Lecture 19

The class NP

Definition:

$NP = \bigcup_{k \geq 1} \text{Nondeterministic-TIME}(n^k)$

I.e., the set of (decision) problems solvable by computers in *Nondeterministic* polynomial time. I.e., $L \in NP$ iff there is a nondeterministic algorithm deciding L in time $T(n) = O(n^k)$ for some fixed k (i.e., k is independent of the input).

Alternate Views of Nondeterminism

NTM – there is a path…

Parallel – make the tree

Search – look for a path (or sat-ing assignment or clique or…)

Guess and Check

Polynomial Verifier

Alternate Way To Define NP

A language L is *polynomially verifiable* iff there is a polynomial time procedure v(-,-), (the "verifier") and an integer k such that

for every $x \in L$ there is a "hint" h with $|h| \le |x|^k$ such that v(x,h) = YES and

for every $x \notin L$ there is *no* hint h with $|h| \le |x|^k$ such that v(x,h) = YES

("Hints," sometimes called "certificates," or "witnesses", are just strings.)

Equivalently:

There is some integer k and language L_v in P s.t.:

$$L = \{ x \mid \exists y, |y| \le |x|^k \wedge \langle x,y \rangle \in L_v \}$$

Example: Clique

"Is there a k-clique in this graph?"

any subset of k vertices *might* be a clique

there are *many* such subsets, but I only need to find one

if I knew where it was, I could describe it succinctly, e.g. "look at vertices 2,3,17,42,...",

I'd know one if I saw one: "yes, there are edges between 2 & 3, 2 & 17,... so it's a k-clique"

this can be quickly checked

And if there is *not* a k-clique, I wouldn't be fooled by a statement like "look at vertices 2,3,17,42,..."

More Formally: CLIQUE is in NP

procedure v(x,h)

if

x is a well-formed representation of a graph
G = (V, E) and an integer k,

and

h is a well-formed representation of a k-vertex
subset U of V,

and

U is a clique in G,

then output "YES"

else output "I'm unconvinced"

Is it correct?

For every x = (G,k) such that G contains a k-clique, there is a hint h that will cause v(x,h) to say YES, namely h = a list of the vertices in such a k-clique

and

No hint can fool v into saying yes if either x isn't well-formed (the uninteresting case) or if x = (G,k) but G does not have any cliques of size k (the interesting case)

The 2 defns are equivalent

Theorem: L in NP iff L is polynomially verifiable

Pf: ⇒ Let M be a poly time NTM for L, x an input to M, |x| = n. If x in L there is an accepting computation history y for M on x. If M runs $T = n^{O(1)}$ steps on x, then y is T+1 configs, each of length ~T, so $|y| = O(T^2) = n^{O(1)}$. Furthermore, a *deterministic* TM can check that y is an accepting history of M on x in poly time. Critically, if x is *not* accepted, no y will pass this check. Thus, L is poly time verifiable.

(We could equally well let y encode the sequence of nondeterministic choices M makes along some accepting path.)

The 2 defns are equivalent (cont.)

Theorem: L in NP iff L is polynomially verifiable

Pf: $\Leftarrow$ Suppose L is poly time verifiable, V is a time n^d -time TM implementing the verifier, and k is the exponent in the hint length bound. Consider this TM:

M: on input x, nondeterministically choose a string y of length at most $\|x\|^k$, then run V on $\langle x,y \rangle$; accept iff it does.

Then M is an NTM accepting L: By defn of poly verifier $x \in L$ iff $\exists y, |y| \le |x|^k \wedge$ V accepts $\langle x,y \rangle$, and M tries (nondeterministically) all such y's, accepting iff it finds one that V accepts.

Time bound for M: $(|x|+|x|^k+3)^d = O(n^{kd}) = n^{O(1)}$

Example: SAT

"Is there a satisfying assignment for this Boolean formula?"

- any assignment might work
- there are lots of them
- I only need one
- if I had one I could describe it succinctly, e.g., "x_1=T, x_2=F, ..., x_n=T"
- I'd know one if I saw one: "yes, plugging that in, I see formula = T..." this can be quickly checked
- And if the formula is unsatisfiable, I wouldn't be fooled by , "x_1=T, x_2=F, ..., x_n=F"

More Formally: SAT $\in$ NP

Hint: the satisfying assignment A

Verifier: v(F,A) = syntax(F,A) && satisfies(F,A)

- Syntax: True iff F is a well-formed formula & A is a truth-assignment to its variables
- Satisfies: plug A into F and evaluate

Correctness:

- If F is satisfiable, it has some satisfying assignment A, and we'll recognize it
- If F is unsatisfiable, it doesn't, and we won't be fooled

Alternate Views of Nondeterminism

NTM – there is a path…

Parallel – make the tree

Search – look for a path (or sat-ing assignment or clique or…)

Guess and Check

Polynomial Verifier

The complexity class NP

NP consists of all decision problems where

You can verify the YES answers efficiently (in polynomial time) given a short (polynomial-size) hint

one among exponentially many; know it when you see it

And

No hint can fool your polynomial time verifier into saying YES for a NO instance

(implausible for all exponential time problems)

Keys to showing that a problem is in NP

What's the output? (must be YES/NO)

What's the input? Which are YES?

For every given YES input, is there a hint that would help? Is it polynomial length?

OK if some inputs need no hint

For any given NO input, is there a hint that would trick you?

FALSE Example

A_{TM} is in NP

Input: a pair <M,w>

Output: yes/no does M accept w

Hint: y, an accepting computation history of M on w

Clearly, such a y exists for all accepted x and only accepted x, so we accept the right x's and reject the rest.

And it's fast – checking successive configs in the history is at worst, quadratic in the length of the history, so the verifier for <x,y> runs in time $|<x,y>|^{O(1)}$.

3' UTR