Habib University

shaping futures

S.No. 010974

End-Term Answer Book

good job!

Number of Sheet(s): _____

Invigilator's Signature: _____

Marks Obtained	
Q. 1	5
Q. 2	
Q. 3	4.5
Q. 4	6
Q. 5	6
Q. 6	6
Q. 7	
Q. 8	
Q. 9	
Q. 10	
Q. 11	
Q. 12	
Q. 13	
Q. 14	
Q. 15	
Total	27.5

Instructor's Signature & Date

Instructions for Students:

a. Read the following instructions carefully.
b. Please ensure that your mobile phone is switched off.
c. **All your personal belongings must be placed at the designated area of the examination hall.**
d. Write your Name, ID, Subject, Section, Instructor's name, and date in the space provided below.
e. Remember to sign the exam attendance sheet circulated by the head invigilator / proctor.
f. Remember to write the serial number of this answer booklet, found above in the right-hand corner, on attendance sheet. (if applicable)
g. Carefully read and follow all instructions given by the instructor on question paper.
h. Write the number of sheet(s) used on answer book's front page (space given on top right corner).
i. Make sure that your answer book is signed by the head invigilator/proctor.
j. If you need to leave the room for any reason, please raise your hand and take permission from head invigilator / proctor.
k. After finishing your exam, raise your hand and give your answer book to the head invigilator / proctor.

Name: Ali Muhammad Asad

Student ID: aa07190 Section: H

Subject: Computational Complexity Theory

Instructor: Abdullah Zafar

Date: 11/12/2024

Q1) $L \subseteq NL = coNL \subseteq P \subseteq NP \subseteq PSPACE = NPSPACE$ 5

1. $L \subseteq NL$: Deterministic log-space machines are just a subset of non deterministic log space machines since non-det log-space machines can simulate determ. Log-space machines → just one branch.

2. $NL = coNL$: By the Immerman Szelepcsényi theorem which proves that non-det. log-space is closed under complement.

3. $NL = coNL \subseteq P$: PATH $\in P$, & PATH is NL-Complete, then any problem in NL can be reduced to PATH, which is in P. Thus $NL \subseteq P$. Since $NL = coNL$, $coNL \subseteq P$.

-0.5
doesn't explain why it's not known to be equal

4. $P \subseteq NP$: Det. Polytime machines are a subset of non det. polytime machines since non det. Poly-time machines can simulate det. poly time machines → just one branch. P = NP is still an open question though.

5. $NP \subseteq PSPACE$: 3SAT $\in$ PSPACE & 3SAT is NP-Complete. Thus all Languages in NP can be reduced to 3SAT, which is in PSPACE. Thus all Languages in NP can be solved in PSPACE.

-0.5
doesn't explain why it's not known to be equal

6. $PSPACE = NPSPACE$: By Savitch's Theorem, PSPACE = NPSPACE.

Q3) $L \in DP \iff L_1 \in NP$ & $L_2 \in coNP$, st. $L = L_1 \cap L_2$. 4.5

To show that SAT-UNSAT [will refer it as SU for case] is DP-Complete, we have to show that $SU \in DP$ & all Languages in DP can be polytime reduced to SU.

1. $SU \in DP$: Let $L_1 \in NP$ & $L_2 \in coNP$ st. $L_1 = \{(\varphi, \varphi') \mid \varphi$ is satisfiable$\}$ $\in NP$ & $L_2 = \{(\varphi, \varphi') \mid \varphi'$ is unsatisfiable$\} \in coNP$
Then its trivial since SU is nothing but $L_1 \cap L_2$, & since $L_1 \in NP$ & $L_2 \in coNP$, $L_1 \cap L_2 \in DP$.
Thus $SU \in DP$. 2.5/3

2. SU is DP-Hard:
We reduce any Language $L \in DP$ to SU. Let $L = L_1 \cap L_2$ where $L_1 \in NP$ & $L_2 \in coNP$. Then for any input $x \in L$, we construct two boolean formulae as so:
   1. $x \in L_1 \iff \varphi$ is satisfiable
   2. $x \in L_2 \iff \varphi'$ is unsatisfiable.

Since $L = L_1 \cap L_2$, the reduction ensures that $x \in L \iff (\varphi, \varphi') \in SU$, & this reduction can easily be done in polynomial time with respect to the length of the input for any $L \in DP$
Thus SU is DP-Hard.

Therefore, SU is DP-Complete.

Instructor comments do not write in this section

Q4) (a) REACH is NL-Complete while UNREACH ∈ L. To reduce REACH to UNREACH, we need to transform a directed graph into an undirected graph such that the existence of a path from a to b in the directed graph implies the existence of a path from a to b in the undirected graph. Given that REACH is NL-Complete, it is typically harder than problems in L. Thus, the reduction is non-trivial & can't be done. If it could, that would imply that L = NL. 2

(b) Given an undirected graph G_1, simulate R as a directed graph G_1' by replacing each directed edge $(a, b) \in G_1$ with directed edges (a, b) & (b, a). A path exists b/w a & b in G_1 iff a path exists b/w a & b in G_1'. The reduction requires a single traversal over the edge list of G_1, which can be stored in $O(\log D)$ space pointers & counters by reusing the same space for all edges. Thus UNREACH $\leq_p$ REACH. 2

(c) UNREACH $\in$ L & L is a subset of P, UNREACH $\in$ P. 2

Q5). 1. BIN-S ∈ SPACE(n) ⇒ UN-S ∈ L.

A log space alg can verify the membership in UN-S by converting the unary input a^k to binary, & checking if it belongs to BIN-S, which is in SPACE(n).

↳ Given a^k, count the length k in unary, & Maintain a binary counter & increment it until it reaches k. This can be done in logspace. With k in binary, simulate SPACE(n) bounded alg for BIN-S. Since SPACE(n) allows a linear amount of space with respect to the binary representation of k, this can be computed efficiently. The overall space usage is dominated by the space needed to convert a^k to binary (log space) & simulate SPACE(n) alg for BIN-S. The alg runs in logspace & decides UN-S. Hence BIN-S ∈ SPACE(n) ⇒ UN-S ∈ L.

2. UN-S ∈ L ⇒ BIN-S ∈ SPACE(n)

A linear space alg can verify the membership in BIN-S by converting the binary input to unary & check if it belongs to UN-S.

↳ Given a binary string x, calculate its integer value k in space proportional to the lenght of x, & generate a unary string a^k for which the steps require space proportional to k, thus can be done in SPACE(n). With a^k, simulate the L bounded alg for UN-S. Since UN-S ∈ L, it uses log space. So the overall space is dominated by the space needed to convert x to a^k

which is linear, & simulate the L-alg for UN-S. The alg runs in linear space & decides BIN-S. Hence UN-S $\in$ L $\Rightarrow$ BIN-S $\in$ SPACE(n).

Thus, BIN-S $\in$ SPACE(n) $\Leftrightarrow$ UN-S $\in$ L.

Q6) To show CKT is NP-Complete,

1. CKT-SAT $\in$ NP:

We can easily make a turing machine that can verify a satisfying assignment for a given circuit. Given a circuit C, & an assignment u, evaluate C(u) & check if C(u) = 1. If C(u) = 1, accept, else reject. Evaluating a circuit C takes time proportional to the size of C, which is polynomial to the size of the input. Hence, CKT-SAT $\in$ NP.

2. CKT-SAT is NP-Hard:

We reduce SAT to CKT-SAT. Given a SAT instance $\varphi(x_1, x_2, \ldots, x_n)$ in CNF, we must decide if there exists a satisfying assign. $u \in \{0,1\}^n$ s.t $\varphi(u) = 1$. Construct a boolean circuit C to compute φ:

- represent φ as a circuit by encoding the CNF formula with gates (AND for clauses, OR for literals, NOT for negations, each literal is a node)
- transformation requires polytime since converting a CNF formula to circuit requires traversing over φ once, thus is polynomial with respect to the size of φ.

* If φ is satisfiable, then there exists an assign. u st. $\varphi(u) = 1$. Thus C outputs 1 → Yes instance.
* If φ is not satisfiable, then no assign. u exists st. $\varphi(u) = 1$. Thus C outputs 0. → No instance.

The reduction only takes polytime since constructing C is a polytime operation with respect to the

size of φ. So CKT-SAT is NP-Hard.

Therefore, CKT-SAT is NP-Complete.

⟶ Next Page.

I accidentally attempted 6.
Sorry. (:|)

Q7) a) Take a language L which is not in EXP. Now consider $L' = \{1^m \mid m \in L\}$. $L \in \text{TIME}(2^{2^{cn}})$

L' is clearly in P/poly but not in P. If it were decidable in $O(n^k)$, then we would decide L in $O((2^n)^k)$, so $L \in$ EXP. It works as so:

→ on input w of length n ≥ $\log m$, run the alg. for L' on input 1^m.

↳ This runs in time $O(n^k) = O((2^n)^k)$

It remains to ensure that L' is decidable, to that end, we need to choose some $L \notin E$ which is decidable, as that makes L' trivially decidable.

↳ given an input, if it's not of the form 1^m, reject.

↳ else answer according to whether $m \in L$.

The existence of a decidable language $L \notin$ EXP is guaranteed by the time hierarchy theorem.

(b) HALT. We know from the lectures that the unary rep. of HALT is decidable and is in P/poly, but not in P. Define $L' = \{1^m \mid m$ is the encoding of unary encoding. $\langle M, w\rangle \in$ HALT$\}$.

For L' we can provide the advice string for all unary strings upto length n, encoding whether each unary string 1^m corresponds to an instance $\langle M, w\rangle \in$ HALT.

The advice string is polynomial in size.

L' is in P/poly for each input length.

L' is not in P as if it were decidable in P, it would solve the Halting Problem. which contradicts undecidability of HALT.

(c) By the time heirarchy theorem, there exists a language $L \notin EXP$ which is greater than EXP. Then consider that language L, for which we can't build a proper encoding to solve in P/poly. Since $P \subseteq P/poly$ & it is not in P/poly then it is not in P.

↓

Theorem 6-21 of the Book: For every $n \geq 1$, there exists a function $f: \{0,1\}^n$ that cannot be computed by a circuit C of size $2^n/10n$.

↳ Make a language L of these. Clearly not decidable, not in P/poly, hence not in P.

[*Note: I accidentally attempted all 7 (not one-off) kindly accept the best 5 of 6 please really need an A in this course PhD sir. All the best! Was fun taking a course with you finally].

Q2) (a) 1. If $\varphi \in$ 2CNF unsatisfiable $\Rightarrow$ paths from x to $\neg x$ & $\neg x$ to x in $G(\varphi)$

Construct a graph G from φ by adding each literal as a node & for all clauses, we add an edge between (a, b) in G. Then if φ is unsatisfiable, that implies that we can trace a path from arbitrary literals (u, v), tracing their connections & towards each other from there edges. Eventually we would land up at 2 nodes x & $\neg x$ s.t. they have a path between them, if they did not, we could easily trace a path from u to v, traversing all the edges, so that would lead to a contradiction that x & $\neg x$ are not at the same time

2. Paths from x to $\neg x$ & $\neg x$ to x in $G(\varphi)$ imply $\varphi \in$ 2CNF is unsatisfiable

This is trivial since having a direct edge & path from x to $\neg x$ & $\neg x$ to x would imply both would have to be true simultaneously which is a contradiction.