

CSE 417: Algorithms and Computational Complexity

I: Organization & Overview

Winter 2007

Larry Ruzzo



University of Washington

Computer Science & Engineering

CSE 417, Wi '07: Algorithms & Computational Complexity

[CSE Home](#)

[About Us](#) [Search](#) [Contact Info](#)

Administrative

[FAQ](#)
[Schedule & Reading](#)

Email

[Class List Archive](#)
[E-mail Course Staff](#)

Assignments

Solutions

Lecture Notes

- [1: Overview & Example \('06\)](#)
- [2-3: Analysis \('06\)](#)
- [4-6: Graphs, B/DFS \('06\)](#)
- [7-11: Greedy](#)
 - [Scheduling \('06\)](#)
 - [Huffman \('06\)](#)
- [12-15: Dyn. Prog.](#)
 - [Fibonacci \('06\)](#)
 - [Stamps \('06\)](#)
 - [RNA Structure \('06\)](#)
 - [Scheduling \('06\)](#)
- [16-19: Divide & Conquer](#)
- [20-26: P & NP](#)

Lecture: [Low 101](#) (schematic) MWF 2:30- 3:20

Office Hours	Phone
MF? 12:00- 1:00? CSE 554	206-543-6298

Instructor: Larry Ruzzo, ruzzo_at_cs
TA: Zizhen Yao, yzizhen_at_cs TBA

Course Email: cse417a_wi07@u.washington.edu. Use this list to ask and/or answer questions about homework, lectures, etc. The instructor and TA are subscribed to this list. All messages are automatically [archived](#). Questions not of general interest may be directed to the instructor and TA: cse417-staff or just to the instructor: ruzzo@cs. You can (and perhaps should) [change your subscription options](#).

Catalog Description: Design and analysis of algorithms and data structures for manipulating graphs and strings. Fast Fourier Transform, Turing machines. Time and space complexity. NP-completeness.

Prerequisite: [CSE 373](#)

Credits: 3

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

... mix of paper & pencil exercises and ... 50%, midterm 15%, final 30%.

... assignments are due at the **start** of class on the due date. 10% off for up to one business day, e.g., Monday for Friday due dates); additional 20% per day thereafter.

- [Algorithm Design](#) by [Jon Kleinberg](#) and [Eva Tardos](#). Addison Wesley, 2006. (Available from [U Book Store](#), [Amazon](#), etc.)

Based on past experience, we will probably have little if any time to cover the "computability" material outlined in the catalog description. If you want additional material on these topics, as well as an alternative presentation of the computational complexity/NP-completeness topics, I recommend:

What you'll have to do

Homework (~55% of grade)

Programming

Several small projects

Written homework assignments

English exposition and pseudo-code

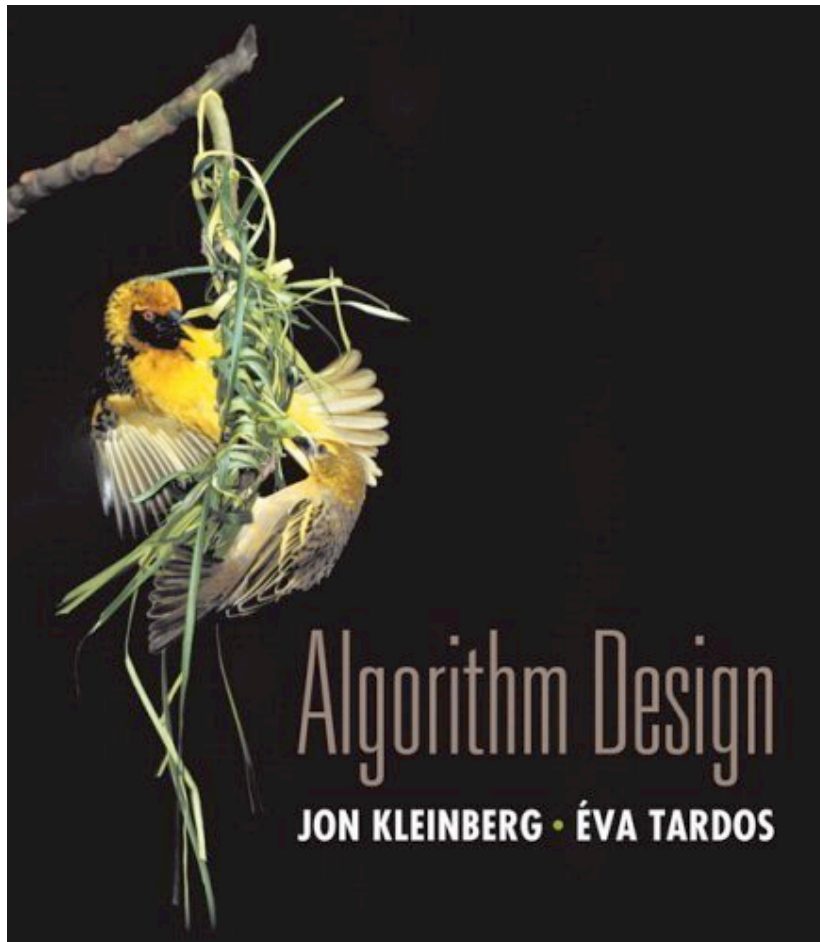
Analysis and argument as well as design

Midterm / Final Exam (~15% / 30%)

Late Policy:

Papers and/or electronic turnins are due at the *start* of class on the due date. 10% off for one day late (Monday, for Friday due dates); 20% per day thereafter.

Textbook



Algorithm Design by Jon Kleinberg and Eva Tardos. Addison Wesley, 2006.

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



University of Washington
Computer Science & Engineering

CSE 417, Wi '06: *Approximate* Schedule

[CSE Home](#)

[About Us](#) [S](#)

		Due	Lecture Topic	Reading
Week 1 1/2-1/6	M		Holiday	
	W		Intro, Examples & Complexity	Ch. 1; Ch. 2
	F		Intro, Examples & Complexity	
Week 2 1/9-1/13	M		Intro, Examples & Complexity	
	W		Graph Algorithms	Ch. 3
	F		Graph Algorithms	

Complexity Example

Cryptography (e.g. RSA, SSL in browsers)

Secret: p, q prime, say 512 bits each

Public: n which equals $p \times q$, 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 versus Machines

We all know about Moore's Law and the exponential improvements in hardware...

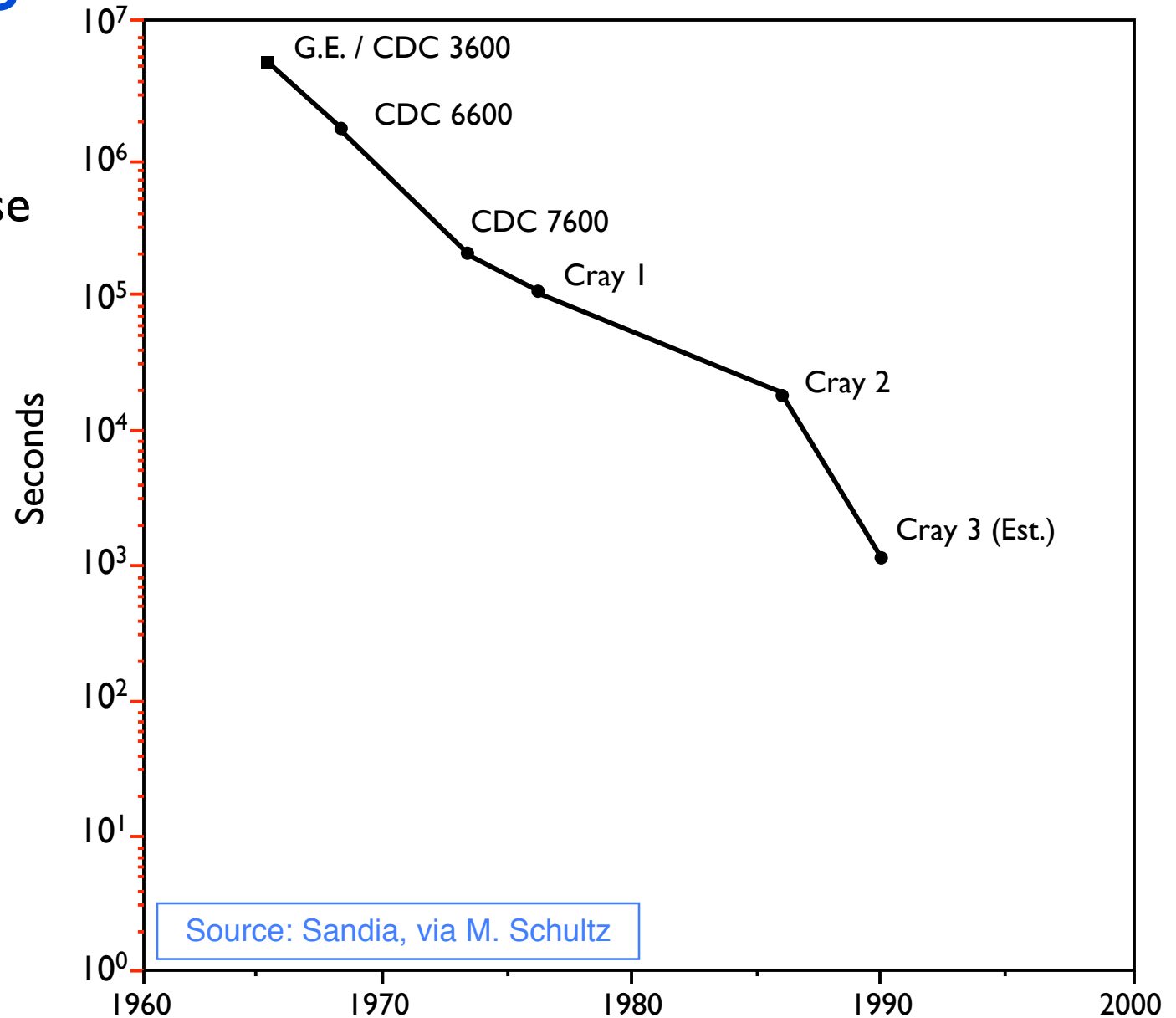
Ex: sparse linear equations over 25 years

10 orders of magnitude improvement!

Algorithms or Hardware?

25 years
progress
solving sparse
linear
systems

hardware: 4
orders of
magnitude

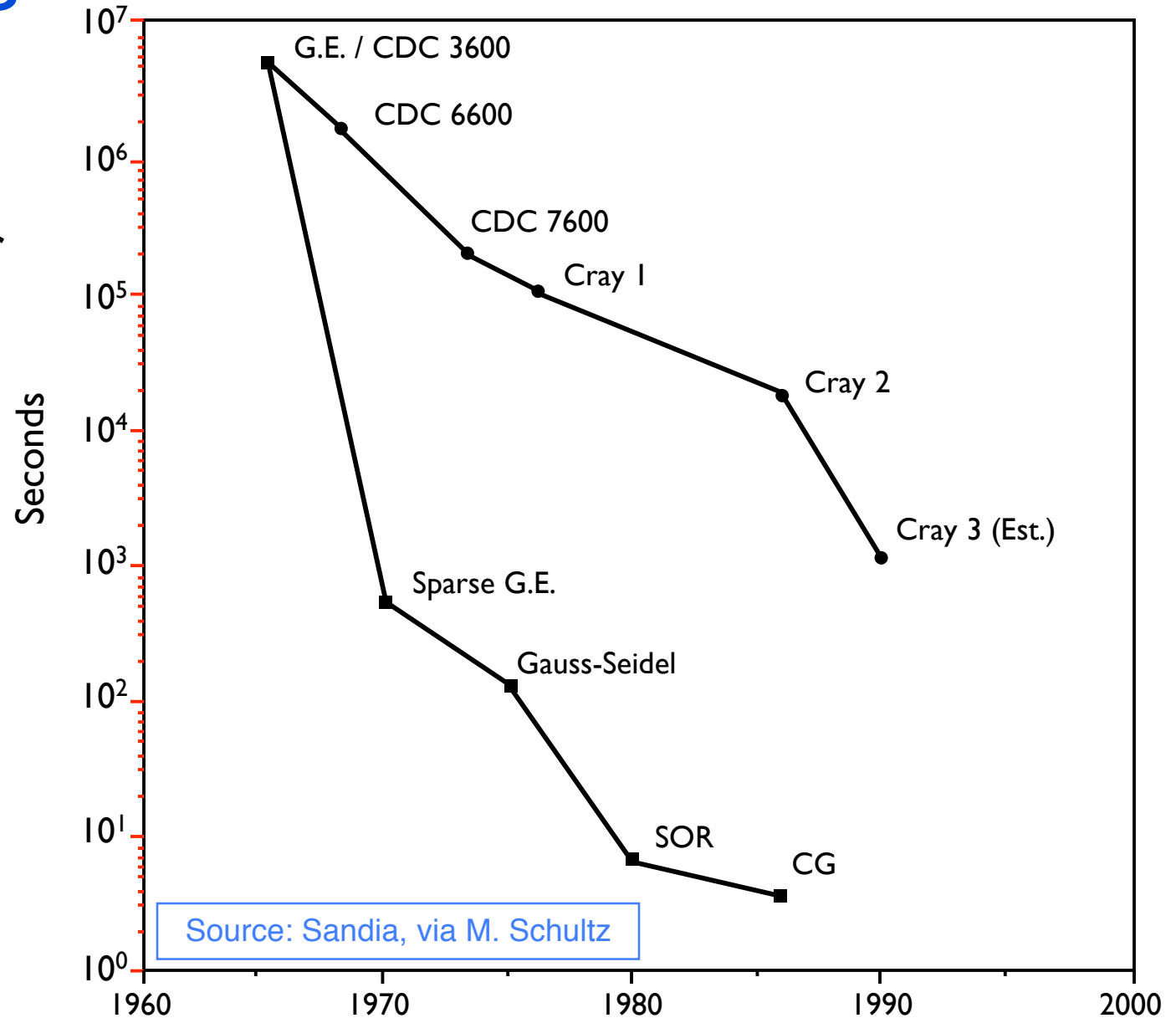


Algorithms or Hardware?

25 years
progress
solving
sparse linear
systems

hardware: 4
orders of
magnitude

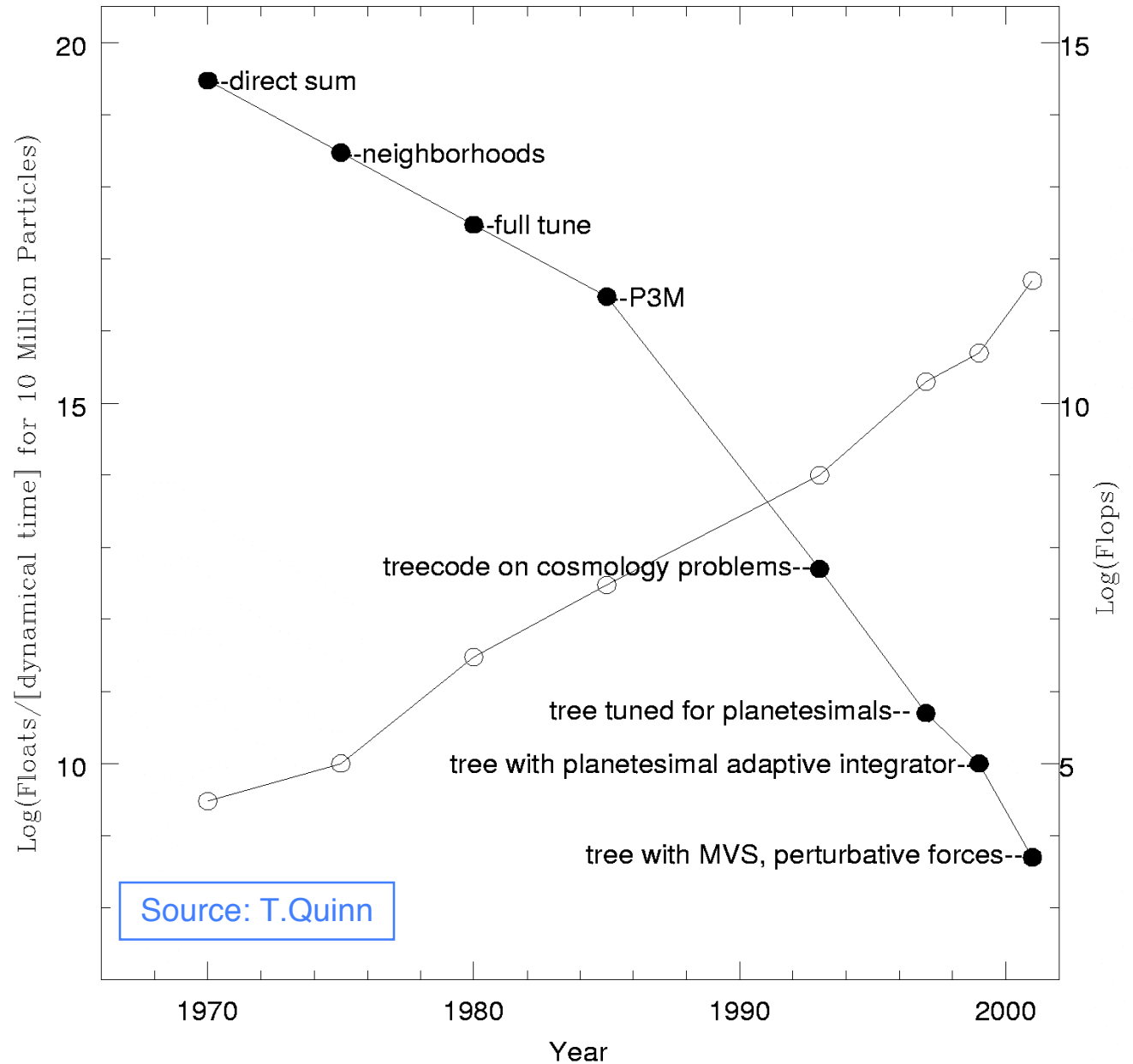
software: 6
orders of
magnitude



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 $+$, $-$, $*$, $/$, \leq)

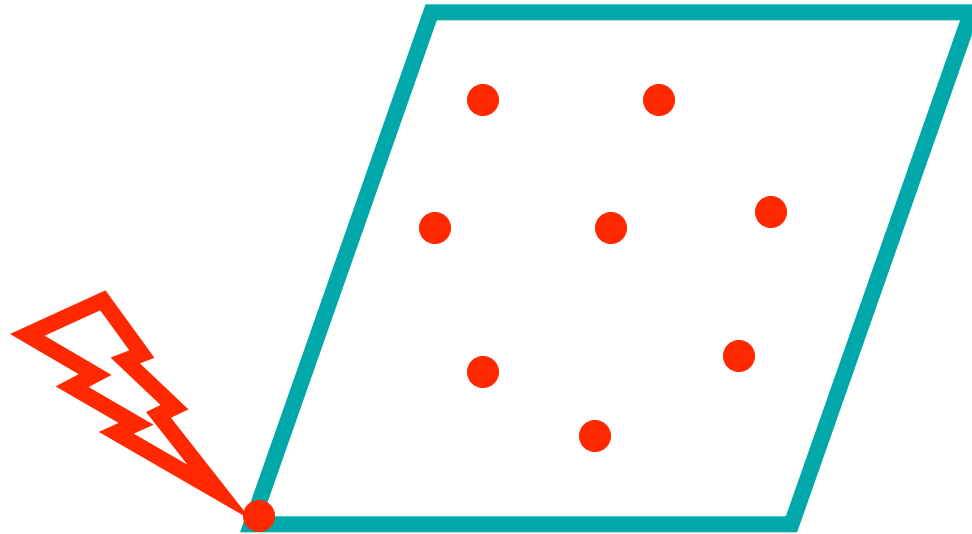
Algorithms: a sample problem

Printed circuit-board company has a robot arm that solders components to the board

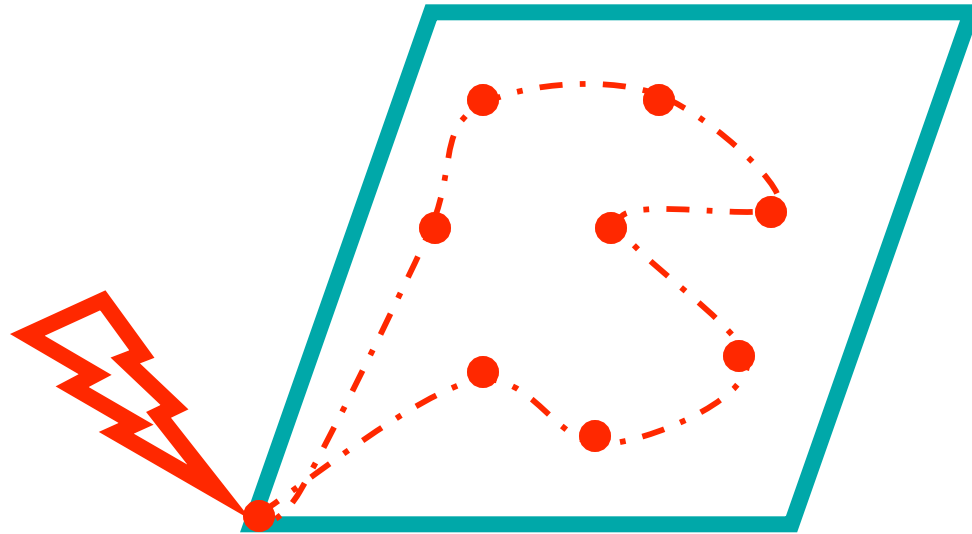
Time: proportional to total distance the arm must move from initial rest position around the board and back to the initial position

For each board design, find best order to do the soldering

Printed Circuit Board



Printed Circuit Board



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

Start at some point p_0

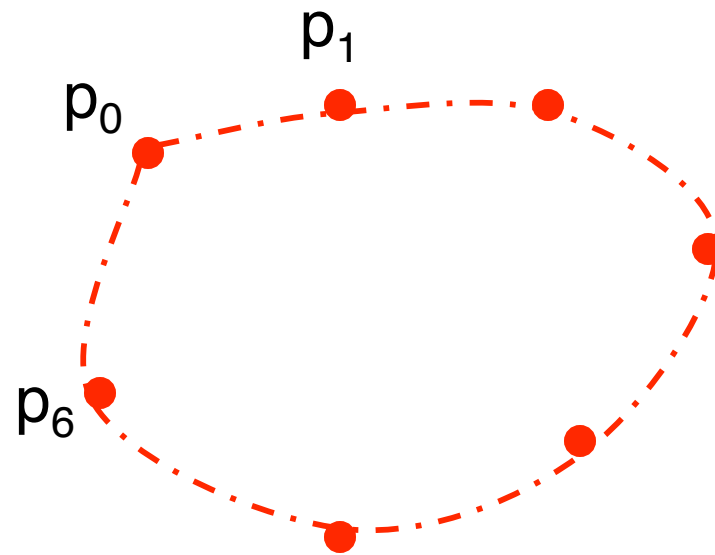
Walk first to its
nearest neighbor p_1

Repeatedly walk to the nearest unvisited neighbor
 p_2 , then p_3, \dots until all points have been visited

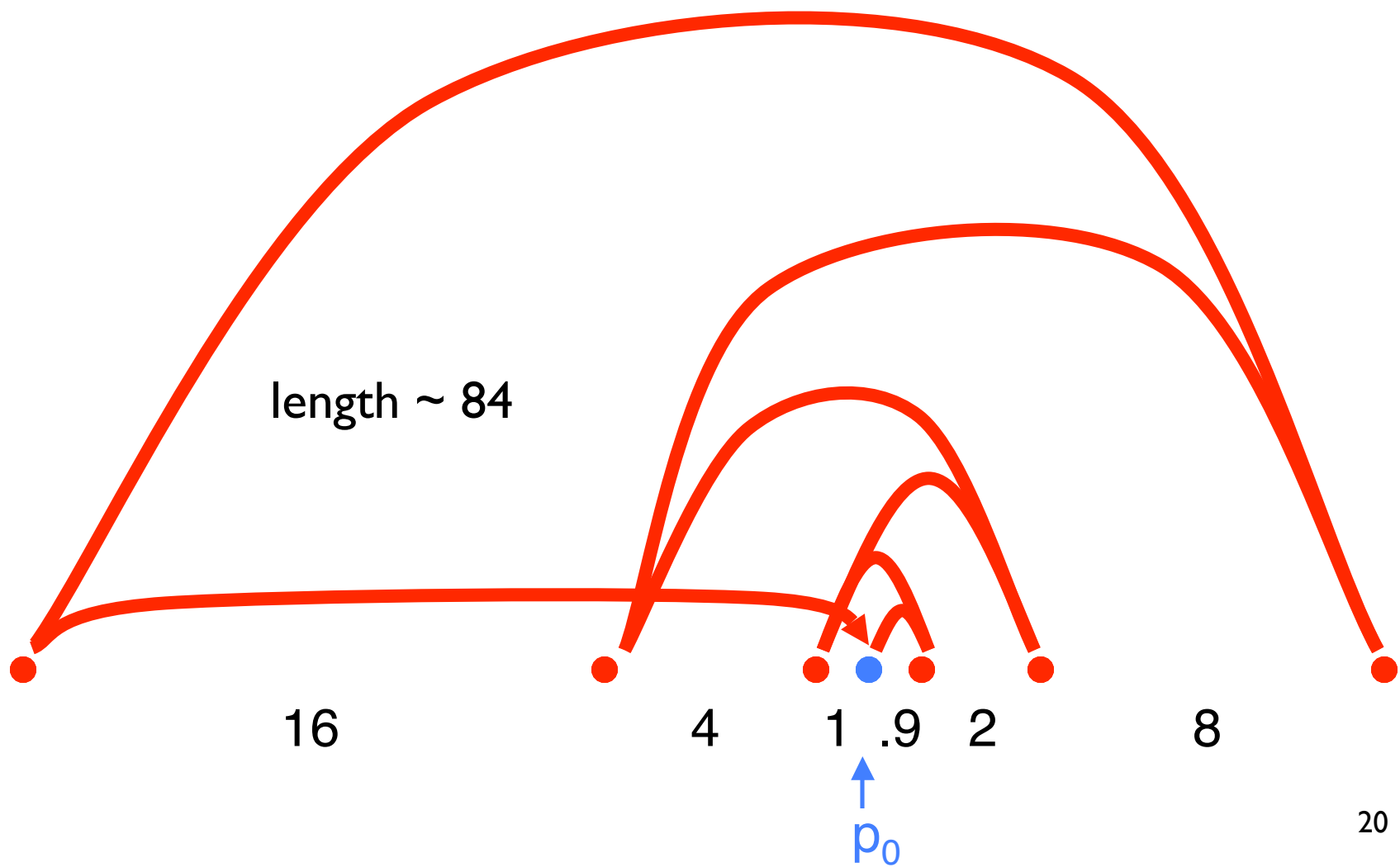
Then walk back to p_0

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

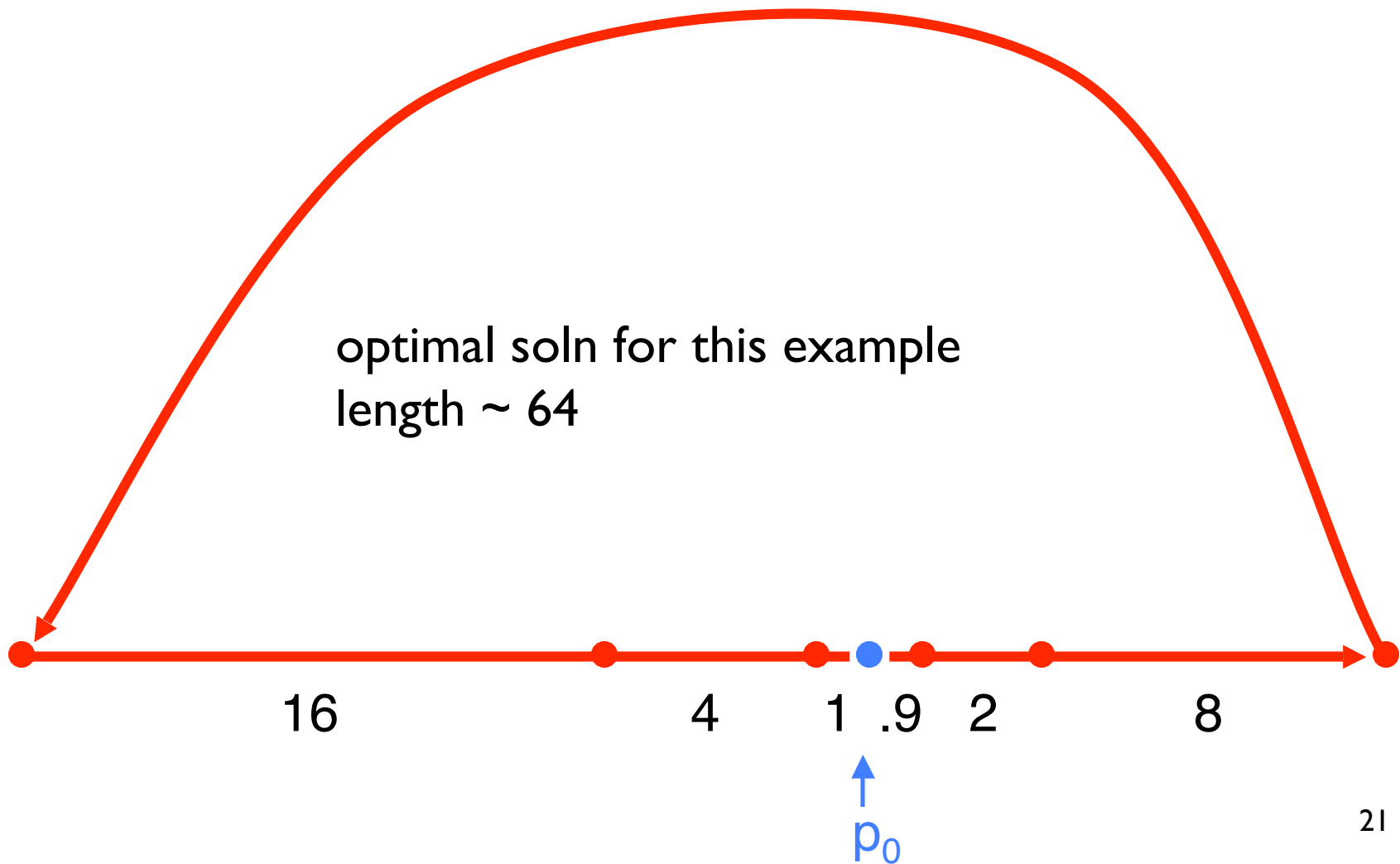
Nearest Neighbor Heuristic



An input where it works badly



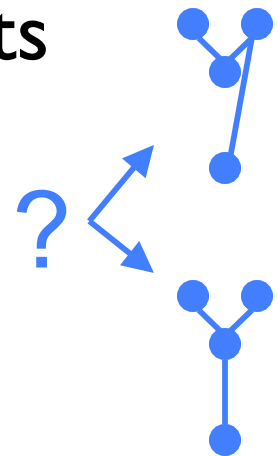
An input where it works badly



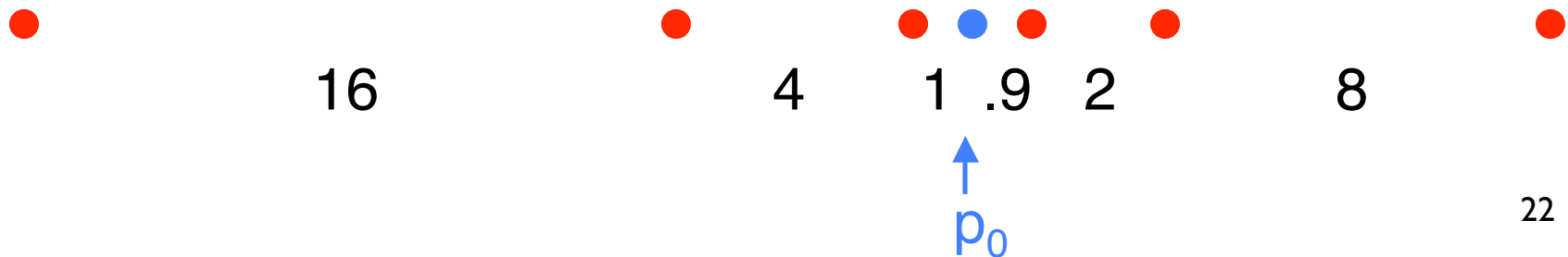
Revised idea - Closest pairs first

Repeatedly join the closest pair of points

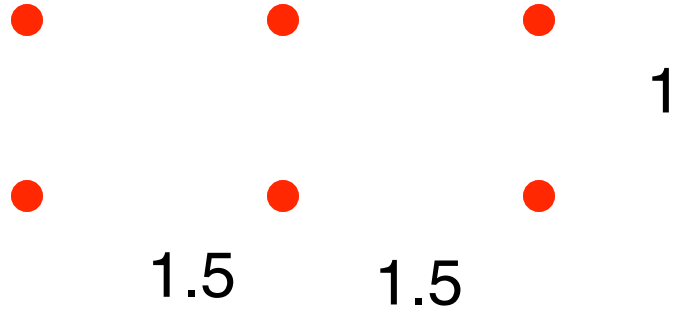
(s.t. result can still be part of a single loop in the end. I.e., join endpoints, but not points in middle, of path segments already created.)



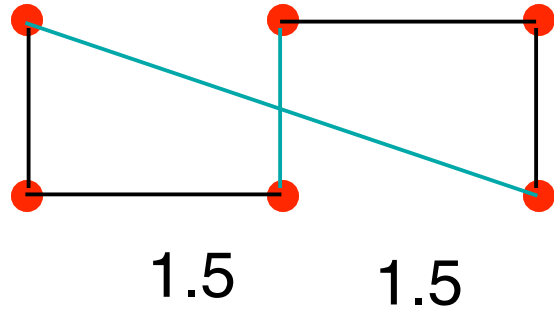
How does this work on our bad example?



Another bad example



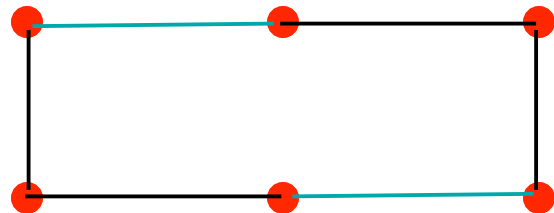
Another bad example



1

$$6 + \sqrt{10} = 9.16$$

vs



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

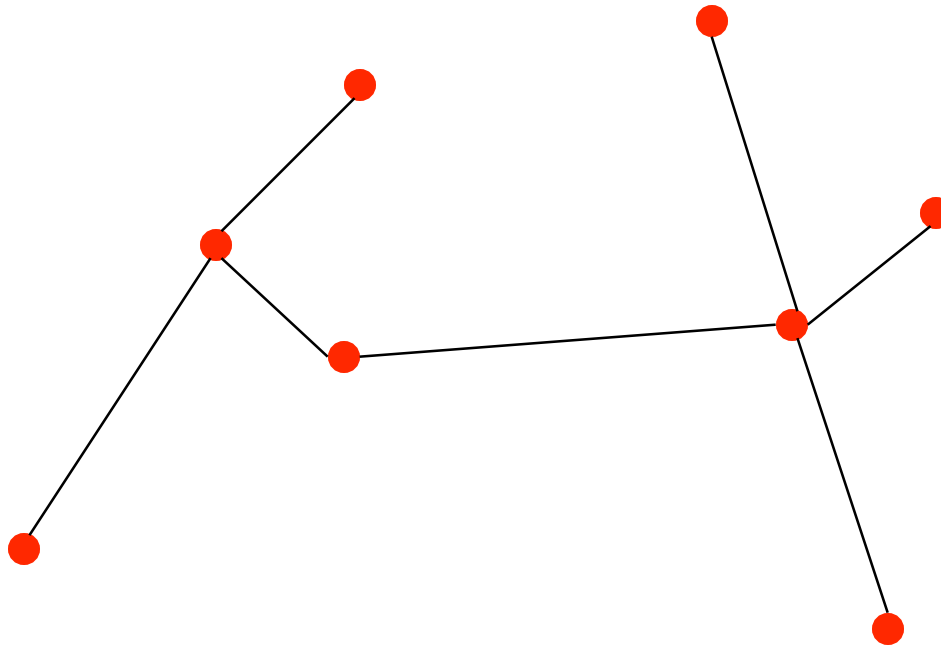
- BUT: often does not work - you get boxed in

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!)

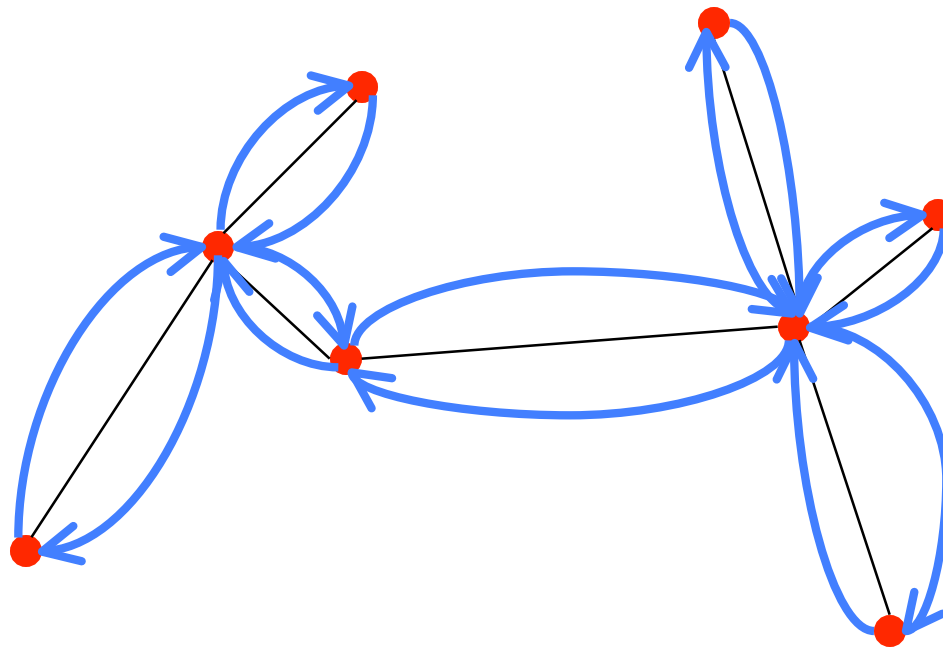
Something that “works” (differently)

I. Find Min Spanning Tree



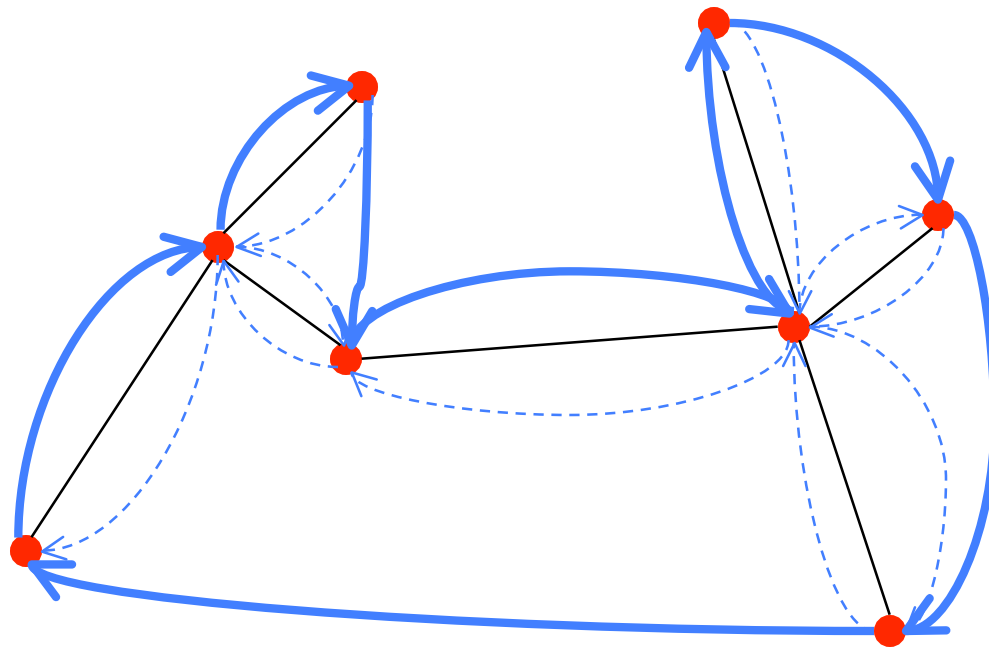
Something that “works” (differently)

2. Walk around it



Something that “works” (differently)

3. Take shortcuts (instead of revisiting)



Something that “works” (differently): Guaranteed Approximation

Does it seem wacky?

Maybe, but it's *always* within a factor of 2 of the best tour!

deleting one edge from best tour gives a spanning tree, so *Min* spanning tree < best tour

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

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