NP- Completeness

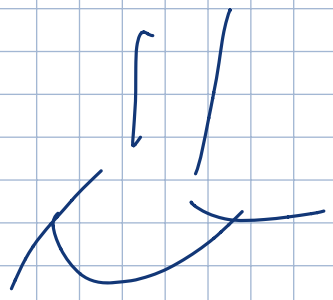
Problems vs. problem instances

I/O tape ... 0 0 1 0 1 ...

Gödelization

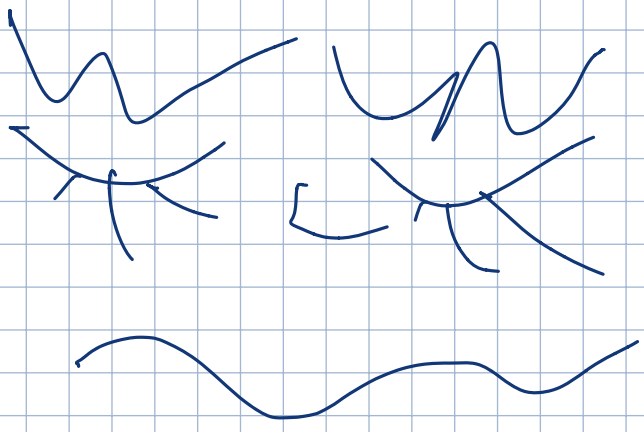
$\Sigma = \{0, 1\}$

$L = \{0, 1, 01, 10, 00, 11, 000, \dots\}$



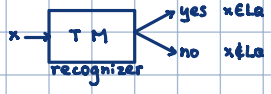
Every problem now can be imagined as a subset of strings $\{0,1\}^*$

• Does a graph have a path of at most k-edges?



Q: Does a graph have a path of length k?

$L_Q = \{x \in \Sigma^* : Q(x) = 1\}$



Q is a decision problem

Shortest paths: what is the SP between those two vertices?

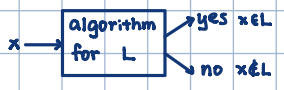
Observation: If general problem is "easy", then the decision version is also "easy"

If the decision is "hard" then the general problem will be at least as "hard"

From now on, we will be dealing ONLY with decision problems

Complexity Class P

$P = \{L \in \{0,1\}^* :$ there exists algorithm A that decides L in polynomial time$\}$



Every algorithm we did so far belongs to class P! $O(n^2), O(n\log n), O(VE^2) \dots$

Theorem:

P = {languages that can be accepted in poly-time}

Proof:

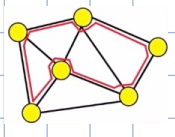
If accepted in poly time, that means machine A if it stops it takes cn^k steps.

Simply create machine A' the simulates A for cn^k steps and halts with acceptance if A halts, otherwise rejects the strings. A' decides the language in poly-time

Hamilton Cycle

HAM-CYCLE: In an undirected graph find a simple cycle (not repeating a vertex) that traverses all vertices.

Commentary: Elegant & simple problem that if we can solve we can address ... computational biology problems.



Today the only exact solutions we have are exponential (i.e: enumerate every cycle)

Verification: Algorithm A verifies a problem if given instance $x \in L$ of the problem there exists a certificate (or witness, basically a "candidate solution") st. $A(x,y) = 1$. The language verified is:

$L = \{x \in \{0,1\}^* : \exists y \in \{0,1\}^* \text{ st. } A(x,y) = 1\}$

Ham cycle: hard to solve but what about if you are to verify a candidate solution?

To verify it, this takes poly-time!

Informally definition: It is the class of problems that have a poly-time verification algorithm

Formally definition: Language (problem) L belongs to complexity class NP if $\exists$ poly-time algorithm A and a constant c st L contains the string:

$L = \{x \in \{0,1\}^* : \exists \text{ certificate } y, |y| = O(|x|^c) \text{ st } A(x,y) = 1\}$

Theorm: $P \subseteq NP$

Why? If $L \in P$ that means $\exists$ algorithm to decide (solve) L in poly time. Hence, ignore the certificate and solve it in poly-time

Complexity class co-NP

$L \in NP \Rightarrow \bar{L} \in$ co-NP (complement NP)

Example co-ham-cycle

$x \in \bar{L}$ is not a graph, or if it is, it doesn't have a ham cycle

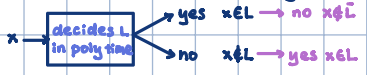
$\{0,1\}^*$

graph encodings with no ham cycles

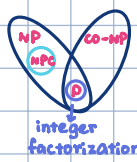
encodings do not represent graph } x

THM: Class P is closed under complement that is $L \in P \Rightarrow \bar{L} \in P$

Proof: If $L \in P$ then $\exists$ TM A deciding (solve) L in poly-time



Relation between complexity classes



Open questions:

$NP \stackrel{?}{=}$ co-NP

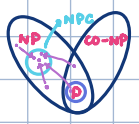
$P \stackrel{?}{=} NP \cap$ co-NP

$NP \stackrel{?}{=} P$

Complexity class NP-complete

Language $L \in NPC$ iff

- $L \in NP$ (verified in poly-time)
- $\forall L' \in NP, L' \leq_p L$ (NP-hardness)



Practice Theorems

THM: if $\exists L \in NPC$ and $L \in P \Rightarrow P = NP$

THM: NP = co-NP iff $\exists L \in NPC$ st $\bar{L} \in NP$

$\Rightarrow$ Straightforward

$\Leftarrow$ Let $L \in NPC$ and $\bar{L} \in NP$. Pick any $L' \in NP$ (where $\bar{L'} \in$ co-NP)

Since $L' \leq_p L$ this implies that $\bar{L'} \leq_p \bar{L}$ which implies that $\bar{L'} \in NP$

Methodology to prove NPC

Given language L to show $\in NPC$:

a). Prove that a solution can be verified in poly-time

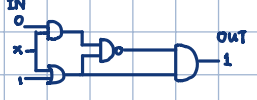
b). Select known $L' \in NPC$

b.1. Describe algorithm f(), that given instance $x \in L'$ it "translates" it to f(x) s.t. $x \in L' \leq_p f(x) \in L$

b.2. Show f() takes poly time to compute

Example Circuit-SAT (Cook 1971)

Given a AND, OR, NOT digital circuit with a single output find an input vector that makes the output 1.



$\forall NP$ problem $\leq_p$ Circuit SAT

First NPC problem: