Assignment 4

Pranav Wadhwa

April 10, 2022

1. Answer each part TRUE or FALSE

   a. $2^{\log^2 n} \in \mathcal{O}(n^k)$

   b. $n! \in 2^{\mathcal{O}(n \log n)}$

   c. $3^n \in \mathcal{O}(2^{n \log n})$

   d. $n^k \in o(2^{\log n})$

   e. $2^n \in o(n!)$

   f. $\frac{1}{n} \in o(1)$

Solution:

a. $2^{\log^2 n} \in \mathcal{O}(n^k)$

   True
   Proof: $2^{\log^2 n} = n$ and $n \in \mathcal{O}(n^k)$

b. $n! \in 2^{\mathcal{O}(n \log n)}$

   True

c. $3^n \in \mathcal{O}(2^{n \log n})$

   False
   As 3^n grows much faster than $2^{n \log n}$

d. $n^k \in o(2^{\log n})$ True

e. $2^n \in o(n!)$ True

f. $\frac{1}{n} \in o(1)$

   True as $1/n$ is strictly smaller than 1 after $n > 1$

2. Show that P is closed under union, complement, concatenation, and star.

Solution: Let M_1 and M_2 be 2 turing machines which decide languages L_1 and L_2 in polynomial time.

1. union
   Construct a turing machine M_U which decides the union of L_1 and L_2.

M_U on input w:
1. Run M_1 on w. If accept return accept.
2. Run M_w on w. If accept return accept.
3. If both reject then return reject.

Since both M_1 and M_2 decide in polynomial time then the union will also decide in polynomial time.

2. complement
Construct a deterministic turing machine $M_{\bar{C}}$ which decies on the complement of L_1.

$M_{\bar{C}}$ on input w:
1. Run M_1 on input w
2. If M_1 accept then return reject, else return accept.

Since M_1 runs in polynomial time then $M_{\bar{C}}$ also runs in polynomial time.

3. concatenation
Construct a determisnistic TM M_C which decides the concatenation of languages L_1 and L_2 in polynomial time.

M_C on input w: 1. Guess the partition of input w into 2 string such as $w = xy$.
2. Run M_1 on input x. If not accept return reject.
3. Run M_2 on input y. If not accept return reject.
4. If both accept then return accept

Since both M_1 and M_2 run in polynomial time then M_C also runs in polynomial time.

4. star
Construct a determisnistic TM M_S which decides on L_1^*. This is similar to concatenation where we divide the input w into several sub strings.

M_S on input w:
1. Guess the partition $w = x_1 x_2 \ldots x_n$.
2. If M_1 accepts all of x_n then return accept else return reject.

Since M_1 runs in polynomial time then M_S also runs in polynomial time.

3. Show that NP is closed under union, concatenation, and star

Solution: The solution is similar to Q2, instead of using a deterministic TM we would use a non-deterministic TM which would go through every branch to check if the input is accept by the TM.

4. We normally assume natural numbers are represented in binary, such that a number $n \in \mathcal{N}$ is represented by the string $b_{\lfloor \log n \rfloor} b_{\lfloor \log n \rfloor - 1} \ldots b_0, b_i \in \{0, 1\}$, and $n = \Sigma_{i=0}^{\lfloor \log n \rfloor} b_i 2^i$. We could also write a number in unary, where a number $n \in \mathcal{N}$ is represented by n consecutive 1s. The problem of factoring a number in binary is not known to be in P, but what if the number is given in unary? Prove your answer.

Solution:

Lets look at a naive algorithm of finding factors of a input n.

for i = 1 to $\sqrt{n}$:
  if (n % i == a):
    add i as a factor

If the numbers are represented in binary then the length of the input is $k = \log_2 n$ bits which makes the time complexity to be $\mathcal{O}(\sqrt{2^k})$ therby making the problem to be exponential.

In the case of unary the lenth of the input is equal to n bits which makes the time complexity to be linear. Therefor this algorithm is in P.

5. Suppose the number k and the graph G are given, and we want to know if there exists a subset S of size k from the vertices of G such that there is no edge between them in G. Prove that this problem is NP-Complete.

Solution:

NO-EDGE = $\{\langle G, k \rangle$: G is an undirected graph and has a subset S of size k with no edges between them $\}$

Show that this problem is NP, a certificate for this problem is given as follows:

Proof: certificate is a subset of vertices c
VG, k, c:
  verify that G has k verticies.
  verify that there are no edges between the nodes in c in graph G
  accept $\leftrightarrow$ both pass

Show that this problem in NP-Complete

Proof: We use reduction from $CLIQUE \leq_p NO-EDGE$

Given a graph G with k clique. We can construst a graph G' with V' vertices and E' edges.

  We find the complement of the graph G to produce G'.

  Traverse through G and where there is an edge between 2 vertices remove it and where there is no edge we add an edge between the pair of verticles. We get the complement graph G' the edges and k is the number of verticles that have no edges between them.

This runs in poly time.

6. Show the following languge is NP-Complete:

$$DOUBLE-SAT = \{\langle \phi \rangle | \phi \text{ has at least 2 satisfying assignments } \}$$

Solution:

Showing DOUBLE-SAT is NP is easy; a certificate is simply a set of values of the variables in ϕ which make the boolean expression evaluate to true.

We will use poly time reduction to prove that DOUBLE-SAT is NP-Complete by showing $SAT \leq_p DOUBLE-SAT$

Proof:
We create a non-deterministic TM M such that it computes a polytime reduction of $\phi \rightarrow \phi'$
M on input $\langle \phi \rangle$ which a Boolean formula such that ϕ has variables $c_1, c_2, \ldots, c_k$.

1. Create $\phi' = \phi \wedge (x \vee \bar{x})$ with x as a new variable.
2. Run ϕ on SAT.
3. If SAT accepts then ϕ' has at least 2 satisfying assignment. If $\phi \notin SAT$ then $\phi' \notin DOUBLE-SAT$

We showed in TM M that $\phi \in SAT$ iff $f(\phi) \in DOUBLE-SAT$ therefore DOUBLE-SAT is NP-Complete.

7. Let S be a set and let C be a collection of subsets of S. A set $S' \subseteq S$ is called a set hitting set for C if every subset in C contains at least an element in S'. Let

$$HITSET = \{\langle C, k \rangle | C \text{ has a hitting set of size } k\}$$

Prove that $HITSET$ is NP-Complete.

Solution: We can show that HITSET is in NP

Proof: certificate is a certificate is simply a the set S' where $S' \cap S_i \neq \emptyset, \forall S_i \in C$.

$V(C, k, c)$:
verify that c is in C
verify that each set in C has $S_i \cap c \neq \emptyset$
accept $\leftrightarrow$ both pass

runs in polytime

Now show that $VERTEX-COVER \leq_p HITSET$

8. a. Prove that NP = coNP iff there is an NP-Complete problem in coNP.
b. Show that if coNP $\neq$ NP then P $\neq$ NP.

Solution:

a. NP = coNP iff there is an NP-Complete problem in coNP

Proof:
Part 1: if there is an NP-Complete problem in $coNP$ then $NP = coNP$

Let L be an language that is NP-Complete and $L \in coNP$. Let L_2 be a problem in NP then there is a polytime reduction f from $L_2 \rightarrow L$ since L is NP-Complete. Now since $L_2 \rightarrow L$ exists this shows that L_2 is also in $coNP$. So $NP = coNP$

Part 2: if $NP = coNP$ then there is an NP-complete in coNP

This statement is true itself since $NP = coNP$ there would be problem which are NP-complete in $coNP$

b. if coNP $\neq$ NP then $P \neq NP$

Proof:

We can prove this by showing the contrapositive.
If $P = NP$ then coNP=NP

Let L be a problem which in in P. Since $P = NP$ then L is also in NP. We know that P is closed under complement then NP would also be closed under complement. Therefore $\bar{L}$ is also in P and NP. This shows that $NP = coNP$.

9. Show that P is closed under homomorphism iff P = NP

10. Let $CNF_k = \{\langle\phi\rangle | \phi$ is a satisfiable cnf-formula where each variable appears in at most k places $\}$.

   a. Show that $CNF_2 \in P$

   b. Show that CNF_3 is NP-complete

Solution:

a. Show that $CNF_2 \in P$

We give a way to fine out if a variable appears more than twice in a binary formula. We say it case by case:

Let x be a variables in ϕ
1. If x appeas only once as x or $\bar{x}$ then we can satisfy the clause it appears in and move onto next variable.
2. If x or $\bar{x}$ appears more 2 times then ϕ is not satisfiable
3. If x or $\bar{x}$ appears in the same clause then ϕ is not satisfiable.
4. If x or $\bar{x}$ appears in different clauses such as $(x \vee \ldots) \wedge (\bar{x} \vee \ldots)$ then it satisfiable only if other variables in the clauses have an satisfying assignment as both x and $\bar{x}$ cannot be satisfiable at the same time.

We use this method and easily see that it reduces the boolean formula each time therefore it runs in polynomial time and is in P.

b. Show that CNF_3 is NP-Complete.

Check that $CNF_3 \in NP$

Proof: certificate is a satisfying argument
V(ϕ, c):
verfiy each variables appeas at most 3 times
evaluate $\phi(c)$
accept $\leftrightarrow$ both pass

Prove that $3 - SAT \leq_p CNF_3$