Computational Complexity

- till now: distinguished only b/twn ~~[illegible]~~ computable / non-computable problems
- no measure of "difficulty" of computation req'd to solve a given computable problem.
- intuitively: some computable problems harder than others – want to make this precise.

- Sps A is a finite alphabet
- a set of words ~~[illegible]~~ $L \subseteq A^{<\mathbb{N}}$ is called a language
  ↳ think of words as codes for objects we're interested in (e.g. finite graphs)
  ↳ think of L as general prob we want to solve, e.g. decide whether a given finite graph has a 2-coloring (decide which words are in L).

- Will again use Turing machines to model computation.

- We consider t.m.'s with exactly <u>two</u> halting states Q_y and Q_n

(recall: a t.m. is specified by instructions (Q_i, a, b, D, Q_j) where:
- Q_i is non-halting state
- a, b in the alphabet
- $D \in \{L, R\}$
- Q_j a state (possibly halting)

is one instruction for each (Q_i, a) "if in Q_i reading a write b go D and to state Q_j")

- Sps $t: \mathbb{N} \to \mathbb{N}$ is a function
- For a word $w \in A^{<\mathbb{N}}$, we write $|w|$ for length of w

<u>Def'n</u> A language L has <u>time complexity</u> t if there is a t.m. M on a finite alphabet $\Sigma \supseteq A$ s.t.

① if $w \in L$, then on input w, M halts in $\leq t(|w|)$-many steps in state Q_y

② if $w \notin L$, then on input w M halts in $\leq t(|w|)$-many steps in state Q_n.

- Say: such an M decides L
- Write TIME(t) for collection of languages that have time complexity t

Def'n P denotes collection of langs L that can be decided in polynomial time, i.e. for which there is $t: \mathbb{N} \to \mathbb{N}$ a polynomial st $L \in$ TIME(t)

- $L \in P$ means L is "tractable"
- $L \notin P$ means "intractable"

- we will sometimes modify a problem by e.g. expanding the alphabet or using multitape t.m.'s
- these changes won't ever alter complexity classes since they slow us down by at most a poly-time factor.

Reductions

↳ would like a notion of "reducing one computable problem to another"
- Spse A, B are alphabets
- a function $f: A^{<\mathbb{N}} \to B^{<\mathbb{N}}$ is a poly-time function if there

polynomial $t: \mathbb{N} \to \mathbb{N}$ and a t.m. M that on input w halts in $\leq t(|w|)$-many steps and outputs $f(w)$

(note: $|f(w)| \leq t(|w|)$, so output words are polynomial (in size of input) length)

Defn Sps $L \subseteq A^{<\mathbb{N}}$ and $K \subseteq B^{<\mathbb{N}}$ are languages. We say L is poly-time reducible to K, and write $L \leq_{PT} K$, if there is a poly-time function $f: A^{<\mathbb{N}} \to B^{<\mathbb{N}}$ s.t.

$$w \in L \iff f(w) \in K$$ ($\Rightarrow$ important direction)

idea: can then decide if $w \in L$ efficiently if we have efficient way of checking membership in K (just compute $f(w)$ and check $f(w) \in K$)

- e.g. if $K \in P$ and $L \leq_{PT} K$ then $L \in P$ too (comp'n of poly's is poly)
- conversely if checking $w \in L$ is intractable, checking $\in K$ is too.

Some languages (from a given class) are maximally complex

<u>Def'n</u> Let $\mathcal{C}$ be a class of languages. Say K is <u>$\mathcal{C}$-hard</u> if for every $L \in \mathcal{C}$ we have $L \leq_{PT} K$.
If furthermore $K \in \mathcal{C}$, say K is <u>$\mathcal{C}$-complete</u>

↳ observe that if K is $\mathcal{C}$-hard and $K \leq_{PT} K'$, then K' is $\mathcal{C}$-hard too.

<u>Problems in P</u>

- we often dispense w/ formalism and describe algorithms that apply directly to objects we care about (e.g. finite graphs, Boolean formulas, etc.)
- to translate to formal setting, would need to code the objects as finite words in a language L
  - ↳ e.g. finite graphs can be encoded by their adjacency matrix, which in turn can be encoded into finite words (need 0 and 1 for matrix entries, some third symbol for row breaks, etc.)

- leave to you to check: "natural" codings never affect time complexity in sense that implementing our intuitive algorithms formally can always be achieved w/o adding more than polynomially many steps to computation.

↳ we first consider examples of tractable problems

Problem (PATH) given a finite graph G, decide whether G is connected

- Sps has size n, i.e. $G = (V, E)$ and $|V| = n$ (formally, code word for G would be longer than n - need to encode E too - but only polynomially longer)
- we describe a poly time alg that checks connectedness by checking whether every vertex can be connected by a path to some given vertex a.

Step 0: mark a
Step k+1: mark all vertices adjacent to some marked vertex

- halt after n-steps (enough time to mark every vertex connected to u)
- check if there is any vertex v that is not marked.

Clearly poly time: - each step req's $\leq n$ marks
- run for n steps, so $\leq n^2$ time to mark all vertices connected to u
- checking for unmarked vertices another n steps at most.

$\Rightarrow$ PATH is in P ✓

Note that if we modify PATH by considering directed graphs, remains poly-time.

<u>Problem</u> (EULER) given a finite graph G, decide if G has an Euler trail (i.e. a closed walk that crosses every edge exactly once)

- Famous theorem that G has an Euler walk iff G is connected and every vertex has even degree

- can check connectedness in poly-time by above
- can count # of edges from each vertex in poly time too.

$\Rightarrow$ EULER is in P.

<u>Problem</u> (2-SAT) Given Boolean variables $x_1, \ldots, x_k$ and a set of clauses $c_1, \ldots, c_n$ on these variables each of which has the form $x_i \vee x_j$, $x_i \vee \neg x_j$, $\neg x_i \vee x_j$, $\neg x_i \vee \neg x_j$, decide if there is a truth assignment $t: \{x_1, \ldots, x_k\} \to \{T, F\}$ s.t.

$$\bigwedge_{i \leq n} c_i \quad \text{holds}$$

(i.e. decide whether $\bigwedge c_i$ is satisfiable)

↳ "2-SAT" since the c_i are 2-disjuncts
↳ view expressions $a \vee b$ as ordered, e.g. $x_i \vee \neg x_j$ distinct from $\neg x_j \vee x_i$
↳ but assume our clauses are closed under symmetry, e.g. $x_i \vee \neg x_j$ a clause iff $\neg x_j \vee x_i$ is too.

<u>Theorem</u> 2-SAT is in P

Pf:— we devise a poly-time reduction of 2-SAT to problem of finding a directed path between vertices in a graph
- define a (directed) $G = (V, E)$
 ↳ V = variables x_i and negations $\neg x_i$
 ↳ $(x, y) \in E$ iff there is a clause c_j of form $\neg x \vee y$ (using $\neg\neg x = x$)
  e.g. if $x_1 \vee x_2$ a clause then $(\neg x_1, x_2) \in E$

"clearly": constructing G from the x_i, c_j is poly in # of clauses n.
 (to make precise: would need explicit codings of 2-SAT problems and directed graphs, then a poly-time translation)

Claim $\bigwedge_{i \le n} c_i$ is not satisfiable iff there are directed paths from x_i to $\neg x_i$ and from $\neg x_i$ to x_i in G

Pf. ($\Leftarrow$)—Sps there are such paths but $\bigwedge c_i$ is satisfiable; fix a truth assignment
- notice: if $(x, y) \in E$ (i.e. $\neg x \vee y$ is a clause), then if x is assigned T