Harvard CS 121 and CSCI E-207

Lecture 21: Nondeterministic Polynomial Time

Salil Vadhan

(lecture given by Colin Jia Zheng)

November 15, 2012

- Reading: Sipser §7.3.

"Nondeterministic Time"

We say that a nondeterministic TM M decides a language L iff for every $w \in \Sigma^*$,

1. Every computation by M on input w halts (in state q_{accept} or state q_{reject});

2. $w \in L$ iff there exists at least one accepting computation by M on w.

3. $w \notin L$ iff every computation by M on w rejects (or dies, with no applicable transitions).

M decides L in nondeterministic time $t(\cdot)$ iff for every w, every computation by M on w takes at most $t(|w|)$ steps.

More on Nondeterministic Time

1. Linear speedup holds.

2. "Polynomial equivalence" holds among nondeterministic models

   e.g. L decided in time T by a nondeterministic multitape TM

   $\Rightarrow L$ decided in time $O(T^2)$ by a nondeterministic 1-tape TM

Definition:

NTIME$(t(n)) =$
$\{L : L$ is decided in time $t(n)$ by some nondet. multitape TM$\}$

NP = $\bigcup_{\text{polynomial } p}$ NTIME$(p) = \bigcup_{k \geq 0}$ NTIME(n^k).

P vs. NP

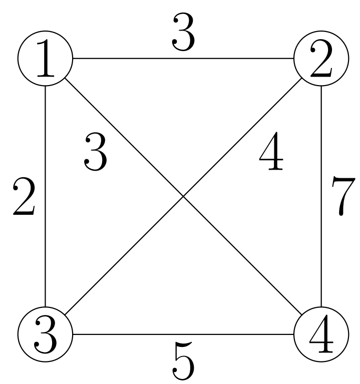
- Clearly P $\subseteq$ NP. But there are problems in NP that are not obviously in P ($\neq$ "obviously not")

- TSP = TRAVELLING SALESMAN PROBLEM.
  - Let $m > 0$ be the number of cities,
  - $D : \{1, \ldots, m\}^2 \to \mathcal{N}$ give the distance $D(i, j)$ between city i and city j, and
  - B be a distance bound

Then TSP $=$

$$\{\langle m, D, B \rangle : \exists \text{ tour of all cities of length} \leq B\}.$$

Traveling Salesman Problem: Example



$n = 4$

$B \geq 15 \Rightarrow \langle m, D, B\rangle \in \mathrm{TSP}$
$B \leq 14 \Rightarrow \langle m, D, B\rangle \notin \mathrm{TSP}$

“tour” = visits every city and returns to starting point

There are many variants of TSP, eg require visiting every city exactly once, triangle inequality on distances...

TSP$\in$ NP

- Why is TSP $\in$ NP?

  Because if $\langle m, D, B\rangle \in$ TSP, the following nondeterministic strategy will accept in time $O(n^3)$, where $n =$ length of representation of $\langle m, D, B\rangle$.

  – nondeterministically write down a sequence of cities $c_1, \ldots, c_t$, for $t \leq m^2$. ("guess")

  – trace through that tour and verify that all cities are visited and the length is $\leq B$. If so, halt in q_{accept}. If not, halt in q_{reject}. (and "check")

  If $\langle m, D, B\rangle \notin$ TSP, above has no accepting computations.

  But any obvious deterministic version of this algorithm takes exponential time.

A useful characterization of NP

- A verifier for a language L is an algorithm V such that

$$L = \{x : V \text{ accepts } \langle x, y\rangle \text{ for some string } y\}.$$

- A polynomial-time verifier is one that runs in time polynomial in $|x|$ on input $\langle x, y\rangle$.
- A string y that makes $V(\langle x, y\rangle)$ accept is a "proof" or "certificate" that $x \in L$.
- **Example:** TSP

  certificate y = ?

  $V(\langle x, y\rangle)$ = ?

- Without loss of generality, $|y|$ is at most polynomial in $|x|$.

NP is the class of easily verified languages

- **Theorem:** NP equals the class of languages with polynomial-time verifiers.

  Proof:

$\Rightarrow$

$\Leftarrow$

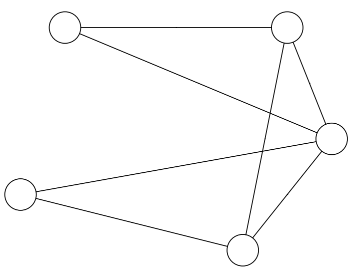
- "L is in NP iff members of L have short, efficiently verifiable certificates"

More problems in NP

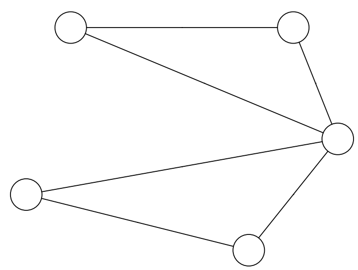
- HAMILTONIAN CIRCUIT

  HC $= \{G : G$ is an undirected graph with a circuit that touches each node exactly once$\}$.

HC



No HC



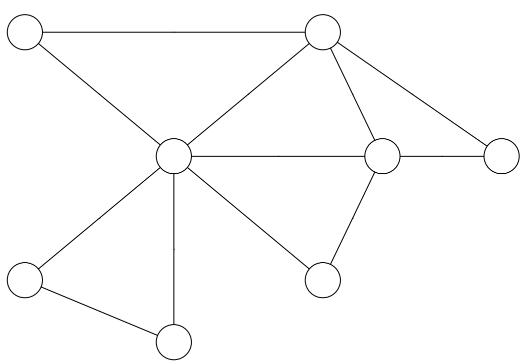
  Really just a special case of TSP. (why?)

- We are not fussy about the precise method of representing a graph as a string, because all reasonable methods are within a polynomial of each other in length.

A "similar" problem that is in P

- EULERIAN CIRCUIT

  EC $= \{G : G$ is an undirected graph with a circuit that passes through each edge exactly once$\}$.



It is easy to check if G is Eulerian...

So EC $\in$ P.

Composite Numbers

- COMPOSITES $= \{w : w$ a composite number in binary $\}$.

  COMPOSITES $\in$ NP

  Not obviously in P, since an exhaustive search for factors can take time proportional to the value of w, which grows as $2^n =$ exponential in the size of w.

  Only recently (2002), it was shown that COMPOSITES $\in$ P (equivalently, PRIMES $\in$ P).

Boolean logic

Boolean formulas

Def: A Boolean formula (B.F.) is either:

· a "Boolean variable" $x, y, z, \ldots$

· $(\alpha \vee \beta)$ where α, β are B.F.'s.

· $(\alpha \wedge \beta)$ where α, β are B.F.'s.

· $\neg\alpha$ where α is a B.F.

e.g. $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$

[Omitting redundant parentheses]

Boolean satisfiability

Def: A truth assignment is a mapping
a : Boolean variables $\rightarrow \{0, 1\}$. [$0 =$ false, $1 =$ true]

The $\{0, 1\}$ value of a B.F. γ on a truth assignment a is given by the usual rules of logic:

- If γ is a variable x, then $\gamma(a) = a(x)$.
- If $\gamma = (\alpha \vee \beta)$, then $\gamma(a) = 1$ iff $\alpha(a) = 1$ or $\beta(a) = 1$.
- If $\gamma = (\alpha \wedge \beta)$, then $\gamma(a) = 1$ iff $\alpha(a) = 1$ and $\beta(a) = 1$.
- If $\gamma = \neg\alpha$, then $\gamma(a) = 1$ iff $\alpha(a) = 0$.

a satisfies γ (sometimes written $a \models \gamma$) iff $\gamma(a) = 1$.

In this case, γ is satisfiable. If no a satisfies γ, then γ is unsatisfiable.

Boolean Satisfiability

SAT $= \{\alpha : \alpha$ is a satisfiable Boolean formula$\}$.

Prop: SAT $\in$ NP

A "similar" problem in P: 2-SAT

A 2-CNF formula is one that looks like

$$(x \vee y) \wedge (\neg y \vee z) \wedge (\neg y \vee \neg x)$$

i.e., a conjunction of clauses, each of which is the disjunction of 2 literals (or 1 literal, since $(x) \equiv (x \vee x)$)

2-SAT $=$ the set of satisfiable 2-CNF formulas.

e.g. $(x \vee y) \wedge (\neg x \vee \neg y) \wedge (\neg x \vee y) \wedge (x \vee \neg y) \notin$ SAT

2-SAT $\in$ P

Method (resolution):

1. If x and $\neg x$ are both clauses, then not satisfiable

   e.g. $(x) \wedge (z \vee y) \wedge (\neg x)$

2. If $(x \vee y) \wedge (\neg y \vee z)$ are both clauses, add clause $(x \vee z)$ (which is implied).

3. Repeat. If no contradiction emerges $\Rightarrow$ satisfiable.

$O(n^2)$ repetitions of step 2 since only 2 literals/clause.

Proof of correctness: omitted