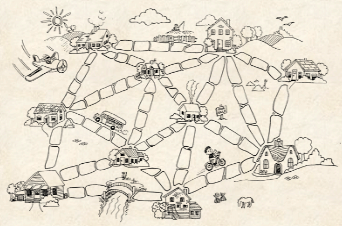
UNIVERSITY of WISCONSIN-MADISON
Computer Sciences Department

CS 202: Introduction to Computation

Professor Andrea Arpaci-Dusseau

What problems stretch the limits of computation?



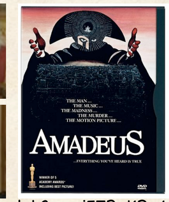
Discussion

Is there an inherent difference between

being brilliant

and

being able to appreciate brilliance?



http://www.youtube.com/watch?v=-ciFTP_KRy4

What is Brilliance?

Ability to find "needle in a haystack"

- Mozart found "satisfying assignments" to our neural circuits for music appreciation
- Relatively easy to identify the fact needle has been found

What is a computational analogue of this phenomenon?

Many hard computation problems require solutions involving "finding a needle in a haystack"

Compare 4 Algorithms

1. Identify path between nodes of graph
2. Minimal spanning tree
3. Monkey puzzle
4. Travelling salesperson

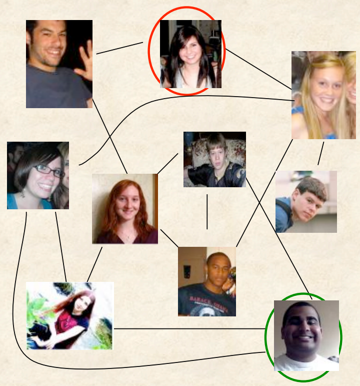
Which ones are easy and which are hard to solve?

Problem 1: Path?

Social network or graph
- Each node represents student
- Two nodes connected by edge if those students are friends

Scenario:
- Kate starts a rumor
- Will it reach Ronak?
- Is there a path or connection between two?

What algorithm could you use?



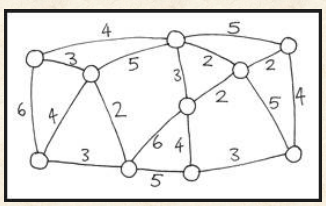
Problem 1: Path?

Social network or graph
- Each node represents student
- Two nodes connected by edge if those students are friends

Scenario:
- Julia starts a rumor
- Will it reach Ronak?
- Is there a path or connection between two?

How does running time depend on network size (number of edges, E)?
- Never need to visit an edge more than once
- At most O(E)
- Not a hard problem

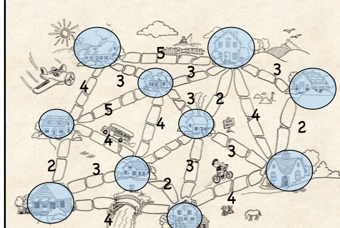
Useful Representation: Weighted Graph



Nodes connected by edges

Edges have "weights" associated with them
- Some cost (or distance) associated with that edge

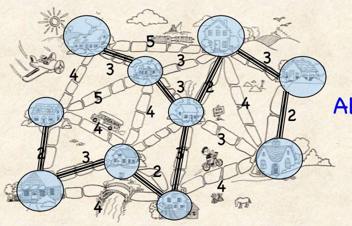
Problem 2: Spanning Trees



Goal: Connect all houses with shortest path
- Uses: roads and utilities
- Uses: Wiring chips on circuit boards

Weighted graph?
- Transform each house to node
- Each existing road to edge
- Label each edge with distance between houses

Problem 2: Spanning Trees



Goal: Connect all houses (nodes) with shortest path (edges)

- Uses: roads and utilities
- Uses: Wiring chips on circuit boards

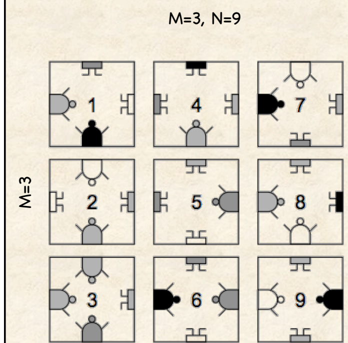
Algorithm?

- Greedy: make step-by-step decisions that work best for current situation
- Begin: Connect closest pair of nodes
- Each step: Connect to next closest
- Don't need to look at different combinations

Multiple equally good solutions

Is not a "hard" problem

Problem 3: Monkey Puzzle



Given:

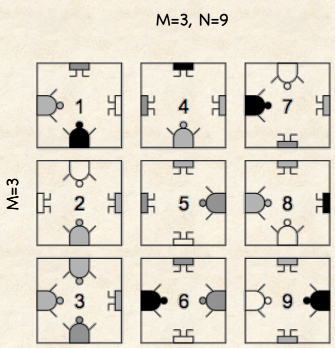
- Set of N square cards with top and bottom halves of colored monkeys
- $N = M^2$

Problem:

- Is there an arrangement of cards such that each pair of adjacent cards completes monkey?

Algorithm?

Problem 3: Monkey Puzzle

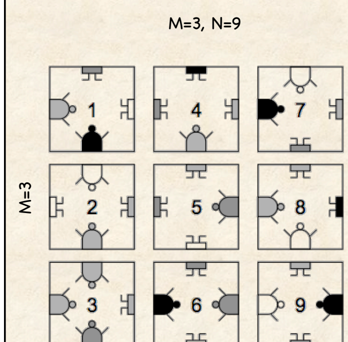


Try every combination of cards, with 4 rotations, and see if it works

- Pick a card for 1st box
- Try each of remaining cards in 2nd box until find possible match
- Try each of remaining cards in 3rd box…
- Etc…

Does a greedy algorithm work?

Problem 3: Monkey Puzzle



How many combinations possible?

- (9 * 4) * (8 * 4) * (7 * 4) * (6 * 4) * (5 * 4) * (4 * 4) * (3 * 4) * (2 * 4) * (1 * 4) = 4 * 9 Factorial = 4 * 9!
- No polynomial-time solutions are known

Analysis of Monkey Puzzle

For N cards, number of arrangements to examine is O(N!)

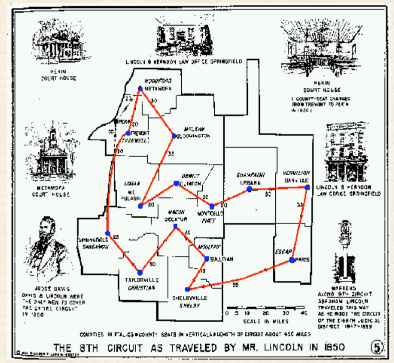
Assume can analyze one arrangement in 1 microsecond

How long to solve for N=9, 16, 25?

N	Time to analyze	
9		
16		
25		

Requires brilliance to solve quickly!

Problem 4: Travelling Salesperson (TSP)




SCIENCE

Santa Claus and the traveling salesman problem

Politicians

Visiting all ball parks in US

Collecting coins from meters

Delivering mail

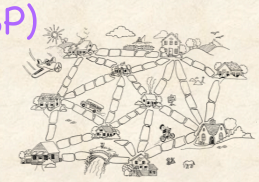
Star imagery

DNA sequencing

Computer networks

Power cables

Problem 4: Traveling Salesperson (TSP)

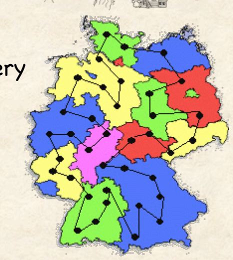


Given:

- Weighted graph of nodes for cities and edges for paths (weight is length)

Problem:

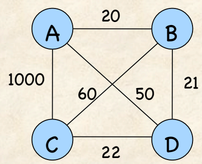
- Is there a route thru every city (and back to start) with cost < K?
  - Can't revisit same cities



Greedy Algorithm?

Try a Greedy Algorithm

Small graph with 4 cities



Find two closest cities
- A – B
- (20)

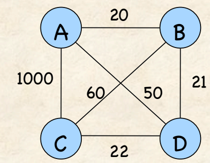
Connect next closest
- A – B – D
- (20 + 21)

Connect next closest
- A – B – D – C
- (20 + 21 + 22)

Connect back to start (A)
- A – B – D – C – A
- (20 + 21 + 22 + 1000 = 1063)

Greedy approach does not work for TSP!

Enumerate All Paths…



A – B – C – D – A
20 + 60 + 22 + 50 = 152

A – B – D – C – A
20 + 21 + 22 + 1000 = 1063

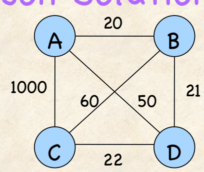
A – C – B – D – A
1000 + 60 + 21 + 50 = 1131

A – C – D – B – A
1000 + 22 + 21 + 20 = 1063

A – D – B – C – A
50 + 21 + 60 + 1000 = 1131

A – D – C – B – A
50 + 22 + 60 + 20 = 152

A Traveling Salesperson Solution



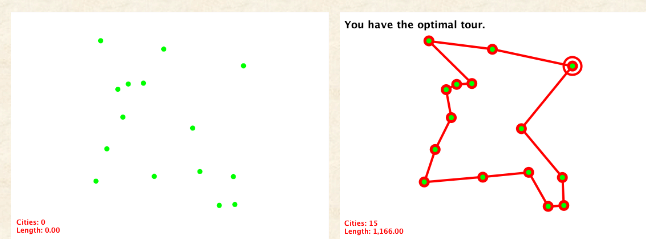
Approach
- Compute cost of every route

Worst-case
- Path connecting every city

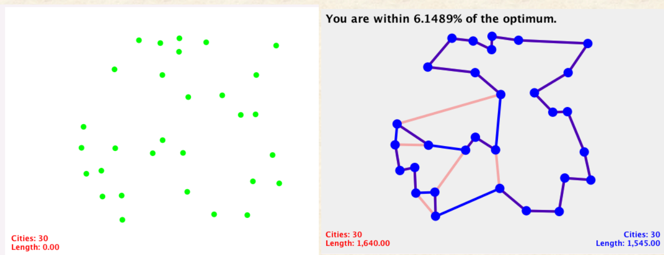
Build every route
- Pick starting city
- Pick next city (N-1 choices)
- Pick 3rd city (N-2) choices

Number of routes?
- O(N!) (N factorial)
- No polynomial solutions are known!

TSP with 15 cities



TSP with 30 cities



Try to Solve TSP Problems

http://www.tsp.gatech.edu/games/tspOnePlayer.html

Common Solution for Problems Requiring "Brilliance"

Exhaustive Search

Naïve algorithms for many "needle in a haystack" tasks involve checking all possible answers

- Combinatorial Explosion
- Exponential running time

Common in many interesting problems

Can we design smarter algorithms?

P vs NP Question

P: Problems for which solutions exist in polynomial time

- cN^k : c and k are fixed integers; N is input size
- $O(1)$, $O(\log N)$, $O(N)$, $O(N \log N)$, $O(N^2)$, $O(N^3)$
- Example: Searching, sorting, Path, Spanning Tree
- Reasonable, tractable

NP: Problems where solution can be ***checked*** in polynomial time

- Examples: Monkey Puzzle, Traveling Salesman
- Current solutions require super-polynomial-time
  - $O(2^N)$, $O(N^N)$, $O(N!)$
- Unreasonable, intractable

Question: Is P = NP?

- "Can we automate brilliance?"
- Computer scientists have not yet proved equal or not equal

NP-complete Problems

Problems in NP that are "the hardest"

- If they are in P then so is every NP problem
- All NP-complete problems essentially equivalent

How do we handle NP-Complete Problems?

1. Heuristics
   - Algorithms that produce reasonable solutions in practice
2. Approximation algorithms
   - Compute provably near-optimal solutions

Today's Summary

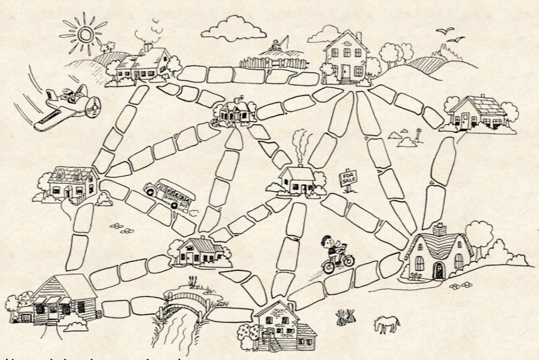
P vs NP

- P problems can be solved in polynomial time
  - Example: Minimal spanning tree uses a greedy algorithm to find shortest path connecting all nodes
- NP problems can only be checked in polynomial time
  - Unknown if polynomial-time solutions exist
  - Naïve solutions exhaustively examine all possibilities

Announcements

- Homework 8 Due Friday
- Exam Review Friday
- Exam 2 on Monday

Handout



Create the minimal spanning tree that connects all of the houses

Handout

Construct the shortest route for a Traveling Salesperson
Must visit all cities; return to starting city