Kernel: SageMath 10.7

Counting Primes and the Riemann Zeta Function

MATH 242 Modern Computational Math

Today we will continue our study of the *prime counting function* $\pi(x)$, which is defined to be the number of primes less than or equal to x.

Remember our big question from last time: ***What is the shape of the prime counting function?*** In other words, can we approximate the prime counting function with simpler functions?

Here is our sieve of Eratosthenes function from a previous class session:

In [1]:
```
def sieveEratosthenes(nMax):
    # initialize a list
    # if you start the list from zero, then each number in the list is
EQUAL to its index in the list, which is very convenient
    nums = list(range(0, nMax + 1))

    # we will replace non-prime numbers in the list with zeros
    # start by replacing 1 with 0
    nums[1] = 0

    # loop over list items until we reach sqrt(nMax)
    i = 2
    while i <= sqrt(nMax):
        # if nums[i] is not zero, then it is prime
        if nums[i] > 0:
            # replace all multiples of nums[i] with zeros
            for j in range(2*i, nMax + 1, i):
                nums[j] = 0

        # testing: print the current state of nums
        #print(f"finished i={i}: nums={nums}\n")

        # increment i
        i += 1

    # return a list containing all nonzero elements of nums
    return [i for i in nums if i != 0]
```

Here is our function from Friday that returns a list of values of the prime counting function $\pi(x)$.

In [2]:
```
# returns a list of values of the prime-counting function π(x), for
integers x from 1 to nMax
def computePiVals(nMax):
    # compute a list of primes up to nMax
    primeList = sieveEratosthenes(nMax)
```

```
    # make a list of nMax+1 zeros
    piVals = [0]*(nMax+1)

    # track how many primes we've found so far
    count = 0

    # loop over integers i from 2 to nMax
    for i in range(2, nMax + 1):
        # if i is the next prime, then add 1 to our count
        if count < len(primeList) and i == primeList[count]:
            count += 1  # we found the next prime

        # store the current count in piVals[i]
        piVals[i] = count

    # return the list of piVals
    return piVals
```

Let's make a big list of $\pi(x)$ values for integers x from 0 up to some large M.

In [14]:

```
nMax = 200000
piVals = computePiVals(nMax)
piVals[:20]
```

Out[14]:

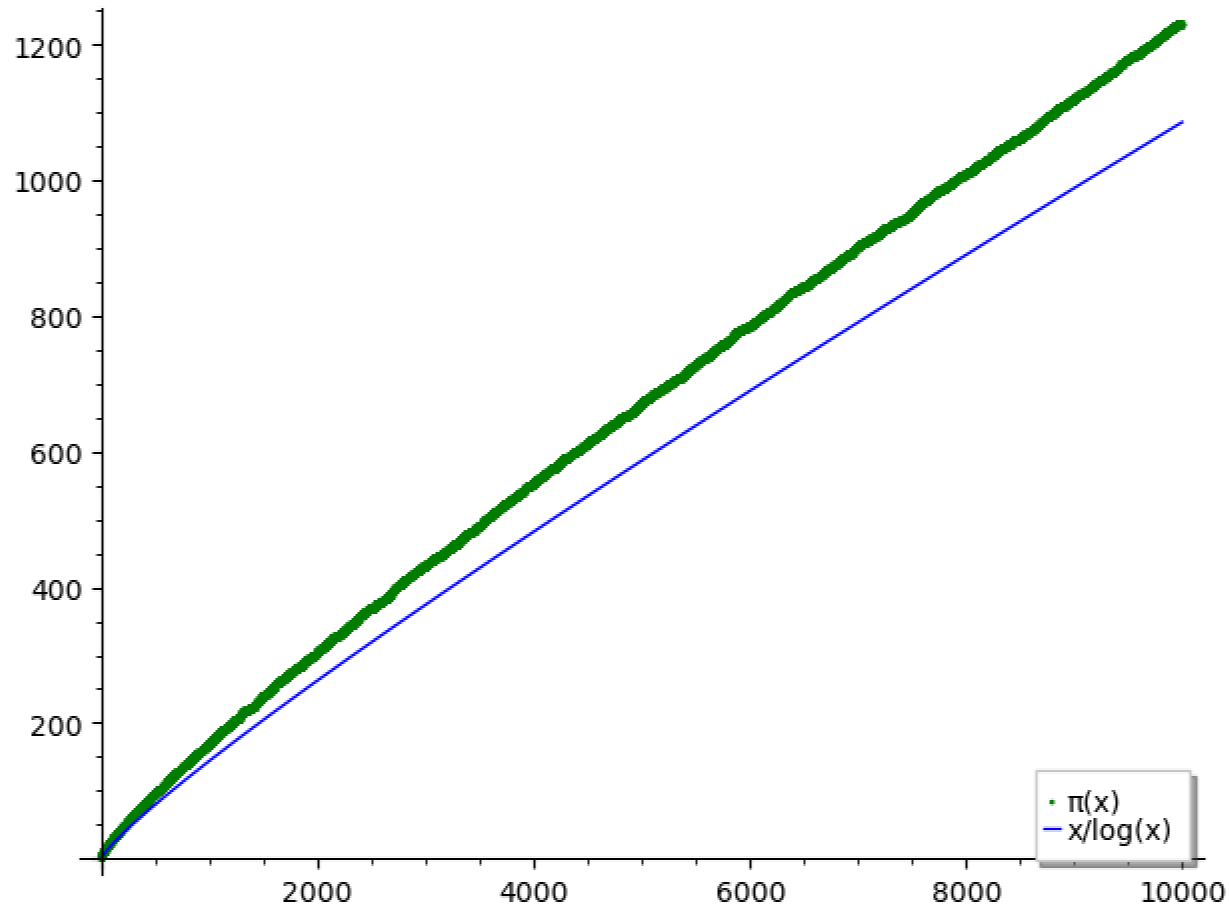
```
[0, 0, 1, 2, 2, 3, 3, 4, 4, 4, 4, 5, 5, 6, 6, 6, 6, 7, 7, 8]
```

In the homework for today, we considered $\frac{x}{\ln(x)}$ as a possible approximation of $\pi(x)$. Make a plot comparing $\frac{x}{\ln(x)}$ and $\pi(x)$.

In [6]:

```
xMin = 2
xMax = 10000
combined = list_plot(piVals[xMin:xMax], color="green",
legend_label='π(x)') + plot(x/log(x),(x,xMin,xMax),
legend_label="x/log(x)")
combined.show(legend_loc="lower right")
```

Out[6]:



Density of Primes

Roughly speaking, the *density* of primes near x is the proportion of integers near x that are prime. We can approximate the density of primes near x by fixing some length ℓ and computing the proportion of integers in the interval $(x, x+\ell)$ that are prime. In other words, the density of primes near x is approximately

$$\frac{\pi(x+\ell)-\pi(x)}{\ell}$$

Complete the following function that approximates the density of primes near x:

In [8]:

```
def primeDensity(x, length):
    return (piVals[x+length] - piVals[x])/length
```

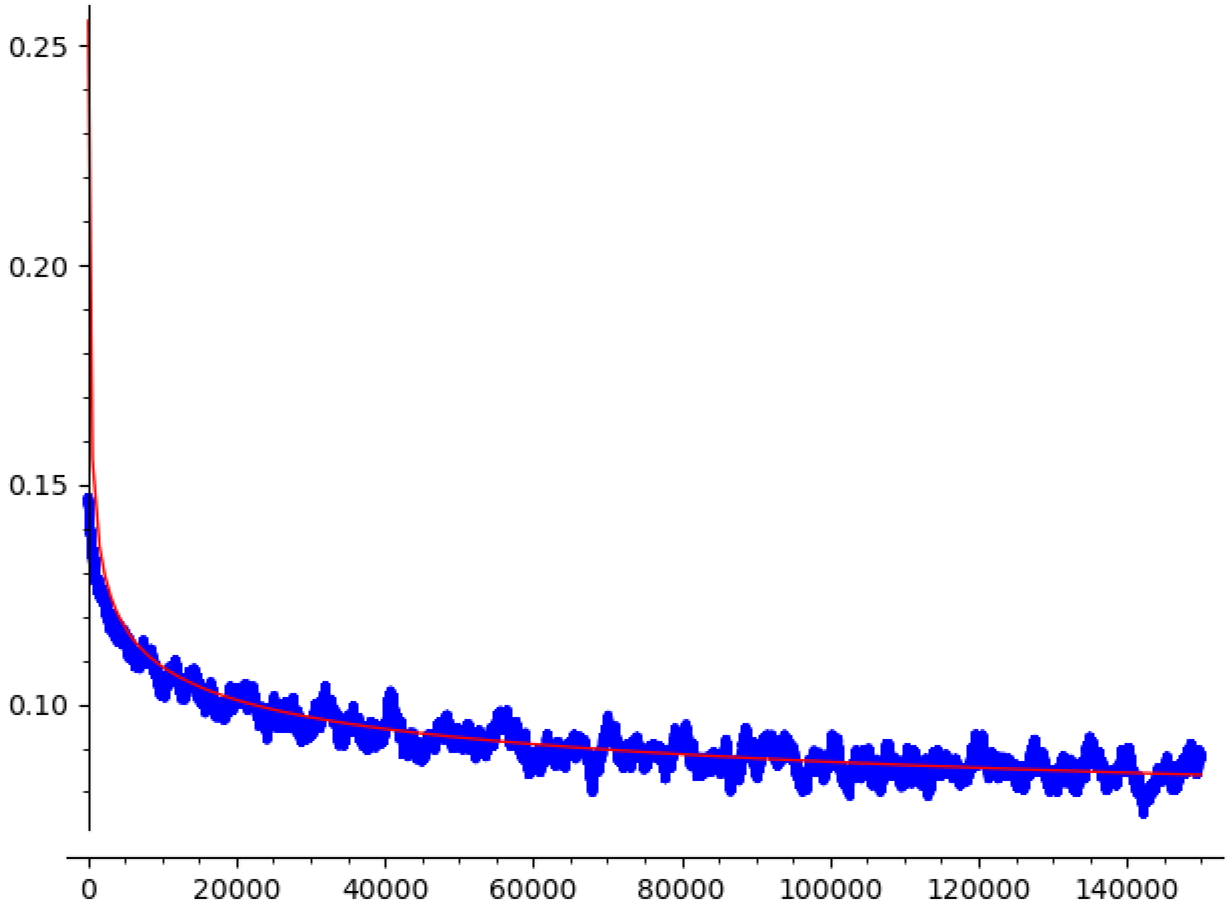
Make some plots of your prime density function for different values of `x` and `length`.

In [19]:

```
xVals = range(50,150001)
length = 2000
densityVals = [primeDensity(x,length) for x in xVals]

list_plot(list(zip(xVals,densityVals))) + plot( 1/log(x), (x, 50,
150000), color="red")
```

Out[19]:



In [0]:

In the late eighteenth century, using printed tables of prime numbers, Gauss conjectured that the density of primes near x is approximately $\frac{1}{\ln(x)}$. Can you provide computational evidence for or against Gauss's conjecture?

In [0]:

In [0]:

The Logarithmic Integral

If the density of primes near x is approximately $\frac{1}{\ln(x)}$, then the *count* of primes up to x should be approximately the integral $\int_0^x \frac{1}{\ln(t)} dt$. This integral has a special name and notation:

Definition: The *logarithmic integral*, $\mathrm{li}(x)$ is defined

$$\mathrm{li}(x) = \int_0^x \frac{1}{\ln(t)} dt.$$

This integral is a bit tricky to compute, but fortunately it is already implemented in Sage as `li()` and also as `log_integral()` .

In [0]:

Compute some values of $\mathrm{li}(x)$. How do they compare to $\pi(x)$ and $\frac{x}{\ln(x)}$?

In [0]:

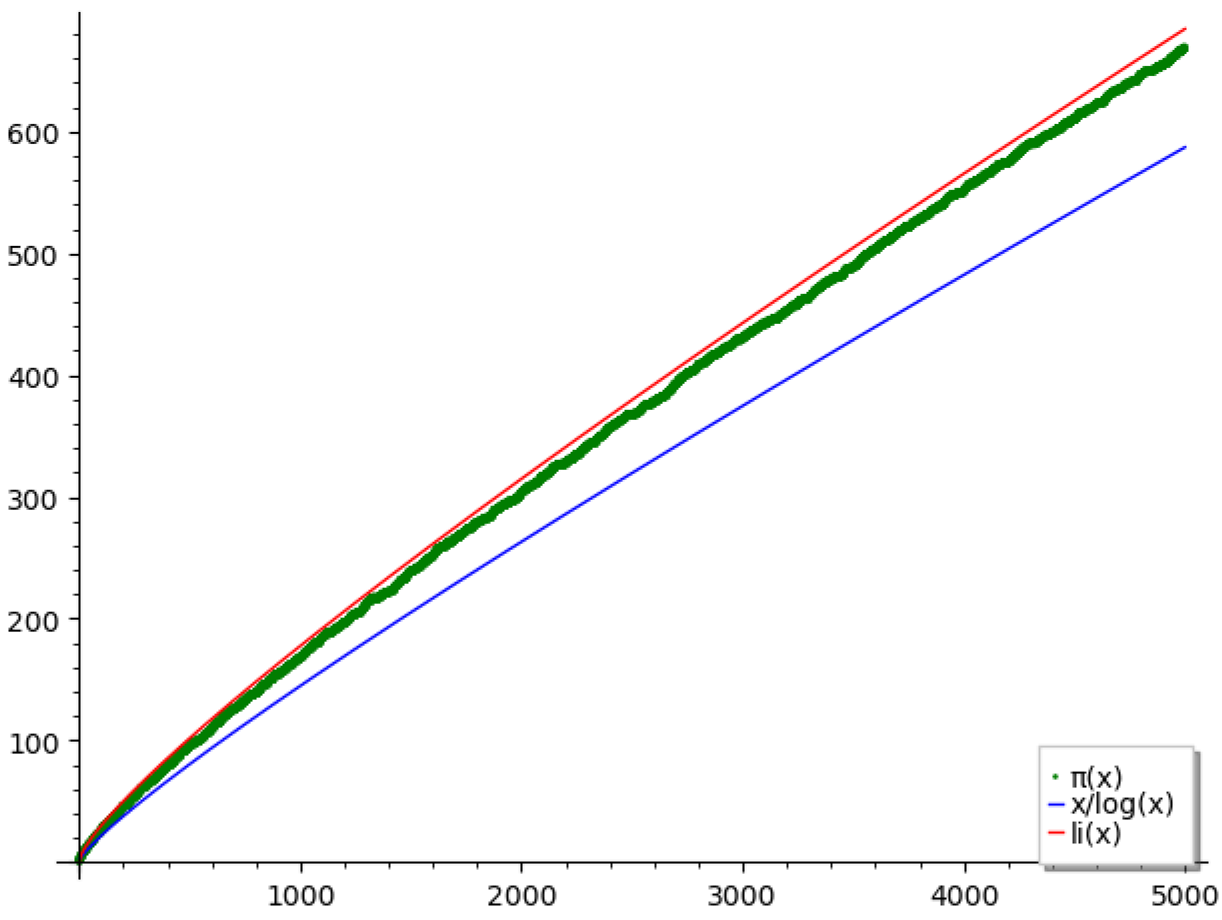
In [0]:

Make a plot showing the values of $\pi(x)$, $\frac{x}{\ln(x)}$, and $\mathrm{li}(x)$. What do you observe?

In [23]:

```
xMin = 2
xMax = 5000
combined = list_plot(piVals[xMin:xMax], color="green",
legend_label='π(x)') + plot(x/log(x),(x,xMin,xMax),
legend_label="x/log(x)") + plot( li(x), (x, xMin, xMax),
legend_label='li(x)', color='red' )
combined.show(legend_loc="lower right")
```

Out[23]:



In [0]:

The Riemann Zeta Function

To find an even better approximation to the prime counting function, we must learn about a function called the *Riemann zeta function*. This function leads to one of the most important open questions in mathematics, the *Riemann Hypothsis*, which is a statement about the zeros of the Riemann zeta function.

The Riemann zeta function, $\zeta(s)$, is defined

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \frac{1}{1^s} + \frac{1}{2^s} + \frac{1}{3^s} + \frac{1}{4^s} + \frac{1}{5^s} + \cdots$$

The Riemann zeta function converges for all $s > 1$. Moreover, it converges for all *complex* numbers s with real part greater than 1. However, for $s = 1$ the sum diverges, so $\zeta(1)$ is undefined.

Write a Python function below that approximates $\zeta(s)$ by computing a partial sum of `numterms` terms.

In [24]:
```
def zeta(s, numTerms):
  return sum( [1/(n^s) for n in range(1, numTerms+1)] )
```

Use your `zeta` function to compute decimal approximations of $\zeta(2), \zeta(3), \zeta(4), \ldots$. What do you notice? Can you find closed-form expressions for any of these values?

In [29]:
```
zeta(2, 10000).n()
```

Out[29]: 1.64483407184806

In [28]:
```
n(pi^2/6)
```

Out[28]: 1.64493406684823

In [0]:
```

```

One connection between the Riemann zeta function and the prime numbers has to do with the following product:

$$\prod_{p \text{ prime}} \frac{1}{1 - p^{-s}}$$

This is a product over all prime numbers p.

Write a Python function below that computes partial products of this infinite product. Take the primes in order, starting with the smallest prime.

In [0]:
```
def primeProduct(s, numFactors):

  # YOUR CODE HERE
```

In [0]:

```

```

In [0]:

```

```

What is the connection between $\zeta(s)$ and $\prod_{p \text{ prime}} \frac{1}{1-p^{-s}}$?