CSE 417: Algorithms and Computational Complexity
Lecture I: Overview

Imran Rashid

University of Washington

Jan 7, 2008

Course Web Page

http://www.cs.washington.edu/417

What the course is about

- Design of Algorithms
  - design methods
  - common or important types of problems
  - analysis of algorithms - efficiency
  - correctness proofs

What the course is about

- Complexity, NP-completeness and intractability
  - solving problems in principle is not enough
    - algorithms must be *efficient*
  - some problems have *no efficient solution*
  - NP-complete problems
    - important & useful class of problems whose solutions (seemingly) cannot be found efficiently, but *can* be checked easily

Very Rough Division of Time

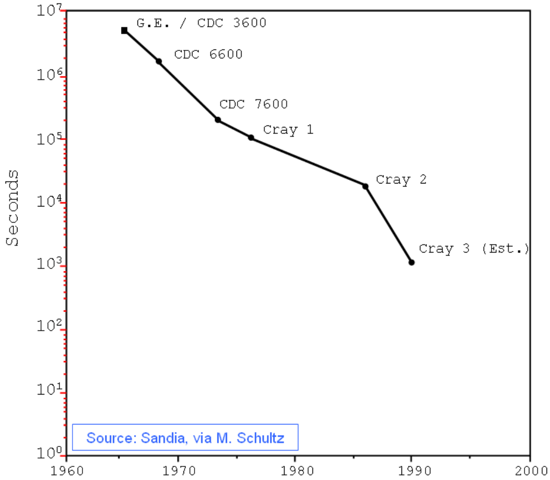
- Algorithms (7 weeks)
  - Analysis of Algorithms
  - Basic Algorithmic Design Techniques
  - Graph Algorithms
- Complexity & NP-completeness (3 weeks)
- Check online schedule page for (evolving) details

Complexity Example

- Cryptography (e.g. RSA, SSL in browsers)
  - Secret: p,q prime, say 512 bits each
  - Public: n which equals p x q, 1024 bits
- In principle
  - *there is an algorithm* that given n will find p and q: try all 2^{512} possible p's, an astronomical number
- In practice
  - *no efficient algorithm* is known for this problem
  - security of RSA depends on this fact

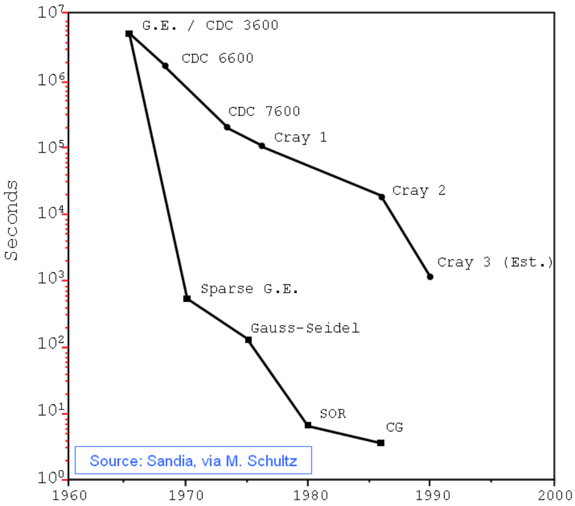
Algorithms or Hardware?

- 25 years progress solving sparse linear systems
- hardware: 4 orders of magnitude
- software: 6 orders of magnitude



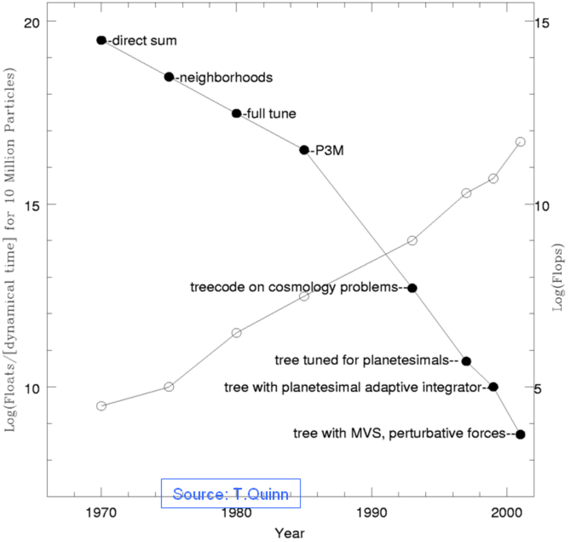
Algorithms or Hardware?

- 25 years progress solving sparse linear systems
- hardware: 4 orders of magnitude
- software: 6 orders of magnitude



Algorithms or Hardware?

- The N-Body Problem:
- in 30 years
  - 10^7 hardware
  - 10^{10} software

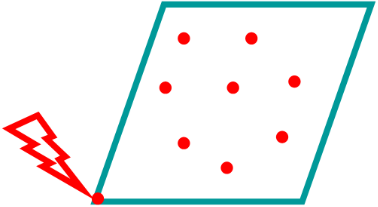


Algorithm: definition

- Procedure to accomplish a task or solve a well-specified problem
  - Well-specified: know what all possible inputs look like and what output looks like given them
  - "accomplish" via simple, well-defined steps
  - Ex: sorting names (via comparison)
  - Ex: checking for primality (via +, -, *, /,)

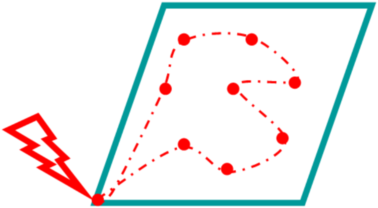
Algorithms: a sample problem

- Printed circuit-board company has a robot arm that solders components to the board
- Time: proportional to total distance the arm must move from initial rest position around the board and back to the initial position
- For each board design, find best order to do the soldering



Algorithms: a sample problem

- Printed circuit-board company has a robot arm that solders components to the board
- Time: proportional to total distance the arm must move from initial rest position around the board and back to the initial position
- For each board design, find best order to do the soldering



A Well-defined Problem

- Input: Given a set S of n points in the plane
- Output: The shortest cycle tour that visits each point in the set S.
- Better known as "TSP"
- How might you solve it?

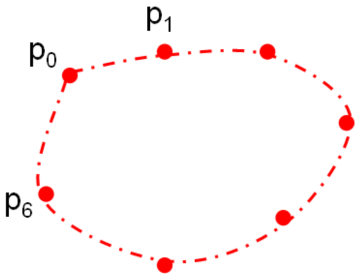
Nearest Neighbor Heuristic

- heuristic A rule of thumb, simplification, or educated guess that reduces or limits the search for solutions indomains that are difficult and poorly understood.
  - May be good, but usually not guaranteed to give the best or fastest solution.

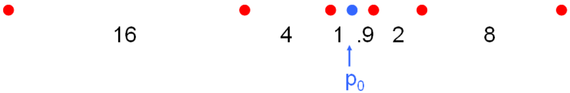
1. Start at some point p_0
2. Walk first to its nearest neighbor p_1
3. Repeatedly walk to the nearest unvisited neighbor p_2, then $p_3, \ldots$ until all points have been visited
4. Then walk back to p_0

Nearest Neighbor Heuristic

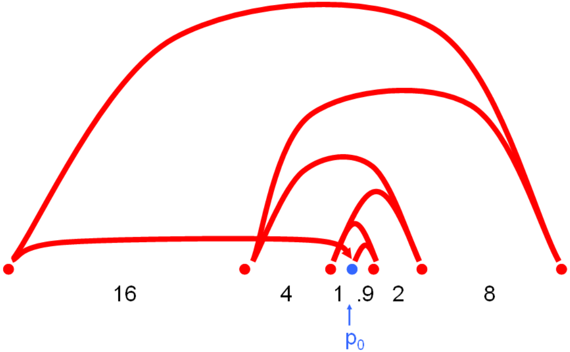
1. Start at some point p_0
2. Walk first to its nearest neighbor p_1
3. Repeatedly walk to the nearest unvisited neighbor p_2, then $p_3, \ldots$ until all points have been visited
4. Then walk back to p_0



An input where it works badly

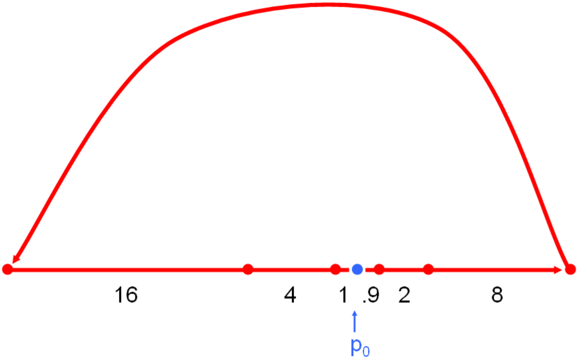


An input where it works badly



- Nearest Neighor Length= 84

An input where it works badly



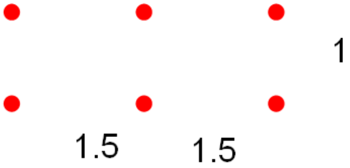
- Nearest Neighor Length= 84
- Optimal Length = 64

Revised idea - Closest pairs first

- Repeatedly join the closest pair of points
  - (s.t. result can still be part of a single loop in the end. I.e., join endpoints, but not points in middle, of path segments already created.)
- How does this work on our bad example?

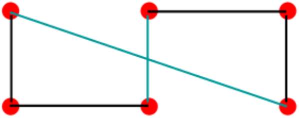
Another bad example

Closest Pairs



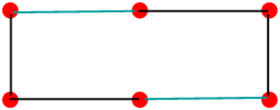
Another bad example

Closest Pairs



- Length $=$ $6 + \sqrt{10} \approx 9.16$

Optimal



- Length $= 8$

Something that works

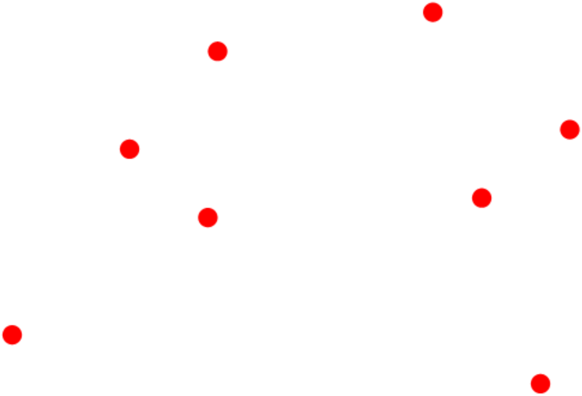
- For each of the $n! = n(n-1)(n-2)\dots 1$ orderings of the points, check the length of the cycle you get
- Keep the best one

Two Notes

- The two incorrect algorithms were greedy
  - Often very natural & tempting ideas
  - They make choices that look great "locally" (and never reconsider them)
  - When greed works, the algorithms are typically efficient
  - BUT: often does not work - you get boxed in
- Our correct alg avoids this, but is incredibly slow
  - 20! is so large that checking one billion per second would take 2.4 billion seconds (around 70 years!)

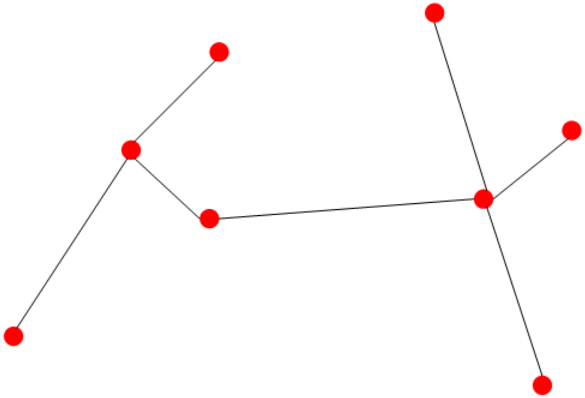
Something that "works" (differently)

1. Find Min Spanning Tree
2. Walk around it
3. Take shortcuts (instead of revisiting)



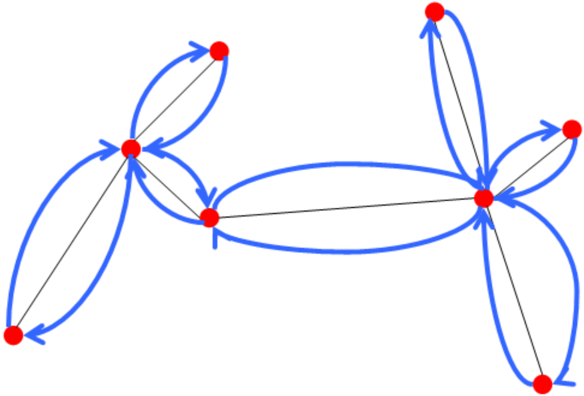
Something that "works" (differently)

1. Find Min Spanning Tree
2. Walk around it
3. Take shortcuts (instead of revisiting)



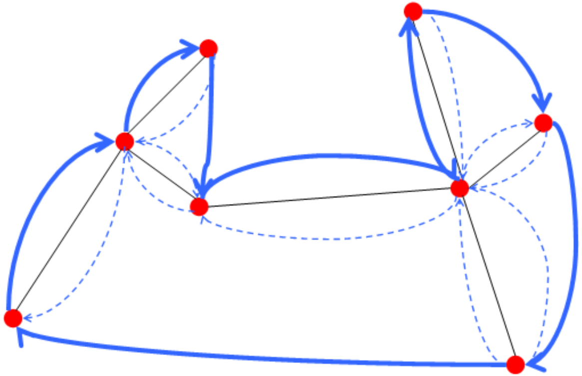
Something that “works” (differently)

1. Find Min Spanning Tree
2. Walk around it
3. Take shortcuts (instead of revisiting)



Something that "works" (differently)

1. Find Min Spanning Tree
2. Walk around it
3. Take shortcuts (instead of revisiting)



Guarenteed Approximation

- Maybe seems a little wacky ...
- but its always within a factor of 2 of the best tour!

Proof.

Deleting one edge from best tour gives a spanning tree, so:

$$\text{Min spanning tree} < \text{best tour}$$
$$\text{best tour} \leq \text{wacky tour}$$
$$\text{wacky tour} \leq 2 * MST \leq 2 * \text{best}$$

□

The Morals of the Story

- Simple problems can be hard
  - Factoring, TSP
- Simple ideas don't always work
  - Nearest neighbor, closest pair heuristics
- Simple algorithms can be very slow
  - Brute-force factoring, TSP
- Changing your objective can be good
  - Guaranteed approximation for TSP