RADHIKA PATWARI

18CS10062

5/10/2021

Class Test-2

2\. (a) A language $L \subseteq \Sigma^*$ is **hard** for a class C of recursive enumerable languages under polynomial time many-one reductions if and only if every $L' \in C$, $L' \leq_m L$ is satisfied.

If $L \in C$, then L is said to be **complete** for class C.

(b) Halting problem is a re-complete language

$HP = \{ M\#w \mid M \text{ halts on input } w \}$

PAGE NO
DATE / /

2.(c) let $\overline{FIN} = \{ \langle M \rangle \mid L(M) \text{ is infinite} \}$

Now we can show $L_\epsilon \le \overline{FIN}$

$(M, x) \rightarrow N$

N: on input y,
1) if $|y| \le |x|$, accept y
2) else, ~~[illegible]~~ run M on x, if M accepts x, accept y

$$L(N) = \begin{cases} S & \text{if } S \in \Sigma^*, |S| \le |x|, \text{ M doesn't halt on } x \\ \Sigma^* & \text{if M accepts } x \end{cases}$$

Hence $\overline{FIN}$ is re-hard

~~[illegible]~~ (FIN & $\overline{FIN}$) $\notin$ re set.

18CS10062

PAGE NO
DATE / /

1. $S = \{ 0^{2^i + 3^i} \mid i \geq 1 \}$

$$\left.\begin{array}{l} S \rightarrow ACOB \\ CO \rightarrow OOC \\ CB \rightarrow DB \\ OD \rightarrow DO \\ AD \rightarrow AC \end{array}\right\} \text{for } 2^i$$

$$\left.\begin{array}{l} CB \rightarrow A'C'O'B' \\ C'O' \rightarrow O'O'O'C' \\ C'B' \rightarrow D'B' \\ O'D' \rightarrow D'O' \\ A'D' \rightarrow A'C' \\ C'B' \rightarrow E' \\ O'E' \rightarrow E'O' \\ A'E' \rightarrow \varepsilon \end{array}\right\} \text{for } 3^i$$

4. Using a special case of NonTrivial SAT,

4-SAT → given a 4-CNF formula ϕ, it is satisfiable if each clause contain atleast 1 literal = True and atleast 1 literal = false.

If we can prove 4-SAT is NP-complete, then since it is special case of NonTrivial SAT, we can show
4-SAT $\leq_m$ NonTrivial SAT

18CS10062

4-SAT is NP-complete

4-SAT is in NP as there exists a certificate that can be verified in polynomial time.

Certificate → an assignment of variable such that each clause contain atleast 1 literal = true and atleast 1 literal = false.

4SAT is in NP-Hard → Using reduction from 3-SAT

3-SAT = Given a 3-CNF formula ϕ, ϕ is satisfiable if in each clause, atleast 1 literal is true.

So, using 3-SAT $\leq_m$ 4-SAT, we can show 4-SAT is np-hard as 3-SAT is np-hard.

Reduction from 3-SAT to 4-SAT,

For every clause in 3-SAT, add an extra variable to the clause to generate 4-CNF clauses.

$(a_1 \vee a_2 \vee a_3) \longrightarrow (a_1 \vee a_2 \vee a_3 \vee y_1)$

Add variables $y_1, y_2 \ldots y_j$ for each clause 1 to j.

⇒)

① If 3-SAT is satisfiable, every clause contain a literal which is true.

18CS10062

PAGE NO.
DATE: / /

Set the value of this new variable y_j as 0 $\forall$ j clauses

Thus clause contain a true literal and a false literal.

Thus 4-SAT is also satisfied for ϕ.

($\Leftarrow$) If 4-SAT is satisfied,

atleast 1 true literal and atleast 1 false literal in ϕ.

If the additional variable y_j is false, remove it from all clauses and resulting ϕ'' is in 3-SAT.

If additional variable y_j is true, remove any other literal from the CNF to generate a 3-SAT cnf.

$\therefore$ 3-SAT $\leq_m$ 4-SAT

3-SAT is np-complete iff 4-SAT is np-complete.

$\therefore$ 4-SAT is np-complete.

18CS10062

Now Showing 4-SAT $\le_m$ NonTrivial SAT

~~For each clause~~

($\Rightarrow$)

If 4-SAT is satisfied, then each 4-cnf clause contain atleast one true literal and atleast 1 false literal.

So irrespective of values of other literals, each clause contain atleast 1 true and atleast 1 false literal.

$\longrightarrow$ So NonTrivial SAT is satisfied

($\Leftarrow$)

If non-trivial SAT is satisfied, ~~so~~ each clause contain atleast 2 literals such that one literal is true and other literal is false.

- We can remove extra variables from each clause if # of clause > 4 whose values are false.

So that new clause has size 4 and is ~~of~~ having 1 True and 1 false literal.

- If # of clause = 4, then it is 4-CNF
- If # of clause < 4, add extra variable contain value false to the clause, so that it become 4-CNF.

Thus 4-SAT $\le_m$ NonTrivial SAT is possible.

18CS10062

PAGE NO
DATE: / /

∴ Since 4-SAT is np-complete,

then Non-Trivial SAT is np-complete.

NonTrivial SAT $\in$ NP as it has a polynomial time certificate $\rightarrow$ an assignment of variables such that atleast 1 literal is true and atleast 1 literal is false.

∴ NonTrivial SAT is Np-Complete

Hence proved.

18CS10062

3. Next-Path is in P

A polynomial time algorithm exists that can determine Next-Path in polynomial time.

$G = (V, E)$ where each edge has length $= 1$

$V = \#$ vertices
$E = \#$ edges in G

Source $= s$

Destination $= t$

$P =$ shortest path between s and t of length k.

~~$P' =$ 2nd shortest path between s and t that has atleast 1 vertex/edge different from those in path P.~~

~~Using Djikstra Algorithm,~~

Let $dp[u]$

Let P contain e edges and v vertices other than s and t. $((s,t) \notin V)$

There are 2 cases,

(1) Remove 1 edge $\in e$ at a time ~~[illegible]~~ from G and check if a path exists between s and t of shortest length using Djikstra(s, t, G)

~~[illegible]~~

Now add this edge back and remove some edge different for this edge. Repeat the above process.

PAGE NO.
DATE: / /

② Remove one vertex at a time ~~from~~ $\in V$ from G.
Now find the shortest length path between s & t
using Djikstra (s, t, G)

Now add this vertex back and repeat the
procedure for other vertices.

Now from all the above paths calculate in ① & ②,
take the shortest length path and
if length of this path $\leq 100k$,

then Next-Path returns **Yes**

else Next-Path returns **No**.

All paths obtained in ① & ② have atleast 1 edge
or 1 vertex different from those in Path P.

Djikstra (s, t, G) runs in $O(V(E+V))$ time for
adjacency list representation of graph G.

Case ① takes $E \times O(V(E+V))$ time as in worst
case P contains all edges.

Case ② takes $V \times O(V(E+V))$ time as in worst
case P contain all vertices.

$\therefore$ Total time Complexity = Case ① + Case ② + finding ~~ratio~~ shortest path from ① and ② and checking $\leq 100k$

$$= O(EV(E+V)) + O(V^2(E+V)) + O(1)$$

$$= O(EV(E+V))$$

$\{$ In worst case $O(E) = O(V^2)$ $\{$ Undirected graph $\}$ $\{ E = \frac{V(V-1)}{2} \}$ $\}$

$$= O(V^3(V^2+V))$$

$$= O(V^5)$$

= polynomial time algorithm.

$\therefore$ Next-path algorithm is in P as it is solvable in polynomial time.