Module 08: Computing in Science and Engineering: Top 10 Algorithms of the Twentieth Century

Monday, March 31, 2014

(borrowing greatly from Wikipedia today)

Overview

- The editors goal was "to assemble the 10 algorithms with the greatest influence on the development and practice of science and engineering in the 20th century"
- My goal – introduce you to some CS and Math history

1946: Monte Carlo method

- John von Neumann, Stan Ulam, Nick Metropolis, Los Alamos Scientific Laboratory
- Aim: derive approximate solutions to numerical problems with many degrees of freedom and combinatorial problems of factorial size, by mimicking a random process
- The development was not just "using" random numbers, but developing algorithms to generate "pseudo" random numbers

1946: Monte Carlo method

- Applications (among many others):
  - Approximate multidimensional definite integrals, particularly with complicated areas of integration
  - central to the simulations required for the Manhattan Project and development of neutron bomb
  - Telecommunications: design of a wireless network depends on many factors: number of users, locations, services. MC methods used to generate users and related information. Evaluate performance and optimize as needed. *(Resource allocation problem)*
  - Artificial intelligence for games.

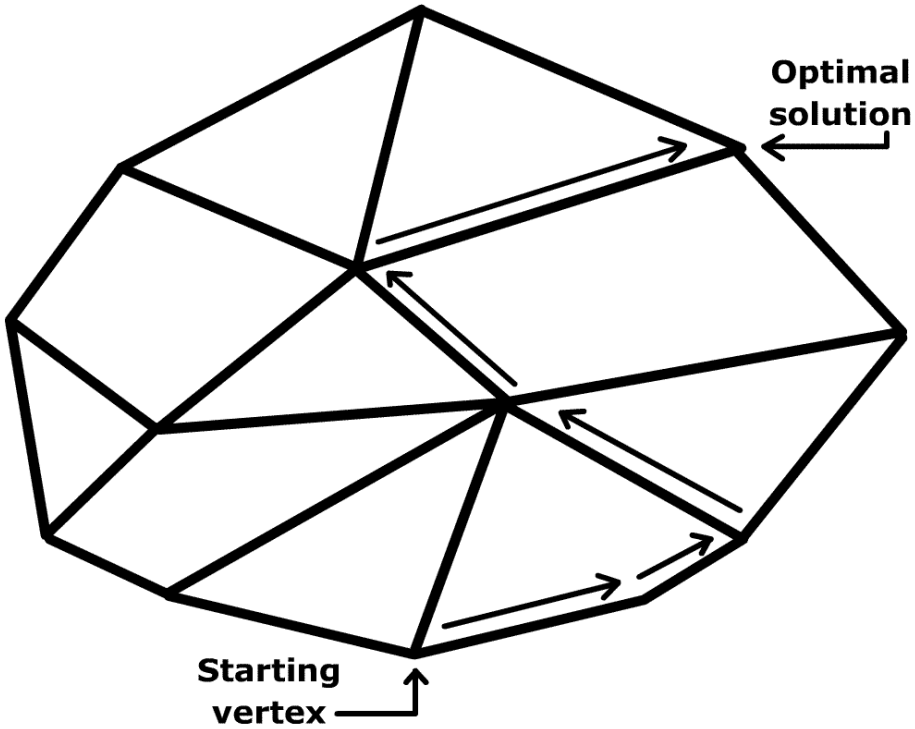
1947: Simplex method for linear programming

- George Dantzig, RAND corporation
- Linear progamming problems in standard form

$$\min_x c^T x$$

$$\text{such that } Ax = b, x \geq 0.$$

1947: Simplex method for linear programming



- Feasible region is convex polytope
- Solution is at a vertex
- Number of vertices is combinatorial
- Choose one vertex
  - If not optimal, there exists an edge along which objective function decreases
  - Move there
  - Repeat until optimal

Subject to stalling and cycling, but performs very well in practice.

1947: Simplex method for linear programming

- Lots of real world applications, e.g.
- Minimizing risk in investment portfolio subject to a certain return
- Assigning crews in airline industry
- Mimimize food costs subject to nutritional requirements
- Routing cell phone calls subject to existing networks and loads
- Many applications are really nonlinear, but simplified to linear to get approximate solutions

1950: Krylov subspace interation methods

- Magnus Hestenes, Eduard Stiefel, Cornelius Lanczos, Institute for Numerical Analysis at the National Bureau of Standards
- Solves Ax = b iteratively for HUGE problems
- Involves solving equations of the form

  $Kx_{j+1} = Kx_j + r_j$, where $r_j = b - Ax_j$,

  with a "simpler" matrix K which is close to A
- Motivated by idea of Krylov subspaces:

  $\text{Span}\{r, Ar, A^2r, \ldots\}$

1950: Krylov subspace interation methods

- Lanczos: found "nifty" way to generate orthogonal basis for subspace when A is symmetric
- Hestenes and Stiefel: found even "niftier" way if A is positive definite as well (conjugate gradient technique)
- In practice, faster than the iterative methods we studied (Jacobi and Gauss-Seidel)

1951: Decompositional approach to matrix computations

- Alston Householder, Oak Ridge National Laboratory
- Developed the Householder transformation that describes reflection about a plan or hyperplane containing the origin
- We used it to produce the tridiagonal matrix for the QR algorithm

1951: Decompositional approach to matrix computations

- Factoring matrices into triangular, diagonal, orthogonal, and other special forms led to development of efficient, specialized software for numerical linear algebra.
- Also aided in analysis of rounding errors.

1957: Fortran optimizing compiler

- John Backus and team, IBM
- Not first high level language, but first "optimizing" compiler which could produce code comparable to that produced by hand in assembly language
- Backus: "Much of my work has come from being lazy. I didn't like writing programs, and so, when I was working on the IMB701, writing programs for computing missile trajectories, I started work on a programming system to make it easier to write programs."

1957: Fortran optimizing compiler

- Initial release contained 32 statements, including
  - Assignment statements
  - Loops
  - Input/Output
  - If
  - Goto
- Quickly adopted by the scientific and engineering communities

1957: Fortran optimizing compiler

- Greatly updated over the years
- Fortran is still the language used for benchmarking the performance of the world's fastest super computers
- Many advances in compilers were motivated by the need for more efficient Fortran code

1959-61: QR algorithm

- J.G.F. Francis, Ferranti Ltd., London
- Stable method of computing eigenvalues
- Eigenvalues are "arguably most important numbers associated with matrices"
- By mid-1960s, the QR algorithm had made eigenvalue determination into routine calculations

1962: Quicksort

- Tony Hoare, Elliott Brothers, London
- Fast, general purpose sorting algorithm
- On average: O(n log n)
- Worst case: $O(n^2)$
- Idea:
  - Choose a pivot value
  - Divide into 3 groups: < pivot, = pivot, > pivot
  - Recursively sort the < and > groups.

1965: Fast Fourier transform

- James Cooley, IBM T.J. Watson Research Centre
- John Tukey, Princeton University and AT&T
- Underlying idea traces back to Gauss
- Cooley-Tukey showed how easily the coefficients could be calculated

1977: Integer relation detection algorithm

- Helaman Ferguson and Rodney Forcade, Brigham Young University
- Given a set of real numbers: $x_1, x_2, \ldots, x_n$
- Are there integers $a_1, a_2, \ldots, a_n$ (not all 0) for which $a_1x_1 + a_2x_2 + \ldots a_nx_n = 0$?
- Applications
  - Bifurcation theory (e.g. study of changes in solutions of families of differential equations)
  - Simplify calculations with Feynman diagrams in quantum field theory
  - Determine if a real number x is algebraic

1987: Fast multipole algorithm

- Leslie Greengard and Vladimir Rokhlin, Yale University
- Problem of N-body simulations:
  - Relate to a dynamical system of particles, such as gravity
  - Astrophysics
  - Physical cosmology
  - Dynamic evolution of star clusters
- Major issue: calculations for each pair of particles required $O(n^2)$ computations
- This algorithm required O(n) computations for each pair
- Includes rigorous error estimates