CSE 417: Algorithms and Computational Complexity

2: Algorithms and Efficiency

Winter 2005

Larry Ruzzo

1

Algorithms: 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 +, -, *, /)

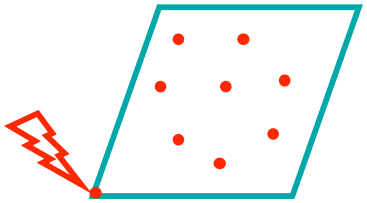
2

Algorithms: a sample problem

- Printed circuit-board company has a robot arm that solders components to the board
- Time to do it depends on
  - total distance the arm must move from initial rest position around the board and back to the initial positions
- For each board design, must figure out good order to do the soldering

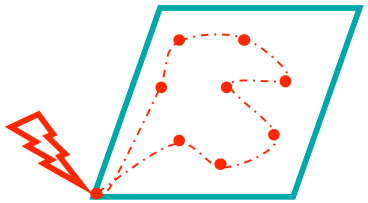
3

Printed Circuit Board



4

Printed Circuit Board



5

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?

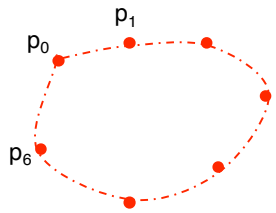
6

Nearest Neighbor Heuristic

- Start at some point p_0
- Walk first to its nearest neighbor p_1
- Repeatedly walk to the nearest unvisited neighbor until all points have been visited
- Then walk back to p_0

7

Nearest Neighbor Heuristic

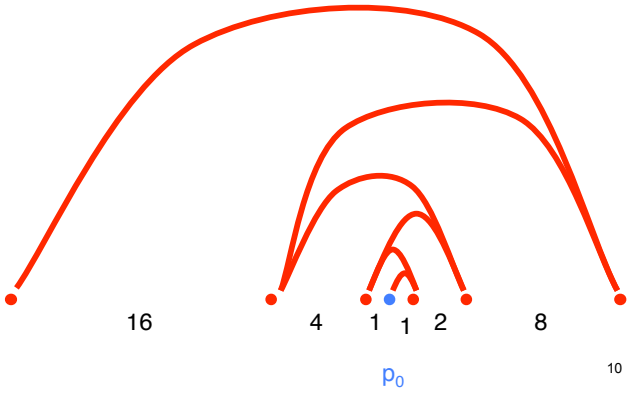


8

An input where it works badly

9

An input where it works badly



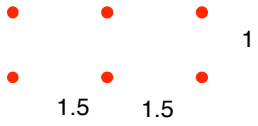
10

Revised idea - Closest pairs first

- Repeatedly pick the closest pair of points to join so that the result can still be part of a single loop in the end
  - can pick endpoints of line segments already created
- How does this work on our bad example?

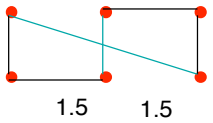
11

Another bad example



12

Another bad example



$1+\sqrt{10}$ vs 3

Something that works

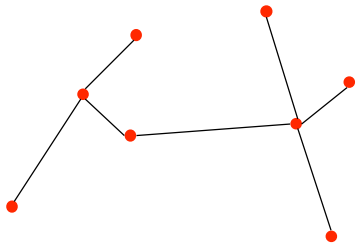
- For each of the n! orderings of the points check the length of the cycle you get
- Keep the best one

Two Notes

- The two incorrect algorithms were **greedy**
  - Often very natural & tempting ideas
  - they make choices that look great "locally" (and never reconsidered them)
  - often does not work - you get boxed in
    - when it does the algorithms are typically efficient
- Our correct algorithm avoids this, but is incredibly slow
  - 20! is so large that counting to one billion in a second it would still take 2.4 billion seconds
    - (around 70 years!)

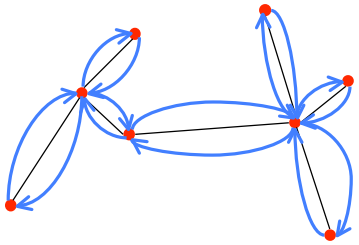
Something that "works" (differently)

1. Find Min Spanning Tree



Something that "works" (differently)

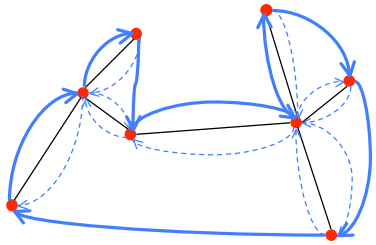
2. Walk around it



17

Something that "works" (differently)

3. Take shortcuts (instead of revisiting)



18

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 ≤ wacky tour ≤ 2 * MST ≤ 2 * best

19

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

20