7

AKHIL 18CS10070

(1) We will show that 2-SAT can be solved in polynomial time.

→ construction: make a directed graph having nodes as $x_1, \bar{x}_1, x_2, \bar{x}_2 \ldots x_n, \bar{x}_n$ (variables and their negations) summing up to $2n$ total nodes.

NOTES

→ the clause $(x_i \vee x_j)$ can be rewritten as $(\bar{x}_i \Rightarrow x_j) \wedge (\bar{x}_j \Rightarrow x_i)$

S	M	T	W	T	F	S
30						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29

SEP 18

→ Now in this graph if for any i, there exists a path from x_i to $\bar{x}_i$ and also from $\bar{x}_i$ to x_i then both x_i and $\bar{x}_i$ should be of same truth value in order to satisfy the constraint graph but it is a contradiction, hence the formula can't be satisfied by a valid ~~set~~ assignment of truth variables.

→ As finding the path / checking for path existence between 2 nodes is in polynomial time, $\forall i$ we will check the existence of path from x_i to $\bar{x}_i$ and from $\bar{x}_i$ to x_i.

→ Total checks $= 2n$ (2 for each i)

Total time $= 2n \cdot p(n)$ { $p(n)$ is time taken to check path existence b/w 2 nodes) also $p(n)$ is polynomial time

$\therefore$ 2-SAT is in P $\because$ if 2-SAT is NP-complete every problem in NP can be reduced to 2-SAT in polynomial time $\therefore$ NP = P.

---

(2) • We will show a reduction from 3-SAT to Max cut

let 3-SAT $\underset{m}{\Rightarrow}$ Max-Cut

• first we will show reduction of 3-SAT to NAE 4-SAT

• NOTES for an instance ~~of~~ of 3-SAT, we replace each clause $x_1 \vee x_2 \vee \bar{x}_3$ to NAE(x_1, x_2, x_3, z) leading to an instance of x.

- Now let $x_1, x_2, x_3 \dots x_n$ be a satisfying assignment to ϕ. $\therefore$ for $z = 0$, $x_1, x_2 \dots x_n, 0$ is satisfying assignment ϕ' for NAE 4-SAT.

- Also, if $x_1, x_2 \dots x_n, z$ is satisfying assignment for NAE 4-SAT, $\bar{x}_1, \bar{x}_2 \dots \bar{x}_n, \bar{z}$ also satisfies.

- In one of these two solutions, $z=0$. This assignment corresponds to a satisfying assignment of ϕ.

- Now, we ~~will~~ show reduction of NAE 4-SAT to NAE - 3SAT
split each clause $C_i = \text{NAE}(x, y, z, w)$ to $C_i^1 = \text{NAE}(x, y, a_i)$ and $C_i^2 = \text{NAE}(z, w, \bar{a}_i)$
then we show reduction of NAE 3-SAT to Max-Cut
~~Max-Cut~~ $\phi_M \Rightarrow (G, c)$

- $\forall x_i$, add two nodes x_i and $\bar{x}_i$ to G and then connect them by edge e_i of $10 \times$ total number of clauses $= 10 \cdot m$

- Now, for each clause $C_i = \text{NAE}(a, b, c)$ we add edges $a-b$ and $b-c$ and $a-c$ each of capacity 1.

so $C = n(10m) + 2m = m(10n+2)$

Now if ϕ is satisfiable, $\exists$ cut in G such that one side contains all vertices that have value 1 and other side contains all vertices with value 0 in ϕ.

- NOTES Since either x_i or $\bar{x}_i$ is 1 in assignment, all ~~variables~~ edges contribute $(N \cdot 2m)$ capacity to the cut.

- Also for each clause, exactly 2 edges go through
8 cut as ϕ is satisfying assignment.
If G contains cut of size $n \cdot M + 2 \cdot m$
9 first each variable edge goes through cut as if any variable gets missed, max capacity possible
10 is $(n-1)(10m) + 8m) = 10nm - 7m < 10nm + 2m$

- 11 Since no cut can get all 3 edges of a clause as it forms a triangle $\therefore$ if any clause edge
12 is missed capacity $< 10nm + 2m$

1 $\therefore$ all clauses are satisfiable $\Rightarrow$ satisfying assignment exists.

2 So, the reduction from 3-SAT to Max-Cut is valid.

3

(5) (a) FALSE.
4 We will prove by contradiction.

5 Let $P = \text{SPACE}(n)$ $\therefore \exists$ algorithm to simulate TM with space n in n^c ($c =$ constant) time.
6 But, this means $\exists$ algorithm to simulate a n^2 space TM in n^{2c} time.
7 $\therefore$ $P = \text{SPACE}(n) = \text{SPACE}(n^2)$

But by space hierarchy theorem,
$\text{SPACE}(n) \neq \text{SPACE}(n^2)$

NOTES $\therefore$ this is a contradiction
hence $P = D\,\text{SPACE}(n)$ is false.

(b) TRUE

$L = \{ s \mid s$ contains properly nested paranthesis $\}$

- L can be solved in polynomial time, let ϕ be an instance of L. Let '(' and ')' are opening and closing paranthosis.
- Maintain cnt=0 variable (having initial value =0)
- Iterate ϕ from L to R starting from begining.
- Whenever we encounter '(' (opening parenthesis) increase cnt by 1 and when we encounter closing paranthosis decrement cnt by 1.
- If cnt < 0 at any time while iterating reject ϕ
  elseif cnt > 0 after the iteration is complete, reject ϕ.
  else accept ϕ. —— (1)
- similarly, we ~~can~~ need to iterate from Right to Left set cnt=0, if we encounter ')' do cnt++, else if we see '(' do cnt--.
- apply the condition shown in (1),

so, L is in P and takes O(n) space and is also in SPACE(n).

③ (a) proof by contradiction

say $\exists L \in NP$-complete and $L \in coNP$
as $L \in NP$-complete $\Rightarrow L \in NP$ $\therefore$ $L \in NP$ & $CoNP$
also any NP language (say L_2) can be reduced to L
by using the definition of NP-complete.

So All languages L_2 in NP ~~can~~ also belong to coNP as they can be reduced to L in polynomial time
$\therefore$ $NP = CoNP$
which is a contradiction hence, no NP can be in CoNP.

(b) we know that for a language L in P, $\neg L$ is also in P [because P is closed under negation]

$\therefore$ L is both in NP and CoNP
as $NP \neq CoNP$, there exists some L which is not in P
$\Rightarrow$ L is coNP-complete $\Rightarrow$ L is in CoNP
AS $NP \neq CoNP$,
$\exists L$ in NP but not in CoNP.

Scanned with CamScanner

4 (a) Need of time-constructability is to clock the time when a machine runs for simulating the machine only for $f(n)$ steps on input of n length using only $O(f(n))$ number of steps.

In order to do this we must be able to compute $f(n)$ value in $O(f(n))$ time.

Also, if $f(n)$ cannot be computed in $O(f(n))$ time then total runtime of machine will not be fixed and may take arbitrary values.

16 SUNDAY