CPTH 1

Somnath Bhattacharjee
BMC201954

February 15, 2021

Question 1.

Let $\mathscr{L}$ be a finite language of size k
Now for a word w we can simply do linear search on $\mathscr{L}$ to check whether $w \in \mathscr{L}$ or not. and clearly that will take $O(k)$ time
Hence we have a $O(n)$ time algorithm for the membership of $\mathscr{L}$ as k is constant
Hence $\mathscr{L} \in \text{DTIME}(n)$

Question 2.

Let φ be DNF formula of n variables ($x_1, \ldots, x_n$ say)
We will define a Linear time algorithm to find the satisfibility of φ
Let $\varphi = c_1 \vee c_2 \vee \cdots \vee c_m$
And each $c_i = l_1 \wedge \cdots \wedge l_k$
Let $|c_i|$ denotes the number of literals in c_i
Now note φ is satisfible $\iff \exists c_i$ s.t. c_i is satisfiable .....(i)
Now Let $\max\{|c_i|\} = k$
Now for a fixed c_i if we find a l_j in c_i s.t. $\neg(l_j)$ also is in c_i then c_i is NOT satisfiable.
And if there is no such l_j in c_i then c_i is satisfiable .....(ii)
And this thing can be checked by a linear search on c_i for each literals l_j in c_i which will take a total of $O(k^2)$

Now based on (i),(ii) observations we can design an algorithm which will check (ii) for every clause and if it comes up with such clause then it will accept it otherwise reject
And clearly the entire algorithm will take $O(mk^2)$ as there are m clauses

Now let us write the algorithm formally

DNF-SAT(φ)

```
SAT← 0
for each c_i :
    for each l_j in c_i:
        if ¬l_j is in c_i:
            SAT ← 0
            Break the second for-loop        //i.e. stop checking for that clause
        else:
            SAT←1
return(SAT)
```

Now the language DNF-SAT=$\{\varphi \mid \varphi$ is satisfiable DNF formula$\}$ is clearly in P

So if we can convert any CNF formula to DNF formula in polytime preserving the satisfiability then basically it will be a karp reduction from SAT to DNF-SAT
Hence DNF-SAT will be NP-Complete which is in P
Hence $P = NP$

Question 3.

Let $P = NP$ and $A \subseteq \Sigma^*$, $A \neq \Sigma^*, \phi$
Hence we have a $w_1 \in A$ and $w_2 \in A^c$
Now Let B be any NP problem
Since $B \in P$ we have DTM M which computes B in polytime
Define $\gamma : \Sigma^* \to \Sigma^*$

$$\gamma(w) = \begin{cases} w_1 & \text{if } M(w) = 1 \\ w_2 & \text{otherwise} \end{cases}$$

Clearly γ is a polytime reduction from B to A as M runs in polytime

Hence A is NP-complete

Now if $A = \phi$ and B another language which can be reduced to A
Then for any $w \in B$ has to be mapped e=with some word in A
But A is empty
Hence $B = \phi$
Hence no non empty language can be reduced to A
Hence it is not NP-complete
For similar reason $A = \Sigma^*$ is not NP-complete since it has nothing in its complement hence if any language can be reduced to A then it has to be Σ^*

Question 4.

Note if M_S can decide S in polytime then we have a polytime DTM M which decides SAT by using M_S

Since it is NP-complete

Now we will describe a DTM M' which on input $\psi(x_1, x_2, \ldots, x_n)$ a satisfiable CNF formula returns the assignment
Clearly when $n = 1$ it will take constant time and asnwer correctly
Now for $n > 1$ it will assign $x_1 = 0$ and $x_1 = 1$ separately and converts the formula to a CNF formula of $(n-1)$ variables say $\psi_0(x_2, \ldots, x_n)$ and $\psi_1(x_2, \ldots, x_n)$ respectively
Now it will run ψ_0, ψ_1 on M consecutively
Note if ψ is satisfiable then one of ψ_0, ψ_1 must be satisfiable
So if $M(\psi_0) = 1$ then it will continue with ψ_0 and assign 0 to x_1 and solve it recursively otherwise it will assign 1 to x_1 and continue with ψ_1
Note M' will call M $2n$ times
It can call it one more time ti check whether the input is satisfiable or not
Hence the entire thing can be done in polynomial time

Question 5.

Let for a word w , 1^w represents the unary representation of w

Let for a language $L \in \text{NTIME}(2^{n^c})$
Define $UnL_{pad} = \{1^{\langle x, 2^{|x|^c} \rangle} \big| x \in L\}$

Claim. UnL_{pad} is in NP

Let M be exponential time NDTM for L
We will design a polytime NDTM M' for UnL_{pad}
Which on input y will find out x s.t. $y = 1^{\langle x, 2^{|x|^c} \rangle}$
(if no such x exists then it will simply reject it)
Then it will run x on M and stop after $2^{|x|^c}$ steps and return $M(x)$
Since M is a NDTM M' will be non-deterministic as well
Clearly M' is polytime as $O(|1^{\langle x, 2^{|x|^c} \rangle}|) \ni 2^{|x|^c}$
Hence our claim is true

Now UnL_{pad} is in P according to our hypothesis
So we have a polytime DTM M'' to check whther $1^{\langle x, 2^{|x|^c} \rangle} \in UnL_{pad}$ on input $\langle x, 2^{|x|^c} \rangle$
Hence for a word x we can use a exponential DTM $\mathcal{M}$ which converts x to $1^{\langle x, 2^{|x|^c} \rangle}$ and then returns $M''(1^{\langle x, 2^{|x|^c} \rangle})$
Clearly $\mathcal{M}$ solves the membership problem for L in exponential time
Hence NEXP $\subseteq$ EXP
or, NEXP=EXP

Question 6.

Claim. If $A \leq_P B$ then $A^c \leq_P B^c$

Hence L we have a polytime computable map $\gamma : \Sigma^* \to \Sigma^*$ s.t. $\gamma(A) \subseteq B$ and $\gamma(A^c) \subseteq B^c$
Hence $A^c \leq_P B^c$

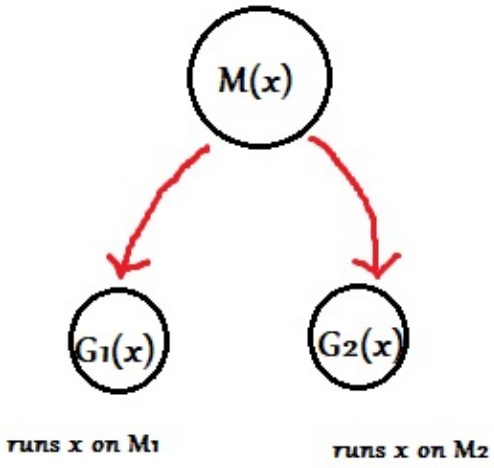
Hence our claim is true

Let $\mathscr{L}$ is a NP-complete problem
Now for a co-NP problem L we can say $L^c \in NP$ and $L^c \leq_P \mathscr{L}$
Hence $L \leq_P \mathscr{L}^c$
And clearly $\mathscr{L} \in$co-NP
Hence $\mathscr{L}^c$ is co-NP-complete

Again Let $\mathscr{L}$ is a co-NP-complete problem
Now for a NP problem L we can say $L^c \in$ co-NP and $L^c \leq_P \mathscr{L}$
Hence $L \leq_P \mathscr{L}^c$
And clearly $\mathscr{L} \in NP$
Hence $\mathscr{L}^c$ is NP-complete

Question 7.

1.
$L_1, L_2 \in NP$ hence we have NDTM M_1, M_2 for them
Now let us construct a NDTM M which on input x runs x on M_1 and M_2 both non deterministically
Now If $G_1(x)$ and $G_2(x)$ are the configuration graph of $M_1(x)$ and $M_2(x)$ respectively then the configuration graph of $M(x)$ will be



Now $x \in L_1 \cup L_2 \iff$ there exists an polytime accepting run in either $G_1(x)$ or in $G_2(x)$
$\iff$ there exists a accepting poly-time accepting run from the initial tape-config in M (as there is only
two edges from initial config of $M(x)$ one towards $G_1(x)$ another towards $G_2(x)$)
$\iff$ x will be accepted by M
Hence $L_1 \cup L_2 \in NP$

Now again we will have DTTM M_1, M_2 for L_1, L_2 which accepts a word in presence of the certificate of the input
Let us construct DTTM M which on input $\langle x, y\#z\rangle$ runs $M_1(\langle x, y\rangle)$ and $M_2(\langle x, z\rangle)$ con-

secutively and returns $M_1(\langle x,y\rangle) \wedge M_2(\langle x,z\rangle)$

Now $x \in L_1 \cap L_2 \iff$ we have poly-length certificate y,z s.t. $M_1(\langle x,y\rangle) = 1 = M_2(\langle x,z\rangle)$
$\iff M_1(\langle x,y\rangle) \wedge M_2(\langle x,z\rangle)$
$\iff$ we have polylength certificate $y\#z$ s.t. $M(\langle x, y\#z\rangle) = 1$
Hence $L_1 \cap L_2$ is in NP

2.
L is NP-complete
$\implies L^c$ is co-NP complete (problem 6)
Hence for a co-NP problem A we have a polytime reduction γ
Again $L^c \in NP$
Hence we have a polylength certificate u for $\gamma(x)$ whenever $x \in B$
Hence we can create a polytime DTTM M which on input x converts it to $\gamma(x)$ and uses the certificate for $\gamma(x)$ to check whether it is in L^c or not
Hence B is in NP
Hence co-NP $\subseteq$ NP

Now for $L \in NP$ we have $L^c \in$ co-NP
Hence $L^c \in NP$ as co-NP $\subseteq$ NP
Hence $L \in NP$
or, $NP \subseteq$ co-NP
or, NP=co-NP

3.
Let $\mathscr{L} \in P$
Hence we have a polytime DTTM M for $\mathscr{L}$
Hence for $x \notin \mathscr{L}$ we have $M(x) = 0$
Now we can construct a DTTM M' which on input returns $\neg M(x)$
Clearly M' runs in polytime
It can be easily observed that M' recognises $\mathscr{L}^c$
Or, co-P $\subseteq P$

Again for $\mathscr{L} \in P$ we have $\mathscr{L}^c \in$co-P $\subseteq P$ hence $\mathscr{L} \in$co-P
Hence $P \subseteq$co-P
or, $P =$co-P
Hence $P \subseteq NP \cap$co-NP

Question 8.

1.
Let $L \in$ NSPACE$(f(n))$
Then We have a NDTM M s.t.

$x \in M \iff \exists$ a path of $f(n)$ space from initial config c_0 to some accepting config c_{acp}
$\iff$ for the config graph $G(x), \langle G(x), c_0, c_{acp}\rangle \in PATH$

Now if the space for the accepting run is $O(f(n))$ the all possible config will be $O(2^{f(n)})$
Hence $|\langle G(x), c_0, c_{acp}\rangle| = O(2^{f(n)})$

Since $PATH \in$ co-NL we have a $O(\log(2^{f(n)})) = O(f(n))$ Space NDTM for L^c
Hence $L \in$ co-NSPACE$(f(n))$
or NSPACE$(f(n)) \subseteq$ co-NSPACE$(f(n))$

Now for $L \in$ co-NSPACE$(f(n))$ we have $L^c \in$ NSPACE$(f(n))$
or, $L^c \in$ co-NSPACE$(f(n))$
or, $L \in$ NSPACE$(f(n))$
Hence NSPACE$(f(n)) \subseteq$ co-NSPACE$(f(n))$
or, NSPACE$(f(n))$ = co-NSPACE$(f(n))$

2.
It is obvious that $EXP \subseteq EXP^{EXP}$

But reverse is not True
Define 2-Exp=$\bigcup_{c \in \mathbb{N}}$ DTIME$(2^{2^{n^c}})$

Now note for a TM M for the languagke $L \in EXP^{EXP}$
M can make exponentially many oracle queries and each query can be exponential in length. Since the query can be $O(2^{n^c})$ length if we use a Exp-TM for that it can take exponential of exponential time i.e. $O(2^{2^{n^c}})$
And since there is only exponentially many oracle calls we can use the exponential TM M' for the oracle queries and every thing can be done in exponential of exponential time
Hence it is clear that $EXP^{EXP} \subseteq$ 2-Exp

Now let us define
2-Exp-comp= $\{\langle M, x, n\rangle \mid$ the TM M accepts the word x in 2^{2^n} steps$\}$
Similarly let define
Exp-comp= $\{\langle M, x, n\rangle \mid$ the TM M accepts the word x in 2^n steps$\}$

Claim. 2-Exp-comp is 2-Exp-complete

It is clear that it is in 2-Exp (we can conclude this by simply run x on M and stop the program after 2^{2^n} steps and examine the output)
Now let L be a 2-Exp language and the DTTM M computes L correctly
Then note for any $x \in \Sigma^*$

$$x \in L \iff \langle M, x, |x|\rangle \in \text{ 2-Exp-comp}$$

Hence we have a polytime reduction from L to 2-Exp-comp by sending x to $\langle M, x, |x|\rangle$
Hence our claim is true

For similar reason Exp-comp will be EXP-complete, in particular it will be in EXP

Now we will deduce a reduction from 2-Exp-comp to Exp-comp
It is obvious that

$$\langle M, x, n\rangle \in \text{ 2-Exp-comp} \iff \langle M, x, 2^n\rangle \in \text{ Exp-comp}$$

But the only problem is $O(|\langle M, x, 2^n\rangle|) = O(2^{|\langle M, x, n\rangle|})$

Hence the reduction will be exponential

Hence if we take a Exp Oracle Machine M' which on input $\langle M, x, n\rangle$ converts it to $\langle M, x, 2^n\rangle$ and then uses the oracle Exp-comp to check whether $\langle M, x, 2^n\rangle$ is in Exp-comp or not.
Clearly M' can compute 2-Exp-comp

Now since it is 2-Exp-complete for any language $\mathscr{L} \in$ 2-Exp we can reduced $\mathscr{L}$ into 2-Exp-comp in polytime and then using M' we can solve the membership problem

Hence 2-Exp $\subseteq EXP^{EXP}$
or, 2-EXP= EXP^{EXP}

And using time hierarchy theorem we can say EXP $\subsetneq$ 2-Exp = EXP^{EXP}

Question 9.

Note $\phi \in \Sigma_2$SAT $\iff \exists x \in \{0,1\}^{p(n)}\ \forall y \in \{0,1\}^{p(n)} \phi(x,y) = 1$
Hence from definition of Σ_2^P it is clear that Σ_2SAT $\in \Sigma_2^P$

Now if P=NP then PH=P
Hence Σ_2SAT $\in$ PH = P

Question 10.

Let $f, g : \Sigma^* \to \Sigma^*$ two logspace reduction with the transducers M_f and M_g respectively
We are done if we can construct a logspace transducer M which computes $f \circ g$
Before that let us modify M_g with a counter C which on a call 'i' returns the i^{th} bit of the output of $M_g(x)$ for a input x
Note the transducer property will not change for this modification

Now for an input x we will start computation for M_f on $g(x)$
For this computation whenever we required some bit of $g(x)$ we will asked it from M_g
After getting the required the bit we will delete the space used for that call
Now it can be observed that the call for the bit can be done in log space and the computation in M_f for each bit can be done in log space
Hence the entire thing can be done in logarithmic space as the space is being reused

For this Assignment I have discussed with **Saswata Mukherjee**