CPTH-EXAM

Somnath Bhattacharjee
(BMC201954)

February 28, 2021

Question 1.

Assume $P = NP$
Now $A \subseteq \Sigma^*$, $A \neq \phi, \Sigma^*$
Hence we can find w_1, w_2 s.t. $w_1 \in A$ and $w_2 \in A^c$
Now for any NP-problem B we have a polytime TM M which recongines B
Now 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

Question 2.

Let m be a natural number
Let its prime factorization be

$$\begin{aligned} & p_1^{k_1} \times \cdots \times p_n^{k_n} = m \\ \implies & k_1 \log p_1 + \cdots + k_n \log p_n = \log m \\ \implies & k_1 + \cdots + k_n \leq \log m \end{aligned}$$

$$(\text{as } p_i \geq 2 \implies \log p_i \geq 1)$$

Now let $L = \{(m, n) \big| m \text{ has a factor in } [2..n]\}$ is clearly in NP as

$$(m, n) \in L \iff \exists p \leq n, p \geq 2 \text{ s.t. } p \big| m$$

Hence we have a polytime DTM M for L by assuming P=NP

Now we will describe an algorithm to find a prime factor of m
We will check whether m has a factor in $[2..\frac{m}{2}]$ using M
if no then we will check for $[\frac{m}{2}..m]$ interval (checking for $[2..m]$ will be same as checking for $[\frac{m}{2}..m]$ as $[0..\frac{m}{2}]$ does not contain any factor)

If we find no factor then we can simply declare m is prime otherwise we will again bisect the smaller interval containing a factor and do the same
Basically we will continue binary search until we get a specific factor
But during binary search we will always choose the interval which contains the smaller numbers
Since we are choosing the smaller interval we will always find the smallest factor hence the factor will be prime

Let p be the factor
Then we will recurse this algorithm on $\frac{m}{p}$
We will continue the recursion until we found a prime number or 1 which will be the base cases
For the base cases problem will be trivial

Now the binary search will use the TM M $\log m$ many times
And each call it will use $O((\log m)^c)$ time
Hence total time will be $O((\log m)^{c+1})$

Now total time will be

$$T(m) = T(\frac{m}{p}) + O((\log m)^{c+1})$$

Now each time we will remove one factor from m then will continue the recursion
Hence the recursion will continue for $k_1 + \cdots + k_n$ many times (where $p_1^{k_1} \times \cdots \times p_n^{k_n} = m$ is the prime factorization of m)
Hence $T(n) \leq (k_1 + \cdots + k_n)O((\log m)^{c+1}) \leq \log m O((\log m)^{c+1}) = O((\log m)^{c+2})$

Question 3.

The given algorithm certifies that $SAT \in$ Exp
But we know $P \subseteq$ Exp
So there is still a chance that SAT has a polytime algorithm since having an exponential algorithm doesn't i,ply it cannot have a polytime algorithm
It may have a polytime algorithm which works in a different way

Question 4.

Note $\langle G, k \rangle \in$ MAX-CLIQUE $\iff \exists\, k$ size clique in G $\forall n$ size clique in G $n \leq k$
Now " $n > k$" can be represented with a polytime TM M s.t. that statement will be equivalent to "$M(G, x, k) = 1$"
And clearly $|k/n$ size clique in $G| = |G.V| + |G.E| = O(|\langle G, k \rangle|)$
Hence MAX-CLIQUE is in Σ_2

Now we know that if $P = NP$ then $PH = P$ or, $\Sigma_2 \subseteq P$
Hence if $P = NP$ then MAX-CLIQUE $\in$ P

We know k-size Clique $= \{G | G$ has a k-size Clique$\}$ is a NP problem

Now for P=NP let we have a $O(P_k(n))$ time total TM for k-size Clique

Now we will design a DTM M which on input $\langle G, k\rangle$ will check for $G \in k$-size Clique
if no then it will reject the input other wise for all i in $[k+1..|G.v|]$ it will check for $G \in i$-size Clique
If for some i, $G \in i$-size Clique then it will reject it
Otherwise accept it

Clearly M will recognise MAX-CLIQUE
And the runnig time will be $O(P_k(n) + P_{k+1}(n) + \cdots + P_{|G.V|}(n)) = O(n^c)$ for some c

Question 5.

Let A_1, A_2 are two DFAs with Q_1 and Q_2 set of states
Then from TOC we know that if $A_1 \neq A_2$ then $\exists$ a word w of lenght at max $|Q_1| + |Q_2|$ s.t. $w \in L(A_1)$ but $w \notin L(A_2)$

Hence for two NFA A_1, A_2 if $A_1 \neq A_2$ then $\exists$ a word w of lenght at max $2^{|Q_1|+|Q_2|}$ s.t. $w \in L(A_1)$ but $w \notin L(A_2)$

Hence we always have an exponential certificate for any member of $\overline{EQ_{REX}}$

Now we will construct a NTM for $\overline{EQ_{REX}}$ which will use a lenght counter which on input R, S creates the NFA s R_n, S_n and two set of states R_q and S_q
initially when the counter will be 0 it will run the 0 length word ϵ on R_n and on S_n and add all states to R_q which can be reached using ϵ in R_n and add all states to S_q which can be reached using ϵ in S_n
Then it will increase the counter by 1
Now from all the states in R_q it will run $1, 0$ non deterministically in R_n and add all those states to R_q which can be reached by this
same for S_q
Then again it will incresae the counter by 1 and non deterministically add all those states to R_q which can be reached from some states of current R_q. Similar for S_q and this will be continued
Basically when the counter will show c it will record non deterministically all the states R_q which can be reached by each strings of length c in R_n
Same for S_q
Then it will record all the states reached by the states in R_q and S_q by $0, 1$ non deterministically and increase the counter by 1
Basically it will record all states recheable by $c+1$ length states
We will stop when counter will show $2^{|R_n|+|S_n|}$
Clearly for $x \in \overline{EQ_{REX}}$ we have a certificate w
We can use that certificate to reach the specific R_q and S_q and then we will check is there any final state present in one but not in another
Note that specific path for the certificate will be poly pspace since any time the size of R_q and S_q will be bounded by $|R_n|$ and $|S_n|$
Note to write the max length which is $2^{|R_n|+|S_n|}$ will take $|R_n| + |S_n|$ bits

Hence $\overline{EQ_{REX}}$ will be in NP-SPACE
Hence EQ_{REX} will be in co-NPSPACE=PSPACE

Question 6.

Assuming A is Dyck Language ,i.e.,
$x \in A \iff \#_x] = \#_x[$ and for any prefix y of x, $\#_y[- \#_y] \geq 0$
We will use a DTM M which on the work tape will use a counter which will be initially 0
If the tape header will be on the ith alphabet in the input x then the counter will remember the value of $\#_y[- \#_y]$ where $y = x[1..i]$ the i lenght prefix of x
Hence from the definition it is clear that the input is in A iff at any point of time during the computation the counter should be non-negative and when the header will be at the end it must be 0

Now on input x the TM will start traversing from left to right whene ever the header will read [then it will increase the counter by 1, and whenever the header will read] it will subtract 1 from the counter
If any time the counter becomes negative it will reject x
If in the end the counter becomes 0 then it will accept x otherwise it will reject it

Clearly M will recoglise A
Now the counter is always less than or equal to the size of the input
Hence it will take $O(\log |x|)$ space to use the counter

Question 7.

We will modify the proof of the existence of oracle A s.t. $P^A \neq NP^A$ for some some extent

We can enumerate the Nondeterministic oracle machine with oracle A s.t. $M_i^A \in$ co-NTIME(n^i)
Now we will construct A step by step

<u>Construction of A at i-th step</u>

1. Choose a n_0 larger than the length of any string whose membership in A is decided within steps $\leq i-1$, also $2^{n_0} > n_0^i$
2. Run M_i on 1^{n_0}, whenever M_i asks the oracle query answer according to the current structure of A
3. If M_i accepts 1^{n_0} then for all string of length n_0, never include them into A
4. If M_i rejects 1^{n_0} then there must exists a poly length certificate w s.t. $M_i(1^{n_0}, w) = 0$ in $O(n_0^i)$ time
   Hence add all the strings of length n_0 which aren't asked as an oracle query during $M_i(1^{n_0}, w)$ computation
   There must be such string since there can be only n_0^i many oracle queries and

2^{n_0} many n_0 length strings and $2^{n_0} > n_0^i$

(Note in this algorithm we never repeat n_0 in any step, hence we will never add any string to A if it is already recorded "no")

Now Lets define $L_A = \{w | \exists x \in A \text{ s.t. } |x| = |w|\}$

We have proved in the class that $L_A \in NP^A$

Claim. $L_A \notin$ co-NP^A

Now if $L_A \in$co-NP^A
Then we have a Non deterministic oracle machine M_i^A for L_A
let n_0 was the integer choose in ith step of construction of A
Now if M_i accepts 1^{n_0} then clearly it will be self contradicting since from construction of A there will be no string of length of n_0 in A

Now if M_i rejects 1^{n_0} then there must exists some word x of length n_0 in A (from construction of A)
Hence from construction of L_A we can say $1^{n_0} \in L_A$ which will contradict that M_i rejects 1^{n_0}
Hence our claim is true

Question 8.

It can be observed that $|pad(s, n^2)| = |s\#^{|s|^2-|s|}| = |s|^2$

Now lets define $\gamma : \Sigma^* \to \Sigma^*$ which basically converts $pad(s, f(|s|))$ to s (by checking the number of # and if the number is equal to $|s|^2 - |s|$ then by removing all #s)
Clearly γ takes $O(|s|^2) = O(|pad(s, n^2)|)$ time

Now we have a $O(n^2)$ TM M for A

Note $|\gamma(x)| = \sqrt{|x|}$ Now we will design a DTM M' which on input x will see whether $\gamma(x)$ exists or not
if no then it will reject it otherwise it will return $M(\gamma(x))$

Clearly M' recognises $pad(A, n^2)$

$$\begin{aligned}\text{Now time taken by } M'(x) &= \text{time taken by } \gamma(x) + \text{ time taken by } M(x)\\ &= O(|x|) + O(|\gamma(x)|^2)\\ &= O(|x|) + O(|x|) = O(|x|)\end{aligned}$$

Hence $pad(A, n^2) \in$DTIME(n)

Question 9.

We know that there exists a language L s.t. $L \in$ DTIME(2^{2^n}) but $L \notin$ EXP
Let 1^L be the unary representation of L
Then note for $w \in L$, $|1^w| = 2^{|w|}$

Claim. $1^L \notin$ P

Suppose $1^L \in$ P then we have a polytime DTM M for 1^L
Now we can construct a TM M' which on input w writes 1^w which takes $O(2^{|w|})$ time
Now M' will use M to check whether 1^w is in 1^L or not which will take $O((2^{|w|})^c)$ time.
Clearly M' will stop the execution in exponential time and it can recognise L completely
which contradicts $L \notin$ EXP
Hence our assumption was wrong, our claim is true

Now clearly $1^L \in P/poly$

Question 10.

We know that if $Exp \subseteq P/poly$ then $Exp = \Sigma_2$
Hence if $P = NP$ then $Exp = \Sigma_2 \subseteq PH = P$ which will contradict the time hierarchy theorem
Hence $Exp \neq P/poly$ if $P = NP$