CATAM Part II - 15.1 - Primality Tests

Introduction

Throughout this project I code in python, using some inbuilt functions from the math package. The only built in functions of note used is gcd.

Question 1

The function *trial_division(n)* takes an integer n as input and returns a boolean True or False. We make a small optimisation by only testing 2 and odd divisors. The function *q1()* runs the tests on the given ranges, and returns the following:

Primes in range: $[1900, 1999]$ are $\{1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997, 1999\}$
Primes in range: $[1294268500, 1294268700]$ are $\{\}$

Question 2

First consider the problem of computing $a^b \mod N$. We do this by considering the binary representation of $b = \sum_{i=0}^{N} 2^i b_i$ where each b_i is 0 or 1. So then we may write a^b as $a^{b_0}(a^2)^{b_1} \dots (a^{2^N})^{b_N}$ and each individual power can be calculated iteratively $a^{2^k} = (a^{2^{k-1}})^2 \mod N$. We can then multiply the terms $(a^{2^k})^{b_k}$, taking the result (mod N) at each stage, hence the largest number ever stored is order N^2. This however is not good enough, so we must devise a cleverer way of multiplying mod N.

We can improve how we multiply numbers x and y. We write for y even, $xy = (2x)(\frac{y}{2})$ and for y odd $xy = x + (2x)(\frac{y-1}{2})$. In the y odd case, add x mod N to the result. Now iterate the process with $x' = 2x$ and $y' = \frac{y}{2}$ or $\frac{y-1}{2}$ accordingly. The process terminates when y becomes 0, which must occur as y becomes strictly smaller each iteration. In doing so we take mod N after every step and are just doing addition so worst case we store at most $2N << 10^{15}$. The algorithm is implemented as $modular\_multiply(x, y, mod)$.

The function $q1()$ produces the following results, including all values from both ranges.

```
2: {1901, 1905, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993,
   1997, 1999}

3: {1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997,
   1999}

4: {1901, 1905, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993,
   1997, 1999}

5: {1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997,
   1999}

6: {1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997,
   1999}

7: {1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997,
   1999}

8: {1901, 1905, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993,
   1997, 1999},

9: {1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997,
   1999}

10: {1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997,
   1999}

11: {1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997,
   1999}
```

```
12: {1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997,
    1999}

13: {1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 1993, 1997,
    1999}
```

These results are consistent with the question 1; we expect primes to pass the Fermat test to each base, hence be included in each of the above lists. The only pseudoprime in the above lists is 1905, to bases 2,4 and 8.

To analyse complexity, fix an a and N. To compute $a^{N-1} \mod N$, we use several steps:
1) We first write N-1 in binary. This can be done by iteratively subtracting from k the largest power of 2 smaller than k and storing the powers of 2 subtracted. This is $O(\log N)$.
2) We then compute all $a^{2^k} = (a^{2^{k-1}})^2$ using our multiplication algorithm. TODO Finish this off.

Question 3

The function $q3()$ performs the given task using fairly naive loops.

Firstly, we compute the so called Sarrus numbers, by applying the Fermat test to each integer, and checking for and ignoring primes using trial division. We get the following list of length 245:

```
{341, 561, 645, 1105, 1387, 1729, 1905, 2047, 2465, 2701, 2821, 3277, 4033,
    4369, 4371, 4681, 5461, 6601, 7957, 8321, 8481, 8911, 10261, 10585, 11305,
    12801, 13741, 13747, 13981, 14491, 15709, 15841, 16705, 18705, 18721, 19951,
    23001, 23377, 25761, 29341, 30121, 30889, 31417, 31609, 31621, 33153, 34945,
    35333, 39865, 41041, 41665, 42799, 46657, 49141, 49981, 52633, 55245, 57421,
    60701, 60787, 62745, 63973, 65077, 65281, 68101, 72885, 74665, 75361, 80581,
    83333, 83665, 85489, 87249, 88357, 88561, 90751, 91001, 93961, 101101,
    104653, 107185, 113201, 115921, 121465, 123251, 126217, 129889, 129921,
    130561, 137149, 149281, 150851, 154101, 157641, 158369, 162193, 162401,
    164737, 172081, 176149, 181901, 188057, 188461, 194221, 196021, 196093,
    204001, 206601, 208465, 212421, 215265, 215749, 219781, 220729, 223345,
    226801, 228241, 233017, 241001, 249841, 252601, 253241, 256999, 258511,
    264773, 266305, 271951, 272251, 275887, 276013, 278545, 280601, 282133,
    284581, 285541, 289941, 294271, 294409, 314821, 318361, 323713, 332949,
    334153, 340561, 341497, 348161, 357761, 367081, 387731, 390937, 396271,
    399001, 401401, 410041, 422659, 423793, 427233, 435671, 443719, 448921,
    449065, 451905, 452051, 458989, 464185, 476971, 481573, 486737, 488881,
    489997, 493697, 493885, 512461, 513629, 514447, 526593, 530881, 534061,
    552721, 556169, 563473, 574561, 574861, 580337, 582289, 587861, 588745,
    604117, 611701, 617093, 622909, 625921, 635401, 642001, 647089, 653333,
    656601, 657901, 658801, 665281, 665333, 665401, 670033, 672487, 679729,
    680627, 683761, 688213, 710533, 711361, 721801, 722201, 722261, 729061,
    738541, 741751, 742813, 743665, 745889, 748657, 757945, 769567, 769757,
    786961, 800605, 818201, 825265, 831405, 838201, 838861, 841681, 847261,
    852481, 852841, 873181, 875161, 877099, 898705, 915981, 916327, 934021,
    950797, 976873, 983401, 997633}
```

Secondly, we compute the Carmichael numbers, $n < 10^6$. It suffices to check which base 2 pseudoprimes fail for larger bases. We get the following list of 43 integers:

```
{ 561, 1105, 1729, 2465, 2821, 6601, 8911, 10585, 15841, 29341, 41041, 46657,
    52633, 62745, 63973, 75361, 101101, 115921, 126217, 162401, 172081, 188461,
    252601, 278545, 294409, 314821, 334153, 340561, 399001, 410041, 449065,
    488881, 512461, 530881, 552721, 656601, 658801, 670033, 748657, 825265,
    838201, 852841, 997633 }
```

We use the following simple observations to speed up our checking:
1) We can assume n is odd. To see this, suppose $n \geq 4$ is even. Then $(n-1)^{n-1} \equiv (-1)^{n-1} \equiv -1 \mod n$ so n fails the Fermat test base $n-1$.

2) We need only check bases up to $\lceil \frac{n}{2} \rceil$. This is because $(-a)^{n-1} \equiv (-1)^{n-1}a^{n-1} \equiv a^{n-1} \mod n$ since n is odd.

To determine how many bases we actually need to get this list, we'll start with the Sarrus numbers minus the Carmichael's, and determine the largest minimum base needed for all numbers in the list to fail the Fermat test. Running *q3ii*(), we find testing up to a=11 suffices, with 721801 the only Sarrus number needing this large a base.

Question 4

First we calculate the Jacobi Symbol. We'll use the following rules

$$(\frac{a}{N}) = (\frac{a \mod N}{N}) \tag{1}$$

$$(\frac{ab}{N}) = (\frac{a}{N})(\frac{b}{N}) \tag{2}$$

$$(\frac{a}{N}) = \begin{cases} 0 & \text{if } \gcd(a,N) \neq 1, \\ \pm 1 & \text{if } \gcd(a,N) = 1. \end{cases} \tag{3}$$

$$(\frac{-1}{N}) = (-1)^{\frac{N-1}{2}} = \begin{cases} 1 & \text{if } n \equiv 1 \pmod 4, \\ -1 & \text{if } n \equiv 3 \pmod 4, \end{cases} \tag{4}$$

$$(\frac{2}{N}) = (-1)^{\frac{N^2-1}{8}} = \begin{cases} 1 & \text{if } n \equiv 1,7 \pmod 8, \\ -1 & \text{if } n \equiv 3,5 \pmod 8. \end{cases} \tag{5}$$

and for M, N odd coprime integers

$$(\frac{M}{N})(\frac{N}{M}) = (-1)^{\frac{(N-1)(M-1)}{4}} = \begin{cases} 1 & \text{if } N \equiv 1 \pmod 4 \text{ or } M \equiv 1 \pmod 4, \\ -1 & \text{if } N \equiv M \equiv 3 \pmod 4 \end{cases} \tag{*}$$

The algorithm is as follows, identical to calculations performed on IIC Number Theory Example Sheets.

1) Reduce the numerator modulo the denominator using rule 1
2) Extract even numerators using rule 2 and 5
3) If the numerator is 1, we get 1. If the numerator and denominator are not coprime, rule 3 gives a result of 0.
4) Otherwise, have odd coprime integers, so flip using (*)

At each stage the denominator is strictly smaller, so the process must terminate. We implement this algorithm in *jacobi_calculate*(a, b) and also the Euler test in *euler_test*(a, N)

The following results are found using *q4*(). Firstly, we compute the Euler pseudoprimes base 2, identically to before, and find 114 of them:

```
{561, 1105, 1729, 1905, 2047, 2465, 3277, 4033, 4681, 6601, 8321, 8481, 10585,
    12801, 15841, 16705, 18705, 25761, 29341, 30121, 33153, 34945, 41041, 42799,
    46657, 49141, 52633, 62745, 65281, 74665, 75361, 80581, 85489, 87249, 88357,
    90751, 104653, 113201, 115921, 126217, 129921, 130561, 149281, 158369,
    162401, 164737, 172081, 188057, 196093, 208465, 215265, 220729, 223345,
    233017, 252601, 253241, 256999, 266305, 271951, 278545, 280601, 294409,
    314821, 323713, 334153, 340561, 348161, 357761, 390937, 399001, 410041,
    427233, 448921, 449065, 458989, 476971, 486737, 488881, 489997, 493697,
    514447, 526593, 530881, 552721, 580337, 588745, 625921, 635401, 647089,
    656601, 658801, 665281, 670033, 683761, 711361, 721801, 741751, 745889,
    748657, 800605, 818201, 825265, 838201, 838861, 841681, 852481, 852841,
    873181, 875161, 877099, 916327, 976873, 983401, 997633}
```

We find there are no absolute euler pseudoprimes in the range. In fact there are non at all, as shown in the IIC Number Theory Course.

We use a similar algorithm to question 3 to determine how many bases we need. Running *q4ii*() we find we need up to base 13, with 399001 the only number needing this high a base.

Question 5

We implement the test in *strong_test*(a, N). Using $q5()$ we find the 46 strong pseudoprimes of base 2 to be:

```
{2047, 3277, 4033, 4681, 8321, 15841, 29341, 42799, 49141, 52633, 65281, 74665,
    80581, 85489, 88357, 90751, 104653, 130561, 196093, 220729, 233017, 252601,
    253241, 256999, 271951, 280601, 314821, 357761, 390937, 458989, 476971,
    486737, 489997, 514447, 580337, 635401, 647089, 741751, 800605, 818201,
    838861, 873181, 877099, 916327, 976873, 983401}
```

and again there are no absolute strong pseudoprimes.

Via $q5ii()$ we see that only bases 2 and 3 are needed to determine the primality of all integers in the range.

Question 6

k	**Fermat**	**Euler**	**strong**	**primes**
5	28	13	3	8392
6	16	9	4	7216
7	6	4	0	6241
8	1	0	0	5411
9	0	0	0	4832

Table 1: $a = 2$

k	**Fermat**	**Euler**	**strong**	**primes**
5	28	17	7	8392
6	13	9	3	7216
7	5	4	0	6241
8	1	1	1	5411
9	0	0	0	4832

Table 2: $a = 3$

k	**fermat**	**euler**	**strong**	**primes**
5	9	3	0	8392
6	4	2	0	7216
7	2	2	0	6241
8	1	0	0	5411
9	0	0	0	4832

Table 3: $a = 2\&3$

We have the relationship, Strong $\Rightarrow$ Euler $\Rightarrow$ Fermat, where A $\Rightarrow$ B means if n passes A for some base a then it will also pass B for base a. This is confirmed by the decreasing numbers moving right across the tables.

The Euler $\Rightarrow$ Fermat implication is clear, since if $a^{(N-1)/2} \equiv \pm 1 \mod N$ then $a^{N-1} \equiv 1 \mod N$ by squaring, while the other implication is a little more involved.

Question 7

It turns out trial division is one of the better strategies to run on numbers this small. The reason being at worst, we end up only trial dividing numbers up to about 100000, and can avoid most of them if we are clever about it. This really isn't that many operations, and checking divisibility is comparatively easy. On the other hand, consider the tests available to us. The fermat test is fairly useless, since it can't tell us with any certainty if a number is prime. Both the euler and fermat tests are ok at detecting compositeness, but to check primality with them is hard (unless we've done some work before to limit how many bases we need to check). They also carry a heavier computational cost, since modular exponentiation, and calculating the jacobi symbol are non trivial.

bases	2	2,3	2,3,5	2,3,5,7
trial division	17.783	17.487	18.6	17.1
my test	19.099	20.785	24.7	25.8

Table 4: Seconds to run the two tests with varying bases for the strong test

It's worth noting since there are about $\frac{x}{\ln x}$ primes up to x, so most of our random sample will be composite. This suggests initially using trial division to be a good idea. We can be clever about how we do this, and picking small primes to test will eliminate the most composites of any choice of divisors. Lets fix this as primes up to 250 for now. After this, we employ a test. The strong test being the stronger of the two (as shown in q6) seems like a better choice, despite being potentially more computationally demanding. Since our tests our so demanding, we are only going to perform a small number of them, and revert to trial division after. We test various numbers of fixed bases used for the strong test stage over 100000 randomly generated integers in the range, and time the total time to determine the primality of them all.

So we don't improve on trial division at all, in fact we make it worse every time we add a strong test base. This is unsurprising in a way since we still are relying on trial division mostly, and simply using the strong test to reduce checking slightly. Thankfully, we can exploit some theory and come up with a much better test. C. Pomerance et al. [1] showed in 1980 that for $n < 25,000,000,000$ the only strong pseudo prime to all of bases 2,3,5 and 7 is 3,215,031,751 and so we can skip the trial division after. Doing so, we improve on our time massively. We also now test the impact of the small prime trial division at the start, and find checking primes up to 250 is quickest.

primes up to	0	50	100	150	200	250	300
running time	41.23	12.29	11.88	10.55	9.84	9.41	10.01

Table 5: Seconds to run my test with varying number of small primes trial divided at the start

The result in the article [1] is in a similar flavour to our thoughts before of how many bases would suffice to determine primality up to 10^6. Similar results exist for larger numbers with different/more bases, enabling this test to extend well to larger numbers. Trial division on the other hand becomes impossible when $n > 10^{20}$ or so, and our prime tests, while detecting compositeness quickly, will struggle to verify primality without the aid of these stronger results.

Question 8

Unsurprisingly, we find the strong test is very good. My implementation may be found under $q8()$. We loop over the whole range of values of N. If N prime, we record it as having passed all rounds of the strong test. Otherwise, we pick a random a coprime to N, and conduct the strong test. We repeat the strong test with more bases going until the composite no longer passes the strong test.

We run trials for k ranging from 15 through 22, and got no composites surviving more than 3 rounds of the strong test.

k/t	**1**	**2**	**3**	**Primes**
22	43	4	0	140336
21	31	2	0	73586
20	19	3	1	38645
19	22	2	1	20390
18	12	1	1	10749
17	15	1	0	5709
16	11	1	0	3030
15	9	0	0	1620

Table 6: Raw number of composites passing the strong for random bases, through t rounds

[1] C. Pomerance, J. L. Selfridge and Wagstaff, Jr., S. S., "The pseudoprimes to 25 · 109," Math. Comp., 35:151 (1980) 1003-1026. Page 1023 in particular

These probabilities agree but are all very small compared to theory, from which we get the following bound: A composite passes t rounds of the strong test to co-prime bases with probability at most $\frac{1}{4^k}$, from which we would expect the top right cell for example to be of order 10^5 taking into account how many composites are in the interval.