CBC 5th, complexity of computing

NP- completeness

→ Allmost all algorithms considered so far in all the lectures (till now 22/11/06) run in worst-case polynomial time

$T(n) = O(n^k)$ for some constant k

n = input size

→ {P class} -:

- The class of algorithms that run in polynomial time is called P

→ A problem Q is a binary relation on a set I of instances and a set S of solution

Example -: shortest path problem

instance -: graph G
vertices U and V

solution -: sequence of vertices
(shortest path)

Decision problems :-

→ A decision problem is a Question that has two possible answers yes/no. The Question is about some input

→ statement of decision problem.

- instance discription
  [defines the information expected in the input]
- Question part
  [states the actual yes/no Question contains variables defined in the instance discription]

Ex: instance : an undirected graph. $G = (V, E)$

Question : Does G contain a clique. of K vertices.

instance : an undirected graph. $G = (V, E)$ and an integer k

Question : Does G contain a clique. of k vertices

Decision problems and algorithms

An algorithm A accepts a string $x \in \{0,1\}^*$ if, given input string x the algorithm outputs $A(x) = 1$.

→ The language accepted by an algorithm A is the set
$L = \{x \in \{0,1\}^* \mid A(x) = 1\}$

→ An algorithm A rejects a string x if $A(x) = 0$

→ A language L is decided by an algorithm A, if every binary string is either accepted or rejected by an algorithm

Ex:- The language PATH is decided by the following algo. in polynomial time
use Bellman-Ford to find shortest path from u to v in G.
if length(path) ≤ k
Then output 1
else output 0

Decision problems and algorithms

A complexity class is a set of languages, membership in which is determined by a complexity measure (e.g. running time) on an algorithm that determines whether a given string belongs to a language }.

Example:

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

Note

Polynomial time verification

Given a problem instance and a solution (certificate), verify that the solution solves the problem

Example:

Given $\langle G, U, V, k \rangle$, path p

verify length$(p) \leq k$

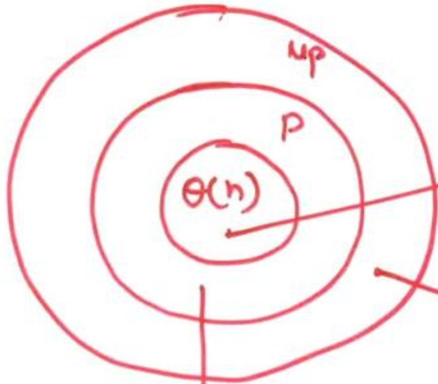
NP

The complexity class NP is the class of languages that can be verified by a polynomial time algorithm

$L \in NP$ if algorithm A verifies language L in polynomial time

Reducibility

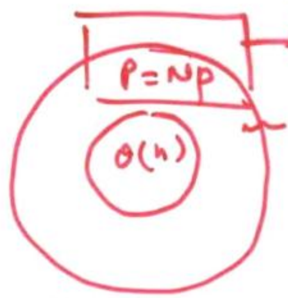
A problem Q can be reduced to another problem Q'. If any instance of Q can be "easily rephrased" as an instance of Q'. whose solution provides a solution to the instances of Q



find Best : θ(n)
{ No. of problems = Infinity }

NP but not in P.
{ No. of problems = either 0 or Infinity }.
depends

sorting θ(n lg n).
{ No. of problems = Infinity }



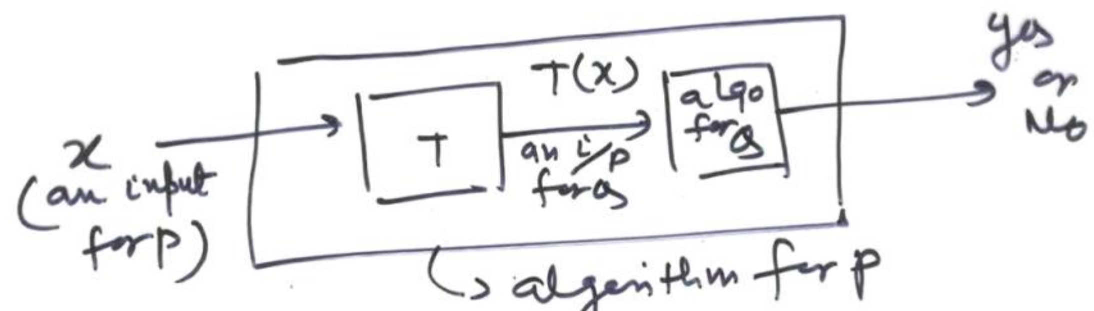
Open Question

A language L_1 is poly-time reducible to language L_2, written $L \leq_p L_2$ if $\exists$ a poly-time computable function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ such that $\forall\, x \in \{0,1\}^*$

$$\boxed{x \in L_1 \text{ iff } f(x) \in L_2}$$

reduction fn.

Note (It's a one way function Q' will not always reduce to Q)



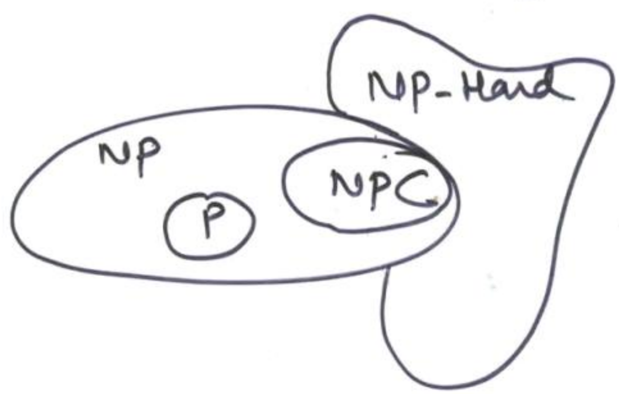
NP Completeness

NP complete problems are the hardest problems in NP i.e every problem in NP reduces to an NP complete problem

→ a language $L \subseteq \{0,1\}^*$ is NPC if $L \in NP$, and $L' \leq_p L$ for $\forall L' \in NP$

→ ~~The~~ a language $L \subseteq \{0,1\}^*$ is NP hard if $L' \leq_p L$ for every $L' \in NP$.

→ A language that is NP hard is not necessarily in NP.



Strategy for proving $L \in NPC$

Step 1 :- Prove $L \in NP$ (poly-time verifiable)

Step 2 :- Select $L' \in NPC$

Step 3 :- Describe a poly-time algorithm computing a function f that maps instances of L' to instance of L

Step 4 :- Prove that $x \in L'$ iff $f(x) \in L \ \forall x \in \{0,1\}^*$

$L' \leq_p L$.

Circuit satisfiability

and gate   OR Gate   Not gate

Not satisfiable for any $x_1 \in \{0,1\}$

satisfiable for $x_1 x_2 =$ 01, 10, or 11 thus satisfiable

Given a boolean combination circuit composed of AND, OR and NOT gate is it satisfiable?

$$\text{CIRCUIT-SAT} = \{ \langle C \rangle \mid C \text{ is satisfiable boolean combinational circuit} \}$$

~~<C> is a binary string encodi~~

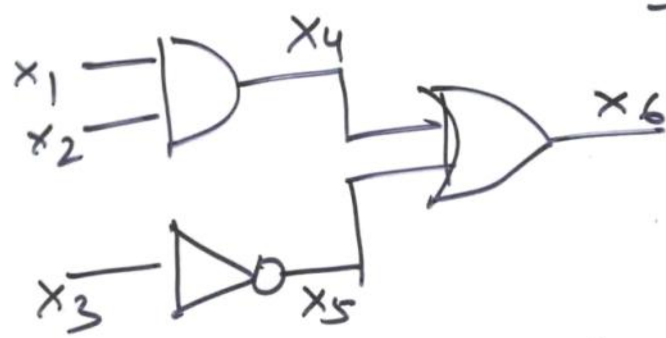
Determining membership in CIRCUIT-SAT would require checking of 2^k possible binary assignment to the K input of a circuit

CIRCUIT SAT $\notin P$

9

Reducibility

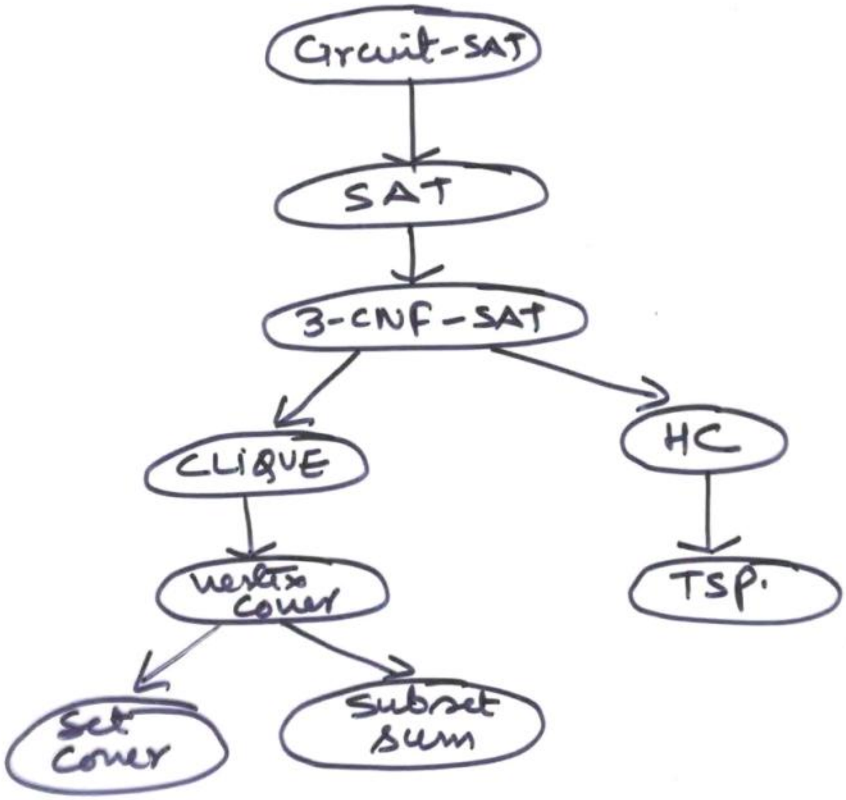
Circuit



ϕ — formula

$$\phi = x_6 \wedge (x_4 \Leftrightarrow (x_1 \wedge x_2)) \wedge (x_5 \Leftrightarrow \neg x_3) \wedge (x_6 \Leftrightarrow (x_4 \vee x_5))$$

→ constructing this formula takes polynomial time


Circuit-SAT
SAT
3-CNF-SAT
CLIQUE
HC
vertex cover
TSP.
Set cover
subset sum

Cook's theorem

→ Cook's theorem shows that the satisfiability problem is NP-complete

→ Without loss of generality, we assume that languages in NP are over the alphabet $\{0,1\}^*$

Lemma-1 : if $L \in NP$, then L is accepted by a 1-Tape NTM N with alphabet $\{0,1\}$ such that for some polynomial $p(n)$, the following properties hold

* N's computation is composed of Two phases, the guessing phase and checking phase
* In the guessing phase, N Nondeterministically writes a string $\sqcup y$ directly after the input string and in the checking phase, N behaves deterministically
* N uses at most $p(n)$ tape cells, never moves it's head to the left of w, and takes exactly $p(n)$ steps in the checking phase

(12)

Theorem -: CNFSAT is NP-Complete

Proof -: To prove that CNFSAT is NP-Complete, we show that for any language $L \in NP$,

$$L \leq_p CNFSAT.$$

Note → Let $L \in NP$ and let N be a NTM accepting L that satisfied the properties earlier mentioned.

→ Transition fn = δ.

- states of $N = q_0 \ldots q_r$
- $s_0 = 0$, $s_1 = 1$, $s_2 = \sqcup$

Note → on input w of length n, how to construct a formula in CNF form f_w, which is satisfiable iff w is accepted by N.

→ The variables of f_{15} are

Variable	Range	Meaning
$Q[i,k]$	$0 \le i \le p(n)$ $0 \le k \le r$	At step i of the checking phase, the state of N is q_k
$H[i,j]$	$0 \le i \le p(n)$ $0 \le j \le p(n)$	At step i of the checking phase, the head of N is at ~~a~~ the tape square j
$S[i,j,l]$	$0 \le i \le p(n)$ $0 \le j \le p(n)$ $0 \le l \le 2$	At step i of the checking phase, the symbol in square j is s_l

→ **Goal** :- To construct f_w so that it's satisfied only by assignments to the variables that corresponds to accepting computations of N on w.

→ The clauses of f_w are constructed to ensure that the following conditions are satisfied

Note

1) At each step i of the checking phase, N is in exactly 1 state $O(p(n))$

$Q[i,0] \vee Q[i,1] \vee \cdots \vee Q[i,r]$

Note Need to ensure that N is not both in state q_j and $q_{j'}$. $O(p(n))$

$\overline{Q[i,j]} \vee \overline{Q[i,j']}$

2) At each step i, the head is on exactly one tape square $O(p(n)^2) + O(p(n)^3)$

3) At each step i, there is exactly 1 symbol in each tape square $O(p(n)^2) + O(p(n)^2)$

4) At step 0 of the checking, the state is the initial state $O(1)$

5) At step $p(n)$ of the checking phase, N is in accepting state $O(1)$

6: The configuration of N at the $(i+1)^{st}$ step follow from that at the i^{th} step, by applying the transition function

→ If at step i, the tape head head of N is pointing to j^{th} tape cell, N is in q_k, s_l is the symbol under the tape head, $(q_k, s_l, q_{k'}, s_{l'}, X) \in \delta$
$X \in \{L, R\}$

$\overline{Q[i,k]} \vee \overline{H[i,j]} \vee \overline{S[i,j,l]} \vee Q[i+1,k']$

→ Then at step $i+1$, the tape head is pointing to $(j+y)^{th}$ tape cell, where $y = -1$ if $X = L$, $y = 1$ if $X = R$ and the symbol N is in the state $q_{k'}$ and the symbol in the cell j is $s_{l'}$

$O(p(n^2))$

$\overline{Q[i,k]} \vee \overline{H[i,j]} \vee \overline{S[i,j,l]} \vee H[i+1, j+y]$

$\overline{Q[i,k]} \vee \overline{H[i,j]} \vee \overline{S[i,j,l]} \vee S[i+1, j, l']$

→ All of the clauses for condition 1 to 6 can be computed in polynomial time

Final → w is accepted by N iff f_w is satisfiable

16