The Sieve of Eratosthenes

Dr. Mattox Beckman

University of Illinois at Urbana-Champaign
Department of Computer Science

Objectives

Your Objectives:

- Be able to implement the Sieve of Eratosthenes
- Enumerate some applications of prime numbers

Calculating Primes the Hard Way

- You need to see if a number is prime / factorize a number. How can you do that?
  - Trial division...

```
pIsPrime = true;
for(i=2; i<p; ++i)
   if (p % i == 0) {
     pIsPrime = false;
     break;
   }
```

A slight improvement

- Improvement 1: only check the odd numbers

```
0 pIsPrime = true;
1 if (p % 2 == 0)
2   pIsPrime = false;
3 else
4   for(i=3; i<p; i+=2)
5       if (p % i == 0) {
6         pIsPrime = false;
7         break;
8       }
```

Improvement 2 – Stop at $\sqrt{p}$

- We can stop at $\sqrt{p}$.
- If $q > \sqrt{p}$ and $q|p$, then there is a factor $k < \sqrt{p}$ such that $kq = p$.

```
0 #include <cmath>
1
2 int sqrtP = std::sqrt(p)
3 pIsPrime = true;
4 if (p % 2 == 0)
5   pIsPrime = false;
6 else
7   for(i=3; i<sqrtP; i+=2)
8     if (p % i == 0) {
9       pIsPrime = false;
10      break;
11    }
```

The Sieve

```
0 // From Competitive Programming 3
1 #include <bitset>
2 ll _sieve_size; // 10^7 should be enough for most cases
3 bitset<10000010> bs;
4 vi primes;
5
6 void sieve(ll upperbound) {
7   _sieve_size = upperbound + 1;
8   bs.set();  // all bits set to 1
9   bs[0] = bs[1] = 0;
10  for (ll i = 2; i <= _sieve_size; i++)
11    if (bs[i]) { // cross out multiples of i starting from i * i!
12      for (ll j = i * i; j <= _sieve_size; j += i) bs[j] = 0;
13      primes.push_back((int)i);
14 } }
```

Factoring

```
0 // From Competitive Programming 3
1 vi primeFactors(ll N) {
2    vi factors;
3    ll PF_idx = 0, PF = primes[PF_idx]; // primes has been populated by si
4    while (PF * PF <= N) {
5       while (N % PF == 0) {
6           N /= PF; factors.push_back(PF); }
7       PF = primes[++PF_idx];
8    }
9    // special case if N is a prime
10   if (N != 1) factors.push_back(N);
11   return factors;
12 }
```