Intractability

CMPT 125
Mo Chen
SFU Computing Science
3/4/2020

Lecture 34

Today:

- Finite Tile Puzzles
- Exponential-Time Algorithms
- NP-Complete Problems

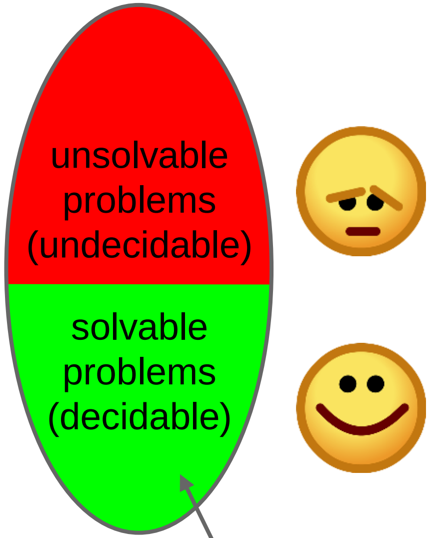
Decidability (Review)

Decision Problem:

- Answers a “Yes” / “No” question.

Some problems don’t have a solution

- called *undecidable*
- E.g., Tiling the plane
- E.g., Does program P have an infinite loop?
- E.g., Is program P correct?



Some undecidable problems are “harder” than others

- Can write algorithms that:
  - are successful on restricted classes of inputs
  - work 99% of the time
- E.g., Lab grading server

These problems might not be that easy!

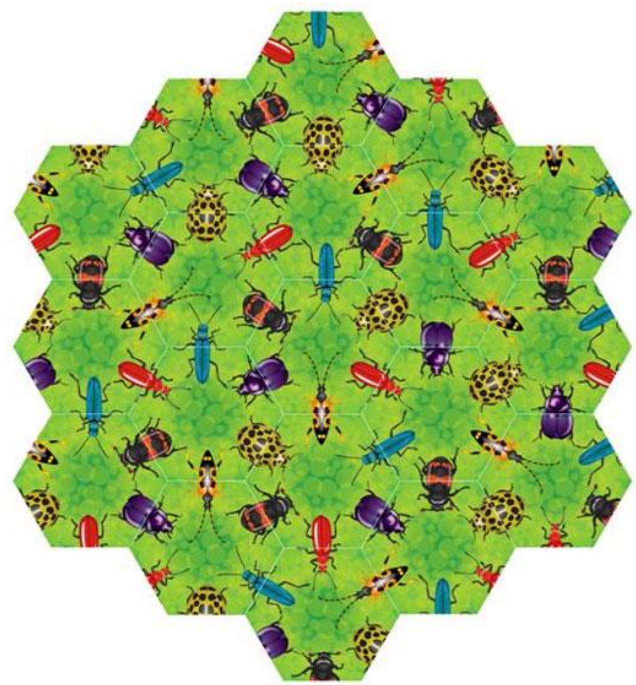
Finite Tiling Puzzles

Problem: Given $N = M^2$ tiles, can you tile an M x M grid?

- harder version: allow to rotate / mirror

A brute force approach:

- Since there are a finitely many ways of arranging the tiles, and each arrangement can easily be tested for legality, try and test all arrangements
- decidable!



What’s the running time?

- N! arrangements means $O(N!)$ time
- By the way, 9! = 362880

Human Solution to Finite Tile Puzzle

You would also use brute-force, but add one tile at a time.

- If tile doesn't fit, then try another.
- If all tries lead to failure, then remove the previous tile.

Algorithm is called *backtracking*.

- recursive
- generates partial solutions
- we didn't do 8! steps, because we rejected many permutations early

What makes a "hard" puzzle?

- many partial solutions, but . . .
- few correct solutions (usually one)

Exponential vs Polynomial Time

$N!$ is the fastest growing function yet

- faster than any polynomial

Fastest known algorithm to solve every bug puzzle is 2^N, which also grows very fast

- but remember: many puzzles can be solved quickly in practice!

Rule of Thumb: A typical computer will do 1 billion operations in around 1 second.

Q. What's the maximum practical N?

	N	$N \log N$	N^2	N^3	2^N	$N!$
1 second	10^9	4.0×10^7	3.1×10^4	10^3	30	12
10 years	3.2×10^{17}	6.0×10^{15}	5.6×10^8	6.8×10^5	58	19

Reasonable vs Unreasonable Time

The functions 2^N and $N!$ easily dwarf the growth of all functions of the form N^k, for any fixed k.

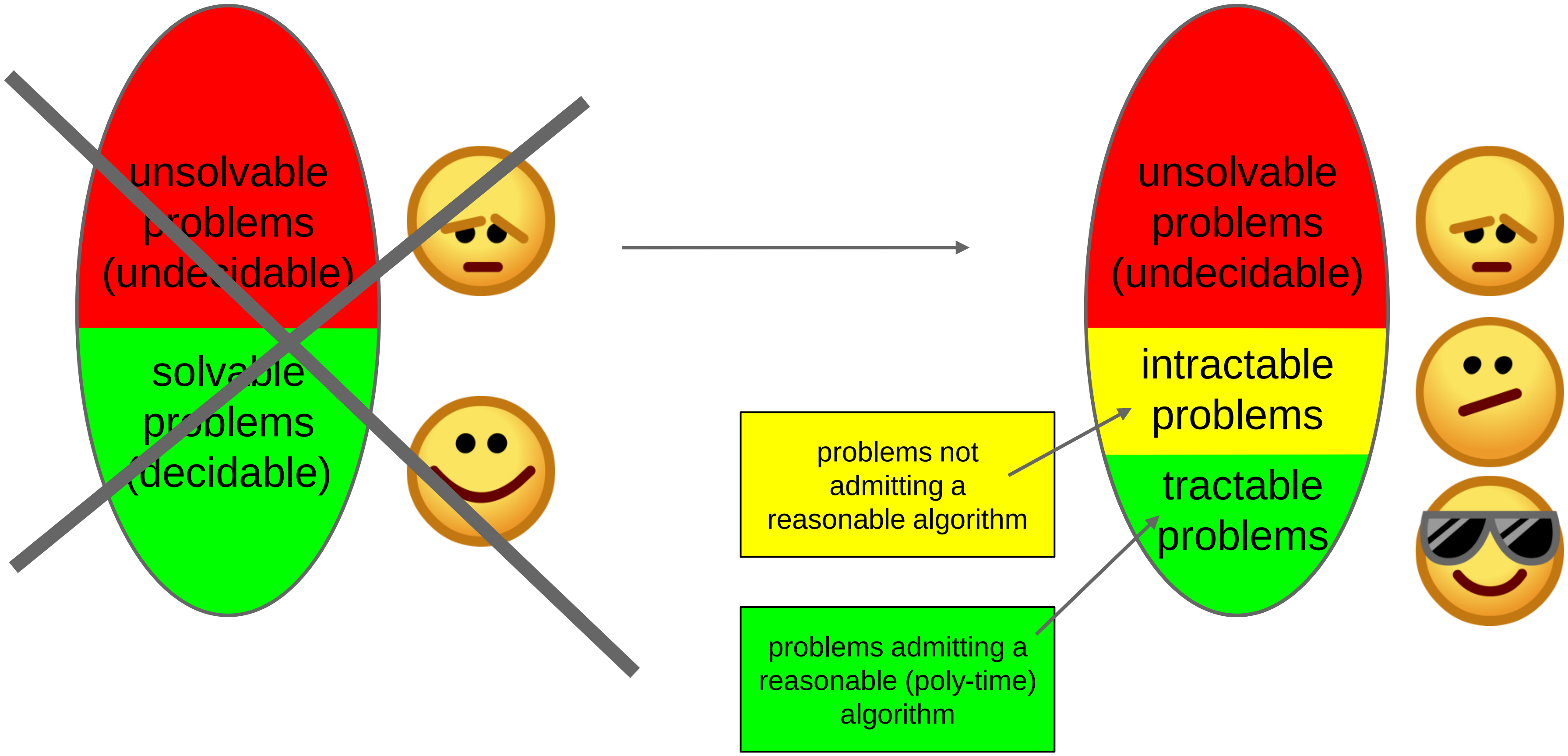
Two provisos:

- N^{1000} also grows really stinkingly fast
- There are some linear algorithms with massive leading constants

But for the most part, the distinction is valid:

- “good” is polynomial — *tractable*
- “bad” is super-polynomial — *intractable*

The World of All Problems



Intractability and NPC

Q. Is the finite tile puzzle intractable or tractable?

Mathematically speaking, it is in no man’s land.

- no one has proven that exponential time is required
  - exponential lower bounds have been proved for some problems (but not this one)
- no polynomial-time algorithm has yet been found
  - backtracking is one algorithm that works well on most bug puzzles, but not all of them.

The finite tile puzzle belongs to a class called NPC — the NP-Complete problems.

Other Problems in $\mathcal{NPC}$

There are many natural problems which are similar to the finite tile puzzle.

They come from a variety of domains.

- **The Travelling Salesman Problem (TSP)**

  Given a road network connecting N cities, plan the fastest route that passes through all N of them.

- **Subset-sum.**

  Given a list of N numbers and a target number t, find a subset of those numbers that sums to t.

More Examples

- **Knapsacking**

  Given a list of N items of weight $w_1, w_2, \ldots, w_N$ and value $v_1, v_2, \ldots, v_N$, pack a car whose maximum load is W such that value is maximized

- **Scheduling**

  Given a list of N students each of which has up to 5 final exams, devise the minimum schedule so that no exams overlap for any students

- **Satisfiability**

  Given a logical expression of length N, find a true/false substitution that will yield “true”

NPC — Rising and Falling Together

There are several hundred problems sharing remarkable properties:

- best known algorithm is exponential
- best lower bound is polynomial
- if one is intractable, then they all are, but . . .
- if one is tractable, then they all are!

$P \stackrel{?}{=} NP$ (Cook-Levin 1971)

- The most important open problem in CS
- Also considered a major open problem in Mathematics
- https://youtu.be/YX40hbAHx3s

A Use for Intractability

Sometimes the bad news can be used constructively:

- in cryptography and security

General Strategy: Devise a cryptosystem so that unauthorized decryption is expensive.

Most public-key crypto relies on large primes

- you can eavesdrop only if you can factor extremely large numbers efficiently
- integer factorization is believed to be intractable

CMPT 125 — Topics Covered

Algorithms:

- measuring performance
- the worst case → big-*O*
- software engineering principles
- brute-force paradigm
- sorting and searching
- assertions, pre/post-conditions, invariants, invariant proofs
- divide & conquer paradigm
- recursion & recursive invariants
- ADTs: stacks, queues
- linked lists, rooted trees
- binary search trees
- regular expressions & FSMs
- floating point encoding
- undecidability, intractability

Coding:

- declare all variables
- pass-by-value
- arrays are fixed length
- strings
- pointers
- coding style
- recursion
- struct
- interfaces
- ADTs
- linked data structures
- struct → class
- templates & the STL