22.5

[Exam 1] Name: Ali Muhammad Asad ID: 07190

Fall 2024: CS 313: Computational Complexity Theory

Due: 12:45 pm, Monday, September 30, 2024. Total Marks: 24

This exam copy contains 3 pages, including this one.

Question 1 [12 points]

For each part, provide brief explanations and/or proofs.

3

1. We have defined a relation $\leq_p$ among languages. This relation is reflexive (i.e. $L \leq_p L'$ for all languages) and transitive (i.e. if $L \leq_p L'$ and $L' \leq_p L''$ then $L \leq_p L''$). Why is it not symmetric, namely, why is it that $L \leq_p L'$ need not imply $L' \leq_p L$?

Consider two languages $L = \emptyset$ & $L' = \{a\}$ over a non empty language Σ and $a \in \Sigma$.

1. $L \leq_p L'$: We transform any input string into "aa" which takes poly time. Since L is empty, & has no strings, it always gives "No". Thus the reduction maps any input to "aa" which determines yes or no. Thus $L \leq_p L'$.
2. $L' \not\leq_p L$: Trivial since $L = \emptyset$ has no strings so we cannot construct a reduction such that 'a' in L' can be mapped onto nothing. Thus $L' \not\leq_p L$. Hence it is not symmetric — disproved symmetry.

1.5

2. Show that NP is closed under union.

Consider 2 arbitrary languages L_1 & L_2 exist in NP having Non-Det TMs M_1 & M_2. We can construct a TM M for $L_1 \cup L_2$ as so:

how would the step 1 and 2 be handled sequentially?

M_2 "On input w:
1. Run M_1 on w. If M_1 accepts, accept.
2. Run M_2 on w. If M_2 accepts, accept.
3. Reject".

Clearly the largest branch ~~would~~ for any input w ~~would~~ of length l would be $O(n^{\max\{k, l\}})$, when k & l are times taken by M_1 & M_2. Thus, $L_1 \cup L_2$ is closed under NP.

3

3. Why is every NP-Hard language not decidable by a Turing Machine?

There are some NP-Hard problems not decidable by a TM, such as Halting Problem. NP-Hard refers to the difficulty of solving a problem while decidability refers to solving a problem.
For undecidable problems in NP-Hard like Halting Problem, no TM can always decide the answer for all inputs which makes ~~them~~ them undecidable since there can be infinite inputs of infinite lengths & no algorithm can handle that in finite time.
Thus, some NP-Hard Problems are undecidable by a TM for all possible

3 4. Show that if $\mathbf{P} = \mathbf{NP}$ then $\mathbf{NP} \subset \mathbf{EXP}$, where $\subset$ denotes the proper subset relation.

By definition, ~~Exp~~ EXP includes all problems that can be solved by a deterministic TM in exponential time $2^{p(n)}$ where $p(n)$ is a polynomial in input size n.

If P=NP, then every problem in NP can be solved in poly time by a deterministic TM, which is faster than EXP time. Since P ⊂ EXP, we have NP ⊂ P ⊂ EXP as P=NP. Thus, NP ⊂ EXP as all problems in NP can also be solved in EXP time.

everyone receives full credit for this

Question 2 [6 points]

In the EXACTLY ONE 3SAT problem, we are given a 3CNF formula ϕ and need to decide if there exists a satisfying assignment such that every clause of ϕ has exactly one true literal. Show that EXACTLY ONE 3SAT is **NP-Complete**.

Exactly One 3SAT ⇒ EO3SAT (aaive representing)

3 We show the EO3SAT is in NP & is NP-Hard.

We can easily build a verifier 'V' that given a certificate c → truth assignments to EO3SAT instance can verify whether the given truth assignments satisfy the conditions of EO3SAT or not. This can be done in poly time as all clauses or literals would be checked. Hence V runs in poly time. Thus EO3SAT ∈ NP.

3 Now we reduce 3SAT to EO3SAT ⇒ $3SAT \leq_p EO3SAT$.

Given a boolean formula φ of 3SAT, assume it consists of clauses $C_1, C_2, \ldots, C_m$, where each clause has exactly 3 literals. We construct a new boolean formula φ' of EO3SAT such that φ' has a satisfying assignment iff the original formula φ has a satisfying assignment.

* For each clause $C_i = (x_1 \vee x_2 \vee x_3)$ in φ, we introduce 2 new clauses for φ'; y_i & z_i. The idea is to ensure that exactly one of the original literals in the clause is true. PTO

Thus we do this by adding 2 new clauses and variables as so:

- For each clause $C_i = (x_1 \vee x_2 \vee x_3)$ construct the following EO3SAT clauses:
    1. $(x_1 \vee x_2 \vee y_i)$
    2. $(\neg y_i \vee x_3 \vee z_i)$

The variables y_i ensures that exactly one of the literals in the original clause is true, if neither or both are true, y_i controls which are allowed to be true. Similarly, the second clause ensures that the third literal x_3 behaves in a similar manner via z_i.

* If a clause in 3SAT has at least one true Literal, the transformation ensures that exactly one litural will be true in each new clause of EO3SAT.

* Conversely, if there is a solution to EO3SAT, it corresponds to a valid solution to the original 3SAT formula φ as it would also have at least one true literal in every clause.

This reduction can be done in polytime as for each clause in 3SAT, we only add a small number of variabl & clauses which can be done in poly time.

Since EO3SAT $\in$ NP, & 3SAT $\leq_p$ EO3SAT, thus we have shown that EO3SAT is NP-Complete as 3SAT is also NP-Complete.

6 **Question 3** [6 points]

Recall that normally we assume that numbers are represented as string using the *binary* basis. That is, a number n is represented by the sequence $x_0, x_1, ..., x_{\log n}$ such that $n = \sum_{i=0}^{\log n} x_i 2^i$. However, we could have used other encoding schemes. If $n \in \mathbb{N}$ and $b \geq 2$, then the *representation of n in base b*, denoted by $\llcorner n \lrcorner_b$ is obtained as follows: First, represent n as a sequence of digits in $\{0, ..., b-1\}$, and then replace each digit $d \in \{0, ..., d-1\}$ by its binary representation.

Show that choosing a different base of representation will make no difference to the class **P**. That is, show that for every subset S of the natural numbers, if we define $L_S^b = \{\llcorner n \lrcorner_b : n \in S\}$, then for every $b \geq 2, L \in$ **P** iff $L_S^b \in$ **P**.

Number $n \Rightarrow x_0, x_1, ---, x_{\log n}$. st. $n = \sum_{i=0}^{n} x_i 2^i$

$n \in \mathbb{N}$ & $b \geq 2$, $\llcorner n \lrcorner_b \Rightarrow$ first $n \Rightarrow \{0, ---, b-1\}$ & replace each digit $d \in \{0, ---, d-1\}$ by binary representation.

Choosing a different base makes no difference to P.

$L_S^b = \{\llcorner n \lrcorner_b : n \in S\}$ for every subset S of natural numbers, then $\forall\ b \geq 2$, $L \in P$ iff $L_S^2 \in P$.

We can prove this by showing that we can convert from one base to the other in polynomial time. Then if the conversion takes poly time & the problem exists in P, then the total time taken after conversion is still in P since conversion took polynomial time.

In binary, $x_0 2^0 + x_1 2^1 + x_2 2^2 + --- + x_{\log n} 2^{\log n}$

n is represented as ↑!

This takes $\log$ time to compute, ~~since we have to~~ , more specifically $O(\log n)$ considering addition takes $O(1)$.

Now in base b, first we represent n as $\{0, ---, b-1\}$, then this should take $O(b)$ time when $b \geq 2$.

Then we need to replace each digit $d \in \{0, ---, d-1\}$ by its binary representation. Since one number takes

$O(\log n)$ to convert into binary, then

b numbers ~~arbitrary length~~

take $O(b \log b)$ time to convert into binary.

Considering $b = n$, $O(n \log n)$.

Since $O(n \log n)$ is upper bounded by $O(n^{c})$,

then the conversion can happen in polynomial time or less, & since the problem already exists in P, it can also be solved in polynomial time.

Hence even after conversion, the problem is still in P.

Thus,

1) If $b \geq 2$, $L \in P$ then $L_s^b \in P$.

The conversion from ~~to binary can~~ n to b can be done in P time, & the problem already exists in P, thus $L_s^b \in P$.

2) If $L_s^b \in P$, $b \geq 2$, then $L \in P$.

Conversely, we can use the same steps to change our base b back to n or binary which would again take P time. Thus, L can also be solved in P time since L_s^b could be solved in P time, & conversion takes P time.

Hence proved that the conversion of base doesn't matter.