INTRACTABLE PROBLEMS

CS340

in·trac·ta·ble

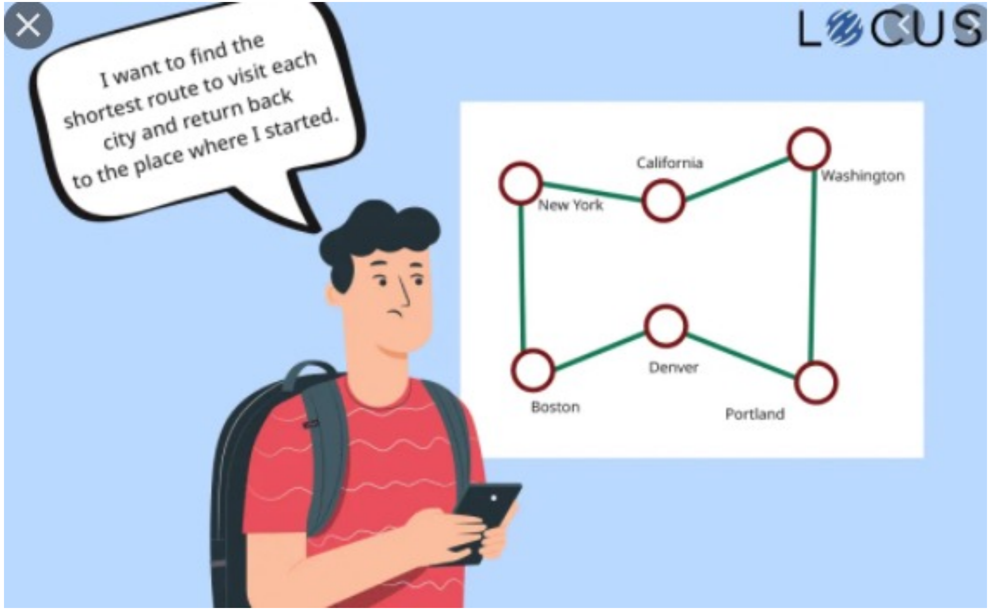
/ˌinˈtraktəb(ə)l/

adjective

hard to control or deal with.
"intractable economic problems"

Traveling Salesman

- Find the shortest tour through n cities with known positive integer distances between them.
- = find the shortest Hamiltonian circuit in a complete graph with positive integer weights

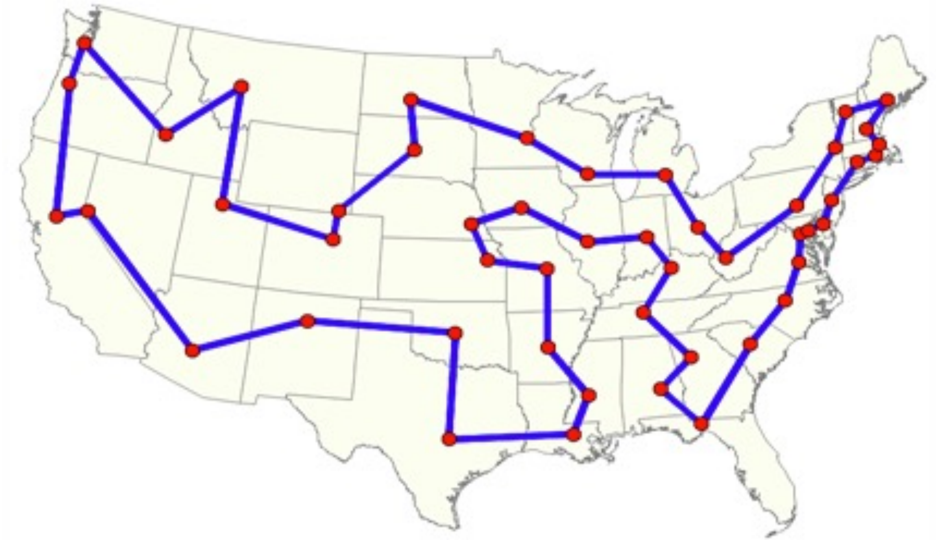


A “Hamiltonian Circuit” is just a path that visits every vertex exactly once, and ends where it begins.

Traveling Salesman

- There are n possibilities for the first city visited
- n-1 possibilities for the second city visited
- etc until the last city is visited

- What is the time complexity?



Traveling Salesman

- O(n!)
- How long does it take by brute force?
  - 10 cities can be calculated in 1 second
  - 11 cities takes ~11 seconds
  - 12 cities takes ~2 minutes
  - 13 cities takes ~26 minutes
  - 14 cities takes ~7 hours
  - 15 cities takes ~4 days
  - 16 cities takes ~2 months
  - 17 cities takes ~3 years

Tractable problems

- Solvable in a reasonable amount of time
- We define “reasonable” as polynomial time
- Polynomial time = $O(p(n))$ where $p(n)$ is a polynomial of the problem's input size, n.
- Why draw the "solvability" line at polynomial time?
  - Polynomial degree is rarely larger than 3 in solvable problems
  - Polynomial functions closed under addition and multiplication

Intractable Problems

- Cannot be solved in polynomial time

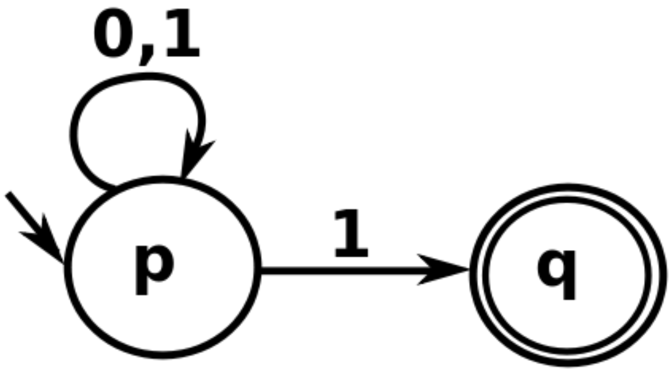
P

- Class P is
  - Decision problems
    - problems with yes/no answers
  - that can be solved in polynomial time
  - by deterministic algorithms

  - Examples?

Non-Deterministic Algorithms

- You may know of nondeterministic finite automaton



In state *p* reading a 1 can lead to *p* or to *q*.

Non-Deterministic Algorithms

1. Nondeterministic ("guessing") Stage

   An arbitrary string S is generated that can be thought of as a candidate solution to the given instance I (but may be complete gibberish as well).

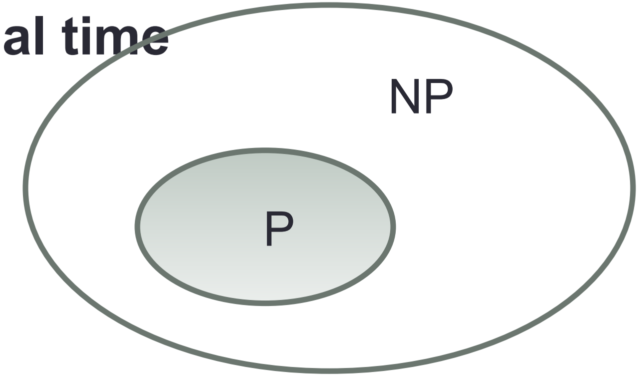
2. Deterministic ("verification") stage

   Takes both I and *S* as its input and outputs yes if *S* represents a solution to instance *I.*

   Is considered polynomial if the deterministic stage can be done in polynomial time

NP

- Class NP is
  - called ***nondeterministic polynomial***.
  - decision problems that can be solved by nondeterministic polynomial algorithms.
  - If a "certificate" containing a solution is provided, the **solution can be verified in polynomial time**
  - Most decision problems are in NP
  - P ⊆ NP



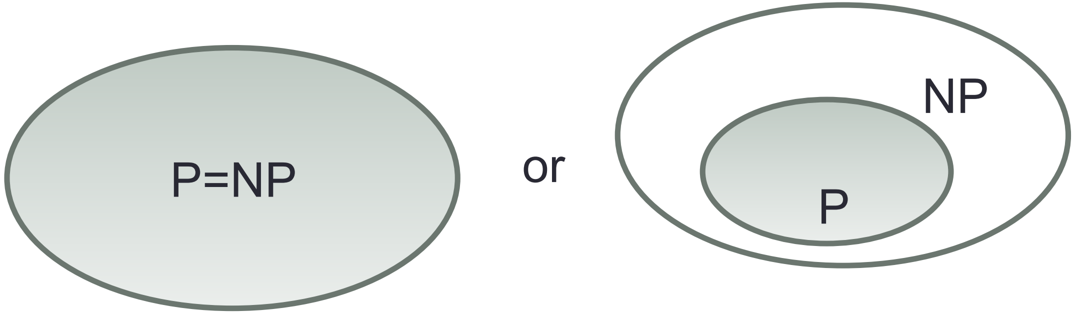
NP

- Some problems that are in NP but not known to be in P
  - Clique
  - Hamiltonian Circuit
  - Traveling Salesman
  - Knapsack / Subset Sum
  - Longest Path

  - These problems do not have known polynomial algorithms, but there is no proof that polynomial algorithms do not exist.

P vs NP

- $P \subseteq NP$ implies the possibility that $P = NP$
- This is the most important open question of theoretical computer science.
- A $1,000,000 prize awaits you
- How likely is it that P=NP?
- What would P=NP imply?



NP-Complete Problems

A decision problem *D* is said to be ***NP-complete*** if:

1. it belongs to class *NP*

2. every problem in *NP* is polynomially reducible to *D*