```
import timeit
# Bir sayının asal olup olmadığını kontrol eden fonksiyon
def is_prime(n):
    if n <= 1:
        return False
    if n <= 3:
        return True

    if n % 2 == 0 or n % 3 == 0:
        return False

    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6

    return True

# Belirli bir üst sınıra kadar olan asal sayıları bulan fonksiyon
def find_primes(limit):
    primes = []
    for num in range(2, limit + 1):
        if is_prime(num):
            primes.append(num)
    return primes

# Kullanıcıdan bir üst sınırı alalım
n = int(input("Asal sayıları bulmak için bir üst sınır girin: "))

# Sezgisel yöntemi kullanarak asal sayıları bulalım
prime_numbers = find_primes(n)

# Sonuçları ekrana yazdıralım
print("2'den", n, "kadar olan asal sayılar:")
print(prime_numbers)

# Performans ölçümü için kullanılacak n değeri
n = 1000

# Hız ve performans ölçümü için timeit kullanalım
execution_time = timeit.timeit("find_primes(n)", globals=globals(), number=1000)

# Ölçülen süreyi ekrana yazdıralım
print(f"Sezgisel yöntemle {n} kadar olan asal sayıları bulma süresi: {execution_time:.6f} saniye")
```

```
Asal sayıları bulmak için bir üst sınır girin: 20
2'den 20 kadar olan asal sayılar:
[2, 3, 5, 7, 11, 13, 17, 19]
Sezgisel yöntemle 1000 kadar olan asal sayıları bulma süresi: 0.676543 saniye
```

Algoritma Yaklaşımı:

Eratosthenes Eleği: Bu algoritma, tüm sayıları içeren bir liste oluşturur ve ardından asal sayıları elemek için bir döngü kullanır.

Sezgisel Yöntem: Bu yöntem, her sayının asal olup olmadığını kontrol ederek asal sayıları bulur. Sayının asal olduğunu belirlemek için bölenlerini arar.

Zaman Karmaşıklığı:

Eratosthenes Eleği, n sayısı için yaklaşık olarak O(n * log(log(n))) zaman karmaşıklığına sahiptir. Büyük n değerleri için daha hızlıdır. Sezgisel Yöntem, n sayısı için yaklaşık olarak O(n * sqrt(n)) zaman karmaşıklığına sahiptir. Büyük n değerleri için daha yavaştır. Bellek Kullanımı:

Eratosthenes Eleği, n sayısı için O(n) bellek kullanır. Bu, büyük n değerleri için daha fazla bellek gerektirir.

Sezgisel Yöntem, sabit bellek kullanır (O(1)). Bu, bellek kullanımı açısından daha verimlidir.