Computational Problem Solving

- Three pillars of science and engineering:
  - Theory
  - Experimentation
  - Computation (Simulation)
- Some problems are difficult to analyze analytically, but easy to simulate.
- Learn to “think computationally” to get results from simple simulations.
- Use computation/simulation to explore.

Computational Example 1

- Birthday problem: Among a group of *n* people, what is the probability that two share a birthday?
  - This is related to hashing.
  - Can you determine this analytically?
  - How can you do this with simulation?

Algorithm #1

```
bool birthday(int count) {
  int myArray[365];
  for (int i=0; i<count; i++) {
    int pos = Random(365);
    if (myArray[pos] != 0)
      return true;
    else myArray[pos] = 1;
  }
  return false;
}
```

Issue: Must do it enough times to get meaningful statistics

Algorithm #2

```
double birthday(int count, int numtrials) {
  int myArray[365];
  int hits = 0;
  for (int trial=0; trial<numtrials; trial++) {
    for (int i=0; i<365; i++) myArray[i] = 0;
    for (int i=0; i<count; i++) {
      int pos = Random(365);
      if (myArray[pos] != 0)
        { hits++; break; }
      else myArray[pos] = 1;
    }
  }
    return (double)hits/(double)numtrials;
}
```

Computational Problem 2

- Analysis of hashing: What should we expect from a good hash function in terms of number of slots hit, length of chains?
  - Possible to analyze “ideal” performance analytically, but harder than simulating
  - Very hard or impossible to analyze performance of real hash functions analytically, but easy with simulation.

Things to Know

- Performance Measures:
  - How many slots were used (average)?
  - What is the minimum for slots used?
  - What is the longest chain ever?
  - What is the average for longest chain?
  - What is the expected cost?
- Issues:
  - Data Distribution
  - Fill factor
  - Table size

Computational Example 3

- Do you know an algorithm to compute a square root?
- Assuming that you know how to multiply, can you think of a way to compute square roots?
- Guess/convergence testing is a fundamental concept for many numerical methods.

Algorithm

```
double squareRoot(double val) {
  double lower, upper;
  upper = val;
  if (val < 1) lower = 0;
  else lower = 1;
  while ((upper – lower) > EPSILON) {
    double curr = (upper + lower)/2.0;
    if ((curr * curr) > val) upper = curr;
    else lower = curr;
  }
}
```