Primes Project

Math 242

due Monday, March 22

This project has two parts, both involving prime numbers.

Part 1: Implement the Sieve of Sundaram and compare it with the Sieve of Eratosthenes. (You may use either your implementation of the Sieve of Eratosthenes or an implementation from class.) Specifically, compare the runtime required for the sieves to compute lists of primes. You should measure the runtime of each sieve for computing the primes up to n, for various n. Then make a plot that shows your results (n on the horizontal axis, runtime on the vertical axis).

Part 2: Investigate at least two of the following conjectures. For each, either provide computational evidence in support of the conjecture, or find a counterexample showing that the conjecture is false.

- **Conjecture A.** Every even integer greater than 2 is the sum of two primes.
- **Conjecture B.** For every N, the number of nonnegative integers less than N with an *even* number of prime factors is less than the number of nonnegative integers less than N with an *odd* number of prime factors. For this, prime factors are counted *with multiplicity*; e.g., $24 = 2^3 \cdot 3$ has 4 prime factors, while $588 = 2^2 \cdot 3 \cdot 7^2$ has 5 prime factors.
- **Conjecture C.** For every positive integer n, there exists at least one prime between n^2 and $(n+1)^2$.
- **Conjecture D.** All odd numbers greater than 1 are either prime, or can be expressed as the sum of a prime and twice a square.

For Part 2, you may use any built-in functions you like, such as Prime, PrimeQ, and FactorInteger. As you investigate these conjectures, try to push the limits of the computational power available to you. For example, don't just test 100 cases and declare that the conjecture must be true. Can you test one million cases? One billion? Of course, you should stop if you find a counterexample, since a single counterexample shows that the conjecture is false.

As usual, your Mathematica notebook should indicate not only what you computed, but also how well you understand what you did. A list of calculations with no reasoning will not suffice. Your goal should be to communicate your work to another person (e.g., another student at your level who is not in this course).

Only submit code that actually runs. If you can't get something complicated to work, try something simpler. It's better to turn in an incomplete assignment that runs instead of a "complete" assignment that doesn't run.

Your notebook will be graded on a scale of 0 to 16 points. The following rubric gives characteristics of notebooks that will merit sample point totals. (Interpolate the following for point totals that are not divisible by 4.)

16 points. Problems and goals are clearly stated, including relevant definitions or parameters. Computations are complete; code runs and is clearly explained. Conclusions are clearly stated and backed up by sufficient computational evidence. Limitations of the methodology, extensions for future work, and conjectures are discussed. Notebook is well-formatted and easy to read.

12 points. Problems and goals are stated well, though relevant definitions or parameters may be missing. Computations are mostly complete; code runs, but explanation is weak. Conclusions are unclear or not well justified. Insufficient discussion of limitations, extensions, and conjectures.

8 points. Statement of problem or goal is unclear. Computations are incomplete; explanation is ambiguous. Code may produce errors when run. Conclusions are possibly correct, but not justified. Little or no discussion of limitations, extensions, or conjectures. Notebook is difficult to read.

4 points. Serious misunderstanding of the problem or goal. Computation is inadequate for the task at hand. Work is not clearly explained. No discussion of limitations, extensions, or conjectures. Notebook is difficult to read.

0 points. Notebook is not turned in.