Languages and automata (Chapters 2–4)

- A hierarchy of models of computation
- Nondeterminism
- Turing machines
- Problem set on automata (needs to be graded. . .)

Undecidability (Chapter 5)

- Definition of undecidability
- The Halting Problem
- Reduction proofs
- Problem set on undecidability proofs (due Wednesday after break)

NP-completeness (Chapters 6 and 7)

- The class $\mathcal{P}$, definition of tractability (§6.1)
- Problems: Reachability , Euler cycle, Hamiltonian cycle, Traveling Salesman, Independent Set, Clique, Node Cover, Integer Partition (§6.2)
- Boolean Satisfiability (§6.3)
- The class $\mathcal{NP}$, $\mathcal{NP}$-completeness, and proofs (§6.4)
- More problems, practice, and applications for $\mathcal{NP}$-completeness
- Problem set on $\mathcal{NP}$-completeness (Due Thurs, Dec 12)

Schedule (recent and imminent)

Date	**Reading**	**In class**
Fri, Nov 22	6 (whole chapter)	Sections 5.(4,6,7), ~~definitions from 6.(1 & 2)~~
Mon, Nov 25	Reread 6.1 Reread 6.2 through pg 282	6.1 Definition of class $\mathcal{P}$ etc 6.2 REACHABILITY, HAMCYCLE
Mon, Dec 2	Reread rest of 6.2 Reread 6.3	6.2 TSP, INDEPENDENTSET CLIQUE, PARTITION 6.3 Boolean satisfiability
Wed, Dec 4	Reread 6.4 Read 7.2	6.4 The class NP 7.1 Polynomial-time reductions

Definition 6.1.1: A Turing machine M is **polynomially bounded** if

$\exists\ p(n)$, a polynomial function such that
 $\forall\ x \in \Sigma*$
  $\forall\ C \in$ (set of configurations), either
   C is unreachable from $(s, \triangleright\underline{\sqcup}w)$, or
   $(s, \triangleright\underline{\sqcup}w) \vdash_M^k C$, where $k \leq p(|x|)$

A language is **polynomially decidable** if

$\exists\ M$, a Turing machine that decides the language, such that
 $\exists\ p(n)$, a polynomial function such that
  $\forall\ x \in \Sigma*$
   $\forall\ C \in$ (set of configurations), either
    C is unreachable from $(s, \triangleright\underline{\sqcup}w)$, or
    $(s, \triangleright\underline{\sqcup}w) \vdash_M^k C$, where $k \leq p(|x|)$

LP pg 276

Ex. 6.1.1

Proof of concatenation. Suppose $L_1, L_2 \in \mathcal{P}$. Then there exist machines M_1 and M_2 that L_1 and L_2 and are polynomially bounded by $p_1(n)$ and $p_2(n)$, respectively. Then build a machine $M*$ that takes an input w of length m and does the following:

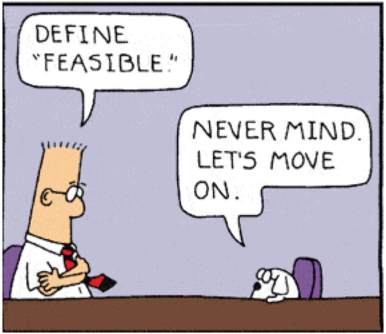
for $i = 0$ to m
    simulate $M_1(w[0..i])$ and $M_2(w[i..m])$
    if both halt y, then halt y
halt n

Suppose $r(n)$ is how long it takes to copy or restore the input. Then the number of steps is bounded by

$$m \cdot r(m) + \sum_{i=1}^{m} (p_1(i) + p_2(m - i)) \leq m \cdot (r(m) + p_1(m) + p_2(m))$$

...which is polynomial.

§6.2. The class of polynomially decidable languages is denoted $\mathcal{P}$. Why is polynomial time used as a measure of tractability/feasibility?



Scott Adams, 1994

Reachability. Given a graph G and vertices v_i and v_j, find a path from v_i to v_j.

Language version: Does there exist a path from v_i to v_j?

$$\{\kappa(G)\mathbf{b}(i)\mathbf{b}(j) \mid \exists \text{ path in } G \text{ from } v_i \text{ to } v_j\}$$

> One of the main points that will emerge from the discussion that follows is that *the precise details of encodings rarely matter*.
>
> Since it is easy to see that $m = O(n^3)$, this is yet another inconsequential inaccuracy, one that will not interfere with the issues that we deem important.
>
> LP pg 280

Euler cycle. Given a graph G, is there a closed path (cycle) that uses each edge exactly once? (Repeated vertices are okay.)

$$\{\kappa(G) \mid \exists \text{ a cycle that uses each edge exactly once}\}$$

Euler's result: A graph has an Euler cycle if all non-isolated pairs are reachable and each node's in-degree equals its out-degree.

Hamiltonian Cycle. Given a graph G, is there a cycle that passes through each vertex exactly once? (Unused edges are okay.)

$$\{\kappa(G) \mid \exists \text{ a cycle that visits each vertex exactly once}\}$$

Despite the superficial similarity between the two problems, Euler Cycle and Hamiltonian Cycle, there appears to be a world of difference between them. After one and a half centuries of scrutiny by many talented mathematicians, no one has discovered a polynomial algorithm for Hamiltonian Cycle.

LP pg 282

Traveling Salesman. Given a complete weighted graph, find a simple cycle with with least weight.

Optimization version: Given $n \in \mathbb{N}$ and an $n \times n$ distance matrix $d_{i,j}$, and letting π range over permutations of $\{1, 2, \ldots n\}$, define $c(\pi) = \left(\sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)}\right) + d_{\pi(n),\pi(1)}$ Find *pi* to minimize $c(\pi)$.

Budgeted version: Given $n \in \mathbb{N}$, an $n \times n$ distance matrix $d_{i,j}$, and $B \in \mathbb{W}$, and using π and $c(\pi)$ as above, find a permutation π such that $c(\pi) \leq B$.

Language version:

$$\{(n, d_{i,j}, B) \mid \exists\ \pi \text{ such that } c(\pi) \leq B\}$$

For next time

Reread 6.2 starting with "Optimization Problems" on pg 282.

Think about TSP carefully. In a previous semester, no one had understood TSP when they got to class—and, worse, they didn't even realize they didn't understand it.

Reread 6.3.

Do 6.3.2.