CPTH-EXAM

Somnath Bhattacharjee
(BMC201954)

May 8, 2021

Question 1.

Let L has a downward self reducible algorithm. We are assuming that for $|x| = 0$ R can determine the membership of x in poly time without using any oracle.

Now we will define the algorithm A inductively
For any input of length n A will use the algorithm A_n
when $n = 0$ $A_n = R$
for $n > 0$ A_n will use $R^{L_{n-1}}$. Now for any oracle call, if the call length is k we know that $0 \leq k \leq n-1$, hence it will use A_k to answer the oracle call. A will use separate tape to answer the oracle calls, and after each oracle call it will remember the oracle answer and clear the space used for the oracle call and reuse the space for the next oracle call.

Clearly the algorithm will always give the correct answer
Let A_n uses $S(n)$ space. clearly $S(0) = O(1)$ from our assumption Now since R runs in polynomial time A_n can do polynomially many oracle calls and for each oracle the spcae required is $\leq S(n-1)$
But the oracle space will be reused. Hence there will be atmax $S(n-1)$ (it will be safe to claim $S(n)$ is increasing) will be used for all oracle calls and since there are only polynomial many oracle calls hence to remember all oracle answers it will take polynomially many bits. and the rest part will be done in polytime hence in poly space as R runs in poly time
Hence we have

$$\begin{aligned} & S(n) = O(n^c) + S(n-1) \\ \implies & \sum S(n) - S(n-1) = \sum O(n^c) \\ \implies & S(n) = O(n^{c+1}) \end{aligned}$$

Hence L is in PSPACE.

Question 2.

From Arrora-Barak we know that $L = \{(A, perm(A)) \big| A \text{ is } n \times n \text{ matrix}\}$ is in perfectly complete -IP. Now the following claim will be sufficient to prove that L is in $PCP(poly\ n, poly\ n)$
(By perfectly complete IP we mean set of languages having an Interactive protocol with perfect completeness)
(From the protocol given in Arrora-Barak we can simply claim that it has a perfect completeness)

Claim. perfectly complete-IP$\subseteq PCP(poly\ n, poly\ n)$

Let P, V be the prover and the verfier for a language $L \in$ perfectly complete-IP with a $k(n)$ round protocol
Let for $k \leq k(n)$ $m_1, \ldots, m_k$ be the messages sent from the verifier upto k^{th} round and π_k be the answer of the prover at the end of the k^{th} round
Clearly length of each m_i is bounded by some polynomial $p(n)$

We will design a new verifier V'.
So to construct the required Probabilistically checkable proof we will do the following:
the proof π will have a bit for all possible $(m_1, \ldots, m_k)$ tuple and for all $k \in [k(n)]$
Now for a specific i possibilites of m_i is $2^{p(n)}$ as length of m_i is bounded by $p(n)$
For a specific k possibilites of $(m_1, \ldots, m_k)$ is $2^{p(n)} \times \ldots 2^{p(n)} = 2^{kp(n)}$
There are $k(n)$ many possibilities of k
Hence length of π will be $\sum_k 2^{kp(n)} = k(n)2^{O(k(n)p(n))}$
Hence with $k(n)p(n)$ random coin tosses verfier V' can access the fragments of π

Now V' will simulate V. If after a specific random choice $r \in \{0,1\}^{p(n)k(n)}$ V sents the messages $m_1, \ldots, m_{k(n)}$ and corresponding proofs for each $(m_1, \ldots, m_k)$ is π_k then verifier V' on random choice r will query the $\pi[m_1, \ldots, m_k]$ bit for all $k \in [k(n)]$ and clearly after this query the fragments of proofs will be π_k and will use $r, m_1, \pi_1, \ldots, m_k\pi_k$ just like V used them for all k
Clearly number of queries are $k(n)$

Clearly V' will accepts π iff so V accepts the corresponding proof (follows from the construction of V' and π)
Now V has perfect completeness hence V' will have perfect completeness
Hence $L \in PCP(k(n)p(n), k(n)) \subseteq PCP(poly\ n, poly\ n)$

Question 3.

Suppose $SAT \in PCP(o(\log n), 1)$.
We will define a polytime recursive algorithm for SAT
Let φ be a CNF formulla with $|\varphi| = n$
Now for a random choice r verifier V can accept a proof π or reject it probabilistically by constantly many queiries. If it is going to accept then for any random choice r V will accept it. otherwise for some fraction of choices of random bits no proof will be accepted.

Now we can define a function $V_{x,r}$ s.t. $V_{x,r}(\pi) = 1$ iff V on input x and random choice r accepts π
Note $V_{x,r}$ depends on constantly many variables not on the entire proof since V will ask constantly many queries i.e. $V_{x,r}$ depends on the queries of V only which are constantly many only.
So if the $i_1, \ldots, i_k$ are the bits corresponds to the queries where k is a constant then $V_{x,r}$ depends on $x_{i_1}, \ldots, x_{i_k}$ and the assignment of x_{i_j} will be the $i_j{}^{th}$ bit of the proof
Now if V accepts a proof π then $x_{i_1}, \ldots, x_{i_k}$ will definitely have some assignments corresponds to the queries and if no proof is accepted then for some random choices r $x_{i_1}, \ldots, x_{i_k}$ will have no assignment
Now $V_{\varphi,r}$ can be written in a form of disjuctive clause c_r of k literals where c_r will have satisfying assignment if so has $V_{\varphi,r}$

Define $\varphi' = \wedge_r c_r$
Clearly φ' is a CNF formula of the variables $x_1, \ldots, x_m$ where each x_i represents the i^{th} bit of a $2^{o(\log n)}$ length proof.

So if φ is accepted then there exists some proof π and the i^{th} bit of π will be an assignment of x_i hence defintely φ' will have a satisfying assignment
otherwise there will be no proof hence no assignment of x_i

Hence φ is satisfiable iff so is φ'

Now there is $2^{o(\log n)}$ many possibilites of random choices. Hence

$$|\varphi'| = 2^{o(\log n)} < n = |\varphi|$$

hence φ satisfiability problem reduced to lesser size φ' satisfiability
We will continue this recursion untill the CNF formula size becomes $\log n$ which will be the base case and can be solved on polytime on n.
And since in each step the size of the CNF formula is strictly decreasing hence total number of recursion will be $O(n - \log n)$

Hence the total time is in polynomial of n

Question 4.

Let $\{\chi_\alpha | \alpha \in [2^3]\}$ be the Fourier basis
Let $f(x_1, x_2, x_3) = Maj(x_1, x_2, x_2)$
Now

$$f_{Re} = \sum_\alpha \tilde{f}(\alpha)\chi_\alpha$$

Now we know that

$$\tilde{f}(\alpha) = \langle f, \chi_\alpha \rangle = \frac{1}{2^3} \sum_{x \in \{0,1\}^3} (-1)^{f(x)}(-1)^{\langle x,\alpha \rangle}$$

.....(i)

Now suppose α be such that $\alpha_1 + \alpha_2 + \alpha_3 = 0$ (i.e. α has even no. of 1)
Define $\overline{x} = (1 - x_1, 1 - x_2, 1 - x_3)$, clearly $f(x) + f(\overline{x}) = 1$

$$\begin{aligned}\langle x, \alpha \rangle + \langle \overline{x}, \alpha \rangle &= \langle x + \overline{x}, \alpha \rangle \\ &= \langle 111, \alpha \rangle \\ &= \alpha_1 + \alpha_2 + \alpha_3 = 0\end{aligned}$$

Now clearly from (i)

$$\begin{aligned}2\tilde{f}(\alpha) &= \sum_x (-1)^{f(x)+\langle x,\alpha \rangle} + (-1)^{f(\overline{x})+\langle \overline{x},\alpha \rangle} \\ &= \sum_x (-1)^{f(x)+\langle x,\alpha \rangle} - (-1)^{f(x)+\langle x,\alpha \rangle} = 0\end{aligned}$$

Now define $X_o = \{x | x \text{ has exactly one } 1\}$, $X_e = \{x | x \text{ has exactly two } 1\}$
$\therefore |X_o| = 3 = |X_e|$

Now for $\alpha_1 + \alpha_2 + \alpha_3 = 1, \alpha \neq 111$

$$\begin{aligned}\sum_{x \in X_o} (-1)^{f(x)}(-1)^{\langle x,\alpha \rangle} &= \sum_{X_o} 1(-1)^{\langle x,\alpha \rangle} \\ &= \sum_{X_o, x=\alpha} (-1)^{\langle x,\alpha \rangle} + \sum_{X_o, x \neq \alpha} (-1)^{\langle x,\alpha \rangle} \\ &= -1 + (1 + 1) = 1\end{aligned}$$

$$\begin{aligned}\sum_{x \in X_e} (-1)^{f(x)}(-1)^{\langle x,\alpha \rangle} &= \sum_{X_e} (-1)(-1)^{\langle x,\alpha \rangle} \\ &= -\sum_{X_e, x=\overline{\alpha}} (-1)^{\langle x,\alpha \rangle} - \sum_{X_o, x \neq \overline{\alpha}} (-1)^{\langle x,\alpha \rangle} \\ &= -1 - (-1 - 1) = 1\end{aligned}$$

(If $x \neq \overline{\alpha}, x \in X_e$ then $x_i \neq \alpha_i$, hence $\langle x, \alpha \rangle = 1$)

Now we will break the (i) sum into four cases $\{\{000\},\{111\},X_o,X_e\}$

$$\begin{aligned}\tilde{f}(\alpha) &= \frac{1}{8}\Big((-1)^{f(000)+\langle 000,\alpha\rangle} + (-1)^{f(111)+\langle 1,\alpha\rangle} + \sum_{x\in X_e}(-1)^{f(x)}(-1)^{\langle x,\alpha\rangle} + \sum_{x\in X_o}(-1)^{f(x)}(-1)^{\langle x,\alpha\rangle}\Big)\\ &= \frac{1}{8}\Big(1+1+1+1) = \frac{1}{2}\end{aligned}$$

Now for $\alpha = 111$ again we do case breaking

$$\begin{aligned}\tilde{f}(\alpha) &= \frac{1}{8}\Big((-1)^{f(000)+\langle 000,\alpha\rangle} + (-1)^{f(111)+\langle 1,\alpha\rangle} + \sum_{x\in X_e}(-1)^{f(x)}(-1)^{\langle x,\alpha\rangle} + \sum_{x\in X_o}(-1)^{f(x)}(-1)^{\langle x,\alpha\rangle}\Big)\\ &= \frac{1}{8}\Big(1+1+\sum_{X_e}(-1)(-1)^{\overline{2}} + \sum_{X_o}(-1)^{\overline{1}}\Big)\\ &= \frac{1}{8}\Big(2-3-3)\Big) = -\frac{1}{2}\end{aligned}$$

Hence $f_{Re} = \frac{1}{2}\chi_{100} + \frac{1}{2}\chi_{010} + \frac{1}{2}\chi_{001} - \frac{1}{2}\chi_{111}$

Question 5.

a.
We will construct a sequence of advice $\{a_n\}$ for L
$a_n = l_1\#l_2\#\ldots\#l_k$ where l_is are all possible n-length strings in L
Clearly k is bounded by some polynomial $p(n)$ as L is a sparse set.
Hence $|a_n| = O(np(n))$

Now consider the TM M which on input x uses the advice $a_{|x|}$ to check x is a substring of $a_{|x|}$ or not and answer accordingly
Clearly M answers the membership of x in L correctly and $M(x, a_{|x|}$ runs in polytime
Hence L is in $P/poly$
(We prove that the advice taking TM definition is equivalent to $P/poly$ in assignment 2

b.
We are done from the assignment 2 problem 3

Alternative proof:
Let for $\varphi \in SAT$, x_φ is the lexicographically smallest certificate of x
Define $LSAT = \{(\varphi, w)\big|\varphi \in SAT, w \geq x_\varphi \text{ lexicographically}\}$
We know such w will always exists as x_φ is polynomially bounded
Clearly $LSAT \in NP$

Now if sparse set L is NP-complete then there is a polytime reduction γ from $LSAT$ to L
Now consider a CNF formula φ
We will define a recursive algorithm for that
We know if x_φ exists then there must be some w s.t. $x_\varphi \leq w$
So we can claim that φ is satisfiable iff $S_0 = [0\ldots w]$ contains certificate(satisfiable assignment) of ϕ.
Now we will recursively define S_i with $|S_i| < |S_{i-1}|$ and the property that φ is satisfiable iff S_i contains certificate(satisfiable assignment) of ϕ .....(i)
$m = O(p(n))$ where $p(n)$ is the polynomial corresponds to L, Hence m can be computed in polynomial time

Assume S_i contains more than $m+1$ elements where $m = \left|\{x \in L\big||x| \leq |\gamma(\phi, w)|\}\right|$
Clearly m is bounded by some polynomial of n as L is a sparse set.....(ii)

Now divide S_i in to $m+1$ equal length intervals.
Let the intervals are $[w_0$ to $w_1], \ldots, [w_m$ to $w_{m+1}]$ where w_is are increasing
Now we can simply assume $|\gamma(\phi, w_i)| \leq |\gamma(\varphi, w)| = m$ as $w_i \leq w$
(otherwise we have to adjust m s.t. $|\gamma(\varphi, S_0)| = m$, m will be still polynomially bounded by n and hence can be computed in polytime)

Hence from PHP we can say there will be some w_i, w_j $i < j$ s.t. $\gamma(\varphi, w_i) = \gamma(\varphi, w_j)$ as $\gamma|(\varphi, S_i)| \leq m$

Now one can easily verify that x_φ will never lie in between w_i to w_j hence we can remove all the elements between w_i to w_j from S_i to construct S_{i+1}
Clearly $|S_{i+1}| \leq |S_i| \frac{m}{m+1}$ .....(iii)
Hence our construction satisfies the statement (i)

Now if S_i contains at max $m+1$ elements then it will be our base case
We can simply check for each $w_j \in S_i$ manually which will take polnomial time in total as each checking needs polytime.....total m checking which is also in polynomial of n

Clearly our algorithm will answer correctly from the statement (i)
Now if the total number of recursion is in polynomial of n then we can say total time will be in polynomial of n
Now from (iii) we can say total number of recursion steps will be in polynomial of n because in the end $|S_k|$ will be $m+1$

$$
\begin{aligned}
& m+1 = |S_k| = |S_0| \Big(\frac{m}{m+1}\Big)^k = 2^{n^c} \Big(\frac{m}{m+1}\Big)^k \\
\implies & \log(m+1) = \Big[n^c + k\Big(\log \frac{m}{m+1}\Big)\Big] \\
\implies & k = O(n^{c'}) \qquad \text{(as from (ii) } m \text{ is a polynomial on } n\text{)}
\end{aligned}
$$

Hence we have a polytime algorithm for SAT which will imply $P = NP$