```
import bisect, math, time

primes = [2]

def next_prime(n):
    while any([n % d == 0 for d in primes]): n += 1
    return n

def add_primes(n):
    r = math.sqrt(n)
    while primes[-1] < r: primes.append(next_prime(primes[-1] + 1))

def pi(n):
    # Knuth's algorithm G (Gray binary generation) is used to generate all
subsets of range(m)
    m = bisect.bisect_right(primes, math.sqrt(n))
    a = [0 for _ in range(m)]
    s, μ = m, 1
    while (True):
        d = math.prod([primes[i] for i in range(m) if a[i]])
        s, μ = s + μ * math.floor(n/d), -μ
        j = 0 if μ == -1 else a.index(1) + 1
        if j == m: break
        a[j] = 1 - a[j]
    return s

add_primes(10000)
for k in range(1, 11):
    n = 1000*k
    t0 = time.time(); p = pi(n); t1 = time.time()
    print('n = {:5d}, pi = {:4d}, time = {:10.6f} s'.format(n, p, t1-t0))
```