to **1 mod 4**. If a prime **p** is congruent to **3 mod 4** then there can be no solution since no combination of **0** and **1 mod 4** for the squares can sum to **3 mod 4** for the prime.

The speed of this algorithm is fairly quick, nearly instantaneous for integers up to 12 decimal digits. As an example, we'll look at one of the known primes used during testing, Cullen Number $\mathbf{C_{141}}$, a 45 digit prime. All Cullen Numbers are congruent to **1 mod 4** since one way of expressing the value of **Cn** is

$$C_n = 4 * [\ C_{n-1} - C_{n-2}\ ] + 1$$

When the value of $\mathbf{C_{141}}$ is entered into the **pssq(p)** function, it takes just under half a second to determine that

$$C_{141} = 393050634124102232869567034555427371542904833$$

$$= 18926813371255233171048^2 + 5901387102518894077727^2$$

and that half second *includes* the time required for a primality test on a 45 digit prime! The algorithm code presented here will only accept a prime number **p** that is congruent to **1 mod 4** as an input, returning **0** when an incompatible integer is entered. The output will be returned as **[x,y]**, such that $p = x^2 + y^2$, or **[0,0]** in the case of an algorithm failure (which has never been seen to happen). The function on the HP Prime is **pssq(p)**, where **PSSQ** stands for '**P**rime as a

Sum of **SQ**uares'. The symbol '**!=**' is used here to indicate '**not equal to**'. Comments are enclosed within **...** after each section of code to give explanation as required.

HP Prime code for pssq(p)

```
#cas
pssq(P):=
BEGIN
LOCAL H,C,SR,N,A,R;

IF isprime(P)!=1 OR irem(P,4)!=1 THEN
RETURN 0;
END;
```

** This section of code simply rejects any input **P** that is not a prime congruent to **1 mod 4**, returning **0** in that case. **

```
C:=1; H:=1; SR:=CEILING(√P);
WHILE H!=-1 DO
C:=C+1; H:=jacobi_symbol(C,P);
END;
```

** Using the Jacobi Symbol, a quadratic non-residue **(C/P)** is selected. This is usually a quick process since half of all integers tested will be a quadratic non-residues. The **SR** parameter, the square root of **P**, is used to determine when to end the descent method of residuals. **

```
N:=P; A:=powmod(C,(P-1)/4,P);
```

** We set the initial parameters for the descent portion of the algorithm. **

```
WHILE A>1 DO
R:=irem(N,A);
IF A<SR THEN RETURN [A,R]; END;
N:=A; A:=R;
END;
```

** The descent calculation. Normally the Extended Euclidian Algorithm would be used but a simple mod calculation will suffice with this method, as established by Stan Wagon. The While Loop continues until the value of **A** falls below that of **SR**, at which point the **[A,R]** pair is returned as the solution. The case of a residual being immediately equal to **1** is caught in the first iteration of the loop. An example would be when **P = 101 = 10^2 + 1^2**. If something goes terribly wrong with the descent loop, the value of **A** drops until it is equal to **1**, and the **[0,0]** result indicates this failure. This will not happen though if everything is coded correctly. **

```
RETURN [0,0];
END;
#end
```

** The PSSQ Algorithm terminates. **

When using this algorithm within some other programming project, you may wish the remove the initial checks on **p** for primality and for congruency to **1 mod 4**. This would give a significant speedup since no time would be wasted on a primality test. However, you must assure elsewhere in your code that **p** is a prime congruent to **1 mod 4** for proper function of the PSSQ algorithm.

Version Information

Version 1.0 was released Mar 13, 2021.

Jim Cullen
March 13, 2021