CS 226 Lecture 1: Introduction

R. Sedgewick
rs@cs.princeton.edu

Intermediate-level survey course
- prerequisite: COS 126
- programming/problem solving

Algorithm: method for solving a problem
Data structure: a way to store information

Efficient algorithms use good data structures

Types of algorithms considered
- Sorting
- Searching
- Strings
- Geometric
- Graph
- Mathematical
- Other

Why study algorithms?

Using a computer?
- want it to go faster
- want it to process more data
- want to do something that would
  otherwise be impossible

Technology improves things by a constant factor
Good algorithm design can do better
- Abacus
- Slide Rule
- Calculator
- Computer
- Supercomputer

A bad algorithm on supercomputer
can be worse than a good algorithm on abacus

Algorithms as a field of study
- old enough that basics are known
- new enough that new discoveries arise
- burgeoning application areas
  - science
  - engineering
  - commercial
- philosophical implications

  .
  .
  .

Analysis of algorithms

Compare algorithms by comparing estimated costs
Refine and verify estimates for the best ones

N: size of the input

Running time usually proportional to

- 1
- log N
- N
- N log N
- N^2
- 2^N

Often can write down a formula for running time

- * worst case (guarantee)
- * average case (prediction)

Analysis depends on detailed information

- * costs of basic operations
- * properties of input

Supress details in approximate analysis

"Within a constant factor" O-notation
Predict time for 2N from time for N

Example: N-body simulation

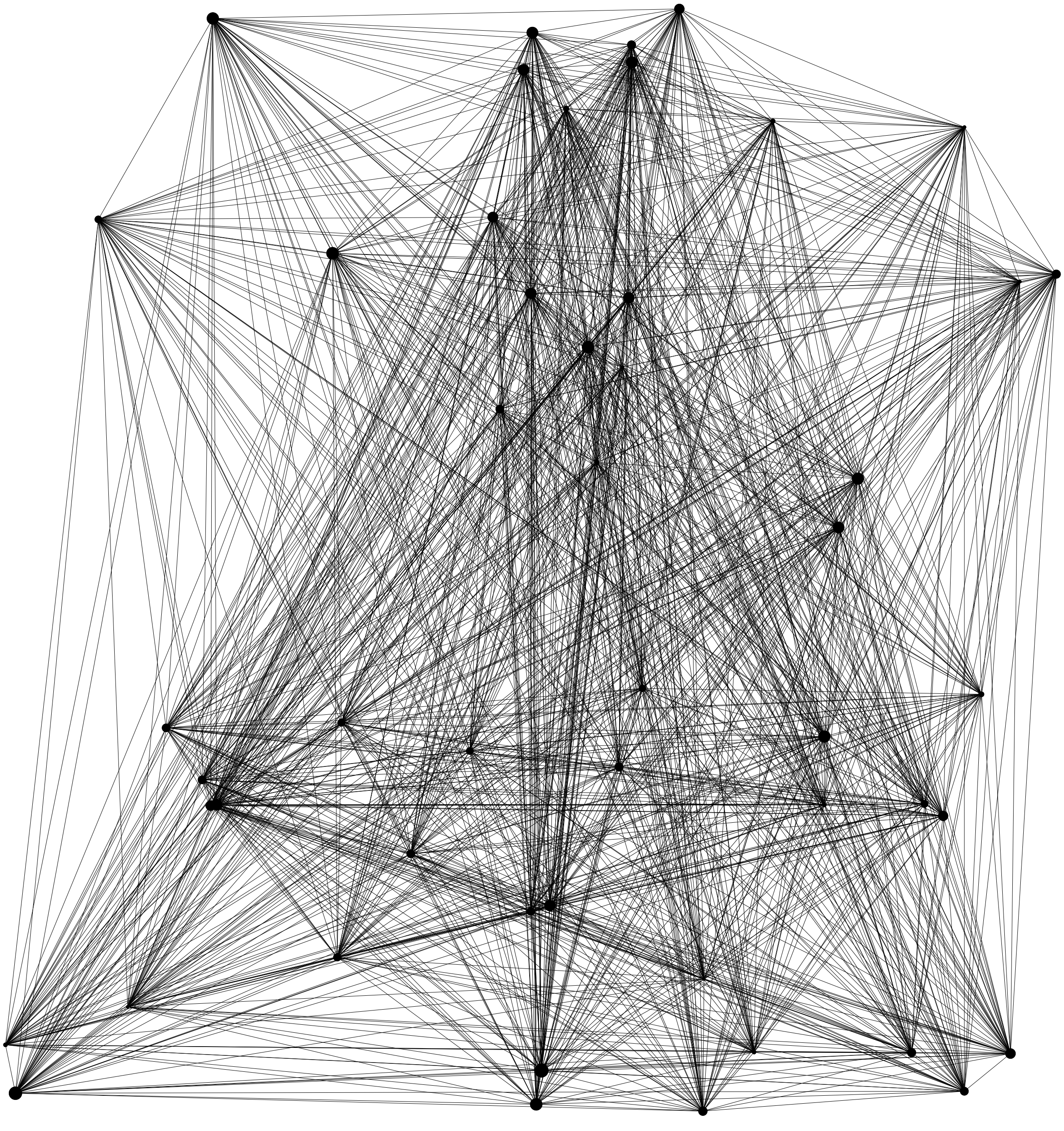
N bodies mutually affected by gravitational force

Problem: compute position after M time steps

N-body simulation (continued)

50 points

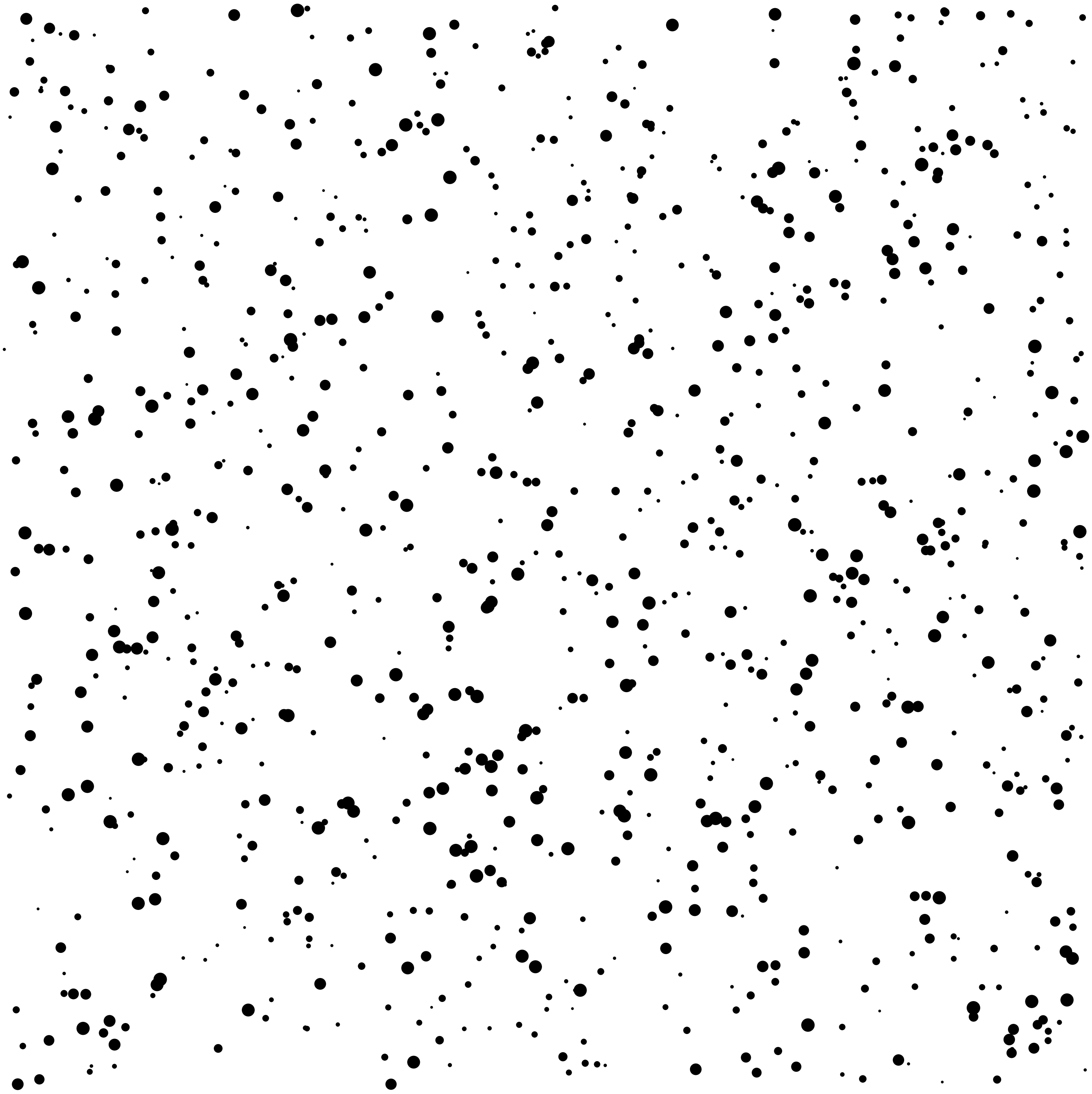
1250 pairs of points



N-body simulation (continued)

1000 points

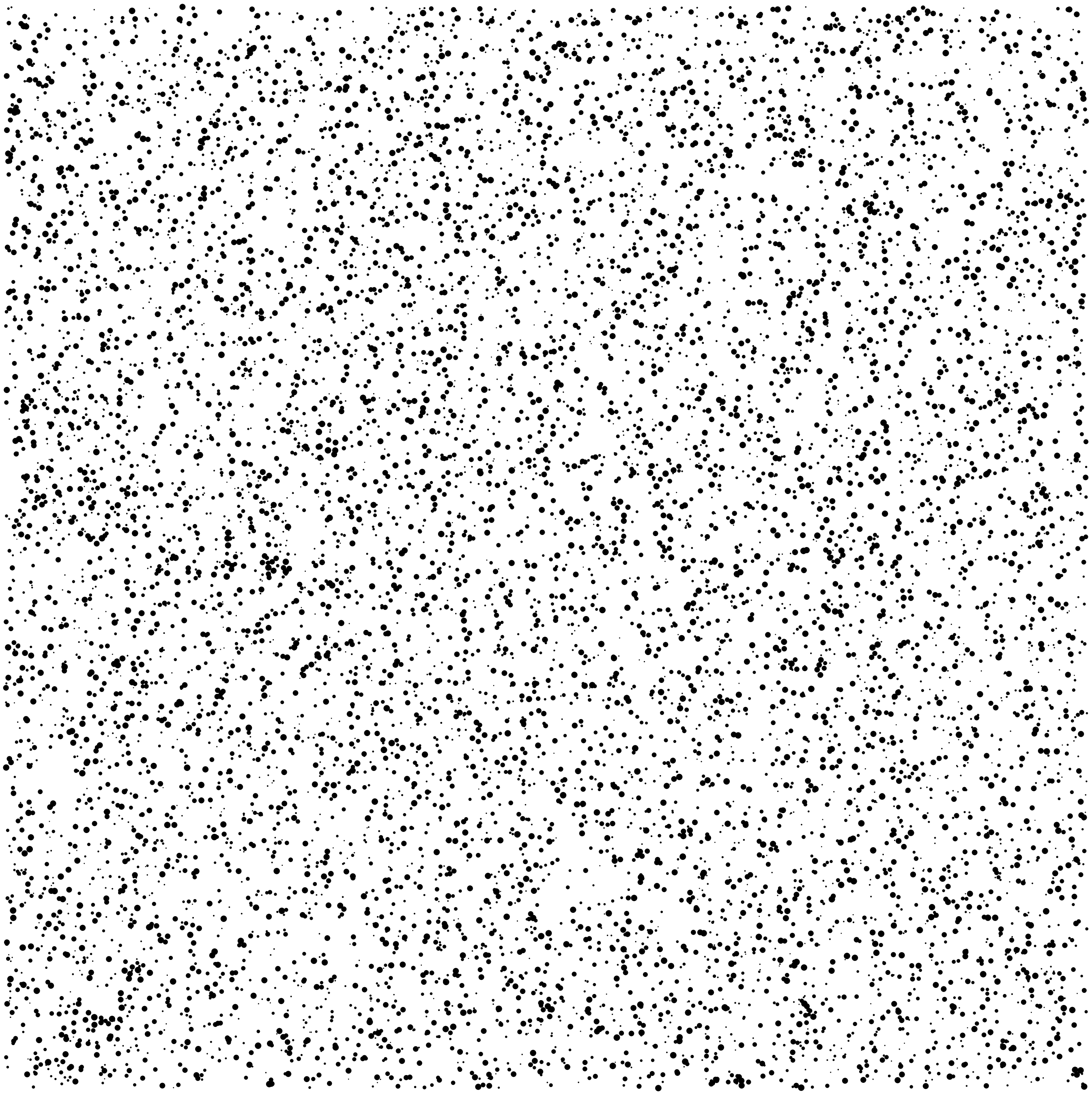
1/2 million pairs of points (edges not shown)



N-body simulation (continued)

10000 points

50 million pairs of points (edges not shown)



N-body simulation (continued)

Brute-force algorithm (from physics)

mass	mass[]
force	fx[], fy[], fz[]
acceleration	ax[], ay[], az[]
velocity	vx[], vy[], vz[]
position	x[], y[], z[]

```
for (T = 0; T < M; T++)
  { << initialize fx[], fy[], fz[] to 0.0 >>
     for (i = 0; i < N; i++)
        for (j = i+1; j < N; j++)
           { distsq = (x[i]-x[j])^2
                         +(y[i]-y[j])^2
                            +(z[i]-z[j])^2;
             f = mass[i]*mass[j]*G/distsq;
             << update fx[i], fy[i], fz[i] >>
             << update fx[j], fy[j], fz[j] >> }
     for (i = 0; i < N; i++)
        { << update ax[i], ay[i], az[i] >>
           << update vx[i], vy[i], vz[i] >>
           << update  x[i],  y[i],  z[i] >> }
  }
```

Takes time proportional to M N^2

N-body simulation (continued)

Estimate of running time

    10000 points
    1000 time steps

Rough estimates

    20 floating point ops in inner loop
    5x10^7 iterations per time step
    = 10^12 (one trillion) floating point ops

        1000000000000

A large number, by any standard!

For useful results, need to do

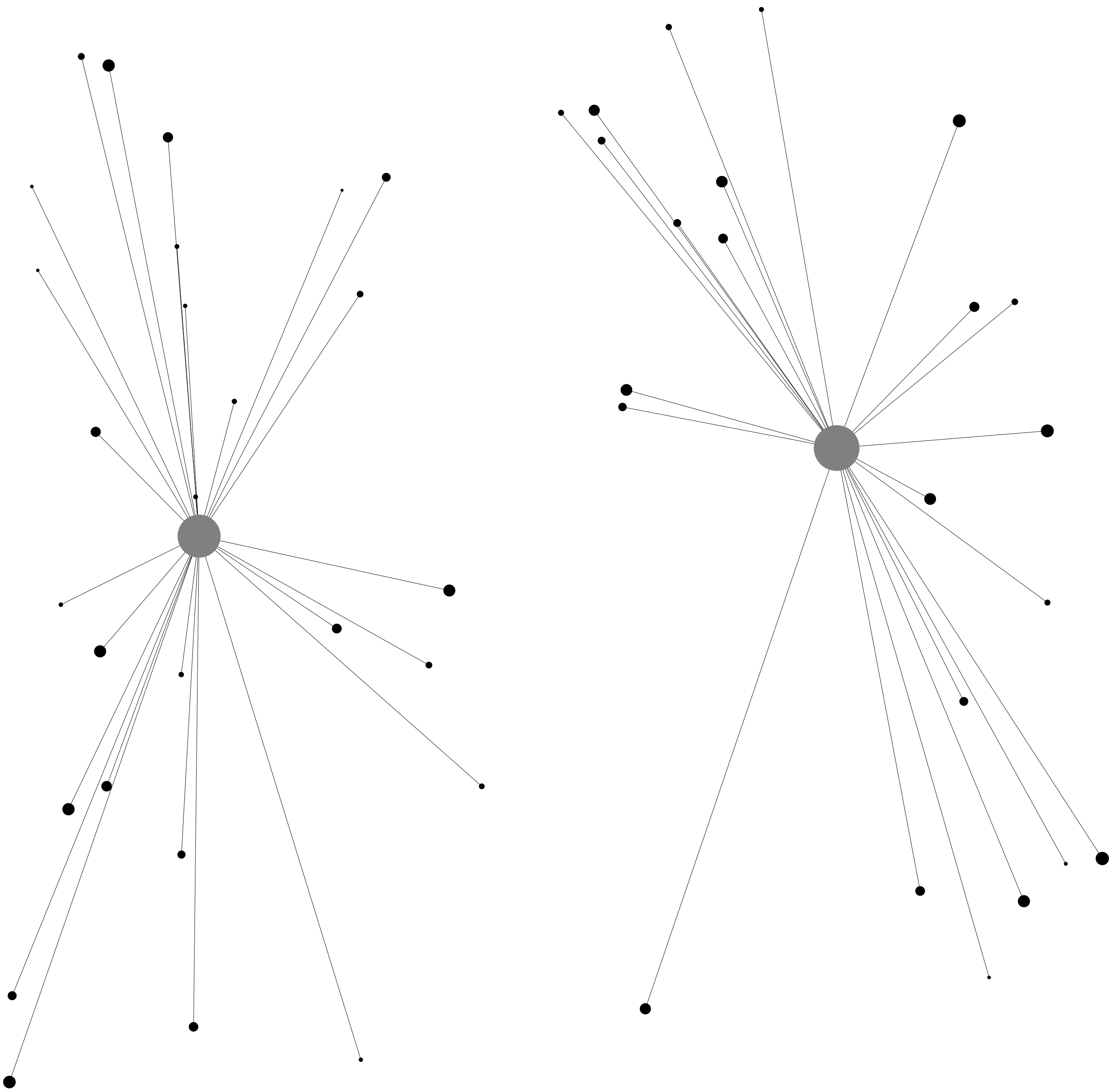
    orders of magnitude more points
    orders of magnitude more time steps

Good algorithm required for scientific progress

N-body simulation (continued)

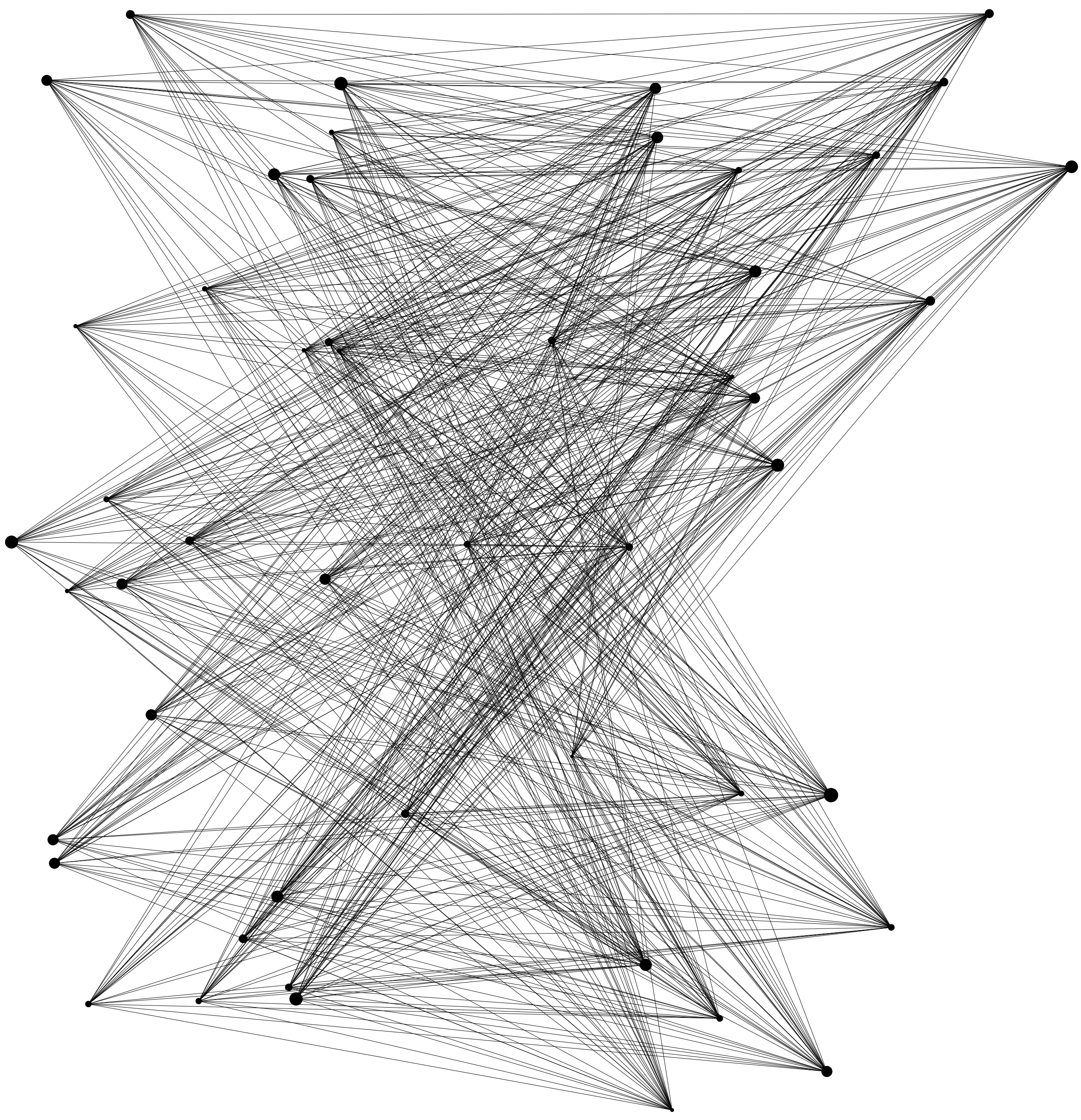
A better method ("divide and conquer"):

* divide points in half
* average the position, sum the masses
* "superpoint" represents effect on other half
    IF halves sufficiently separated
* transfer change back to points



N-body simulation (continued)

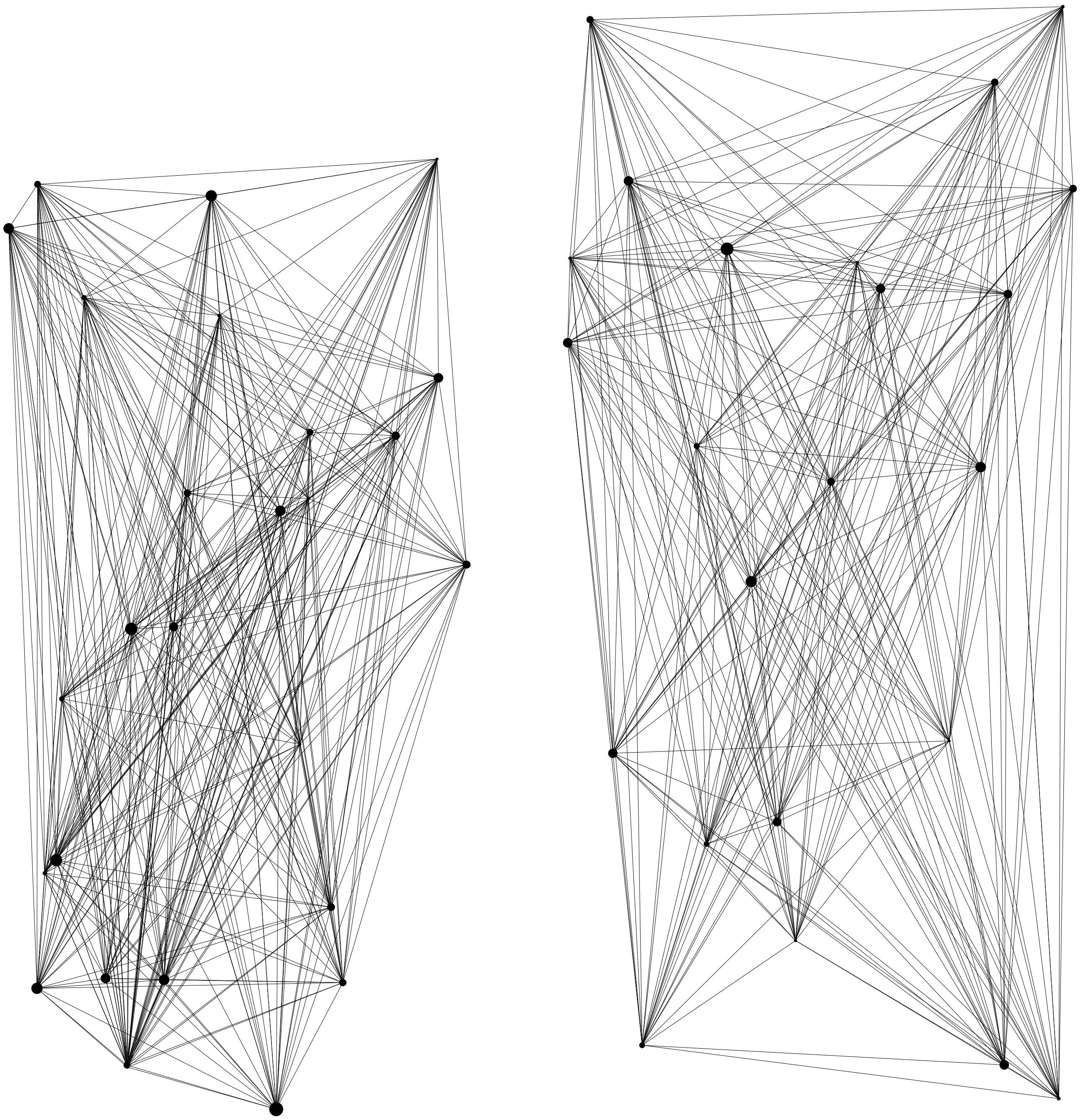
Divide-and-conquer eliminates computations
corresponding to edges between the halves



Not quite true near boundary
(stay tuned until "geometric algorithms")

N-body simulation (continued)

Plenty of edges still left within halves



Apply the method recursively

down to a few points in log N recursive calls

N-body simulation (continued)

Divide and conquer algorithm

* organize points in a tree structure
* treat groups of points as a single point
* total time proportional to M N log N

Better implementation

* use larger time steps
* use less precision
* tune inner loop
* floating point accelerator

Rough estimates for 1000 iterations, 10000 points

50 floating point ops in inner loop
2x10^5 iterations per time step
= 10^7 (ten billion) floating point ops

Factor of 10^5 cost savings
(ask an economist if that's OK!)

Better algorithm means the gap grows
as technology improves

If computers get 1000 times faster (they have!)
old way: can now do 300,000 points
new way: can now do 10,000,000 points

Ref: Appel's 1978 Princeton sr. thesis (!!)

SORTING

Elementary algorithms, Shellsort

Quicksort

Mergesort

Priority queues

Radix sorts

* Sort an array that fills memory
* Make the union of M spelling dictionaries
* Priority queue abstract data type

SEARCHING

Tree searching

Hashing

Trie searching

* Oxford English Dictionary
* Internet search engines
* DNA subsequence library
* Dictionary abstract data type

STRINGS

String searching

Pattern matching

File compression

* file systems, audio and video
* cryptology

GEOMETRIC ALGORITHMS

Elementary algorithms

Convex hull

Multidimensional searching

* N-body simulation
* "Toy Story"

GRAPH ALGORITHMS

Properties of graphs

Searching in graphs

Advanced graph algorithms

* Cycle detection
* matching (e. g. medical residents to jobs)
* networks

OTHER TOPICS

Mathematical algorithms

Dynamic programming

Parallel algorithms

Randomized algorithms

Intractable problems

COURSEWORK

Programming Assignments

* weekly, eleven in all
* electronic submission Thursdays 11:59PM
  (first one due 2/15)
* writeup due following Monday at 11:00AM

Problem Sets

* weekly, nine in all
* due Wednesday at 11:00AM
* first one due 2/14

Exams

* open book, notes
* midterm 3/13
* final at scheduled time

Online course information on homepage
For help, mail cs226@phoenix

READ (and do not lose) HANDOUT ONE
READ ONLINE INFORMATION