Lecture 41:
What problems stretch the limits of computation?

Handout
Create the minimal spanning tree that connects all of the houses

Construct the shortest route for a Traveling Salesperson
Discussion

Is there an inherent difference between

being brilliant

and

being able to appreciate brilliance?

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”
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?

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 $O(E)$
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

Problem 3: Monkey Puzzle

Given:
- Set of N square cards with top and bottom halves of colored monkeys
- \( N = M^2 \)
- Cannot rotate cards

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

Algorithm?
Monkey Puzzle Algorithm

Try every combination of cards and see if it works

• Try every card for 1st box
• Try each of remaining cards in 2nd box
• Try each of remaining cards in 3rd box...
• Etc...

Does a greedy algorithm work?

How many combinations possible?

• $9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1$
  $= 9 \text{ Factorial} = 9!$

Analysis of Monkey Puzzle

For N cards, number of arrangements to examine is N!

Assume can analyze one arrangement in 1 microsecond

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

<table>
<thead>
<tr>
<th>N</th>
<th>Time to analyze</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>

Requires brilliance to solve quickly!
Problem 4: Travelling Salesperson (TSP)

- 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

Algorithm?

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?
- N! (factorial)
- Greedy algorithm will not work here!
Try It Yourself

30 cities is fun!

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

• Check your grades in Learn@UW
• Monday: Last real lecture (Languages other than Scratch)
• Project 3: Due Wednesday – In-class demo
• Friday: Review for Final Exam
• Sunday 10:05 – 12:05: Final Exam