

### 1. Graph Theory

Given an edge-coloring of a multigraph, we say that color  $j$  is *missing* at a vertex  $v$  if no edge incident with  $v$  is colored  $j$ . Let  $G$  be a multigraph and  $v_1v_2 \dots v_n$  ( $n \geq 2$ ) a path in  $G$ . Suppose there is an edge-coloring  $c : E(G - v_1v_2) \rightarrow \{1, \dots, k\}$  of  $G - v_1v_2$ , where  $k \geq \Delta(G) + 1$ , such that

- (1) for each  $2 \leq i \leq n - 1$  there exists  $1 \leq s \leq i - 1$  such that  $c(v_iv_{i+1})$  is missing at  $v_s$ , and
- (2) there exist  $1 \leq i < n$  and  $j \in \{1, \dots, k\}$  such that  $j$  is missing at both  $v_i$  and  $v_n$ .

Show that  $G$  is  $k$ -edge-colorable.

**Solution:** Let  $c$  and  $v_1v_2 \dots v_n$  be chosen, subject to the stated properties, such that

- (i)  $n$  is minimal, and
- (ii) subject to (i),  $n - i$  is minimal.

Then we have

- (iii) no color is missing at two distinct elements of  $\{v_1, \dots, v_{n-1}\}$ .

If  $n = 2$  then clearly a  $k$ -edge-coloring of  $G$  can be obtained from  $c$  by assigning the color  $j$  to  $v_1v_2$ . So we may assume  $n \geq 3$ .

If  $i = n - 1$ , then we modify  $c$  to the edge-coloring  $c'$  of  $G - v_1v_2$  by simply changing the color of  $v_{n-1}v_n$  to  $j$ . Clearly,  $c'$  and  $v_1 \dots v_{n-1}$  satisfy (1). By (1),  $c(v_{n-1}v_n)$  is missing at  $v_s$  for some  $1 \leq s \leq n - 2$ ; so in the edge-coloring  $c'$ , the color  $c(v_{n-1}v_n)$  is missing at both  $v_{n-1}$  and  $v_s$ , and  $c'$  and  $v_1 \dots v_{n-1}$  satisfy (2). This contradicts the choice of  $c$  and  $v_1 \dots v_n$  (specifically, (i) above).

Therefore,  $i \leq n - 2$ . We now consider  $v_{i+1}$ . Since  $k \geq \Delta(G) + 1$ , there exist a color  $\ell \in \{1, \dots, k\}$  that is missing at  $v_{i+1}$ . By (iii),  $\ell \neq j$ . Consider the subgraph of  $G$  induced by the edges with color  $j$  or  $\ell$ , and let  $C$  denote its component containing  $v_{i+1}$ . Note that  $C$  is a path.

Let  $c'$  denote the edge-coloring of  $G - v_1v_2$  obtained from  $c$  by swapping the colors along  $C$ . By (iii),  $C$  does not end in  $\{v_1, \dots, v_{i-1}\}$ ; and by (1) and (iii),  $C$  is edge-disjoint from  $v_1 \dots v_{i+1}$ . So  $c'$  and  $v_1 \dots v_{i+1}$  satisfy (1).

Suppose  $v_i \notin C$ . Then in the edge-coloring  $c'$ ,  $j$  is missing at both  $v_i$  and  $v_{i+1}$ ; so  $c'$  and  $v_1 \dots v_{i+1}$  satisfy (2), which contradicts (i).

Now assume  $v_i \in C$ . Since any edge of  $v_{i+1} \dots v_n$  with different colors in  $c$  and  $c'$  must have color  $j$  or  $\ell$  (which are missing at  $v_i$  or  $v_{i+1}$ ), we see that  $c'$  and  $v_1 \dots v_n$  satisfy (1). Moreover, the color  $j$  is missing at both  $v_{i+1}$  and  $v_n$  in the edge-coloring  $c'$ . So  $c'$  and  $v_1 \dots v_n$  satisfy (2), which contradicts (ii).

## 2. Probability

Assume that we have a Bernoulli variable  $X$  so that  $P(X = 1) = 0.6$  and  $P(X = 0) = 0.4$ . Assume that  $X, X_1, X_2, \dots$  are i.i.d. random variables.

(a) Find the smallest  $c > 0$ , so that

$$P(X_1 + X_2 + \dots + X_n \geq 0.8n) \leq c^n$$

for all  $n$ .

(b) Once you have found  $c$ , explain why  $c$  is the smallest constant satisfying the above equality for all  $n$ .

**Solution:** (a) Assume that  $0.8n$  is an integer. Then the probability  $P(X_1 + X_2 + \dots + X_n \geq 0.8n)$  is of the same order as the probability  $P(X_1 + X_2 + \dots + X_n = 0.8n)$ . But this is the probability for a binomial and hence

$$P(X_1 + X_2 + \dots + X_n = 0.8n) = \binom{n}{0.8n} 0.6^{0.8n} 0.4^{0.2n}. \quad (1)$$

We have that

$$\binom{n}{0.8n} 0.8^{0.8n} 0.2^{0.2n} \leq 1$$

and hence

$$\binom{n}{0.8n} \leq \left( \frac{1}{0.8^{0.8} \cdot 0.2^{0.2}} \right)^n$$

(One could also have used Sterling here). Using the last inequality above with inequality (1) we find

$$P(X_1 + X_2 + \dots + X_n = 0.8n) \leq \left( \frac{0.6^{0.8} 0.4^{0.2}}{0.8^{0.8} \cdot 0.2^{0.2}} \right)^n \quad (2)$$

The above inequality turns out to give the right order of magnitude so that the constant is

$$c = \frac{0.6^{0.8} 0.4^{0.2}}{0.8^{0.8} \cdot 0.2^{0.2}}.$$

(b) One approach is the theorem which says that  $c$  is given by the solution to the following minimizing problem:

$$\min_{t \geq 0} E[e^{(X-0.8)t}] = \min_{t \geq 0} (E[0.6e^{0.2t}] + 0.4e^{-0.8t})$$

To solve the above take the derivative and equate to zero. One could also use Sterling or a subtle argument about the large deviation rate being convex and the fact that there are only a finite a polynomial number of types.

### 3. Analysis of Algorithms

Consider a matrix with numerical data where each entry is rational, but each row and column sum is an integer. Prove that you can “round off” this matrix, rounding each entry to the next integer above or below, without changing the row or column sums.

**Solution:** Subtract the unique integer (positive or negative) from each entry so as to make the remainder a rational number in  $[0, 1)$ , simultaneously subtracting the same amount from the corresponding row and column sums. A solution to this new problem gives us a solution to the original problem.

The key to solving the remainder of the problem is thinking in terms of network flows. We consider the reduced problem where all entries are non-negative and less than one. Let’s make a bipartite graph where edge  $(i, j)$  is present if the corresponding matrix entry is non-zero, and let the entry be the capacity of the edge. Now connect all of the vertices on one side of the bipartition to a “source” vertex, taking the row sums as the capacities of these edges, and connect all of the vertices on the other side of the bipartition to a “sink” vertex, taking the column sums as the capacities of these edges. Because of the way we constructed this graph, we know that there is a valid flow from the source to the sink that is equal to the sum of the entries in the reduced matrix. Let us now raise the capacities on the edges that are not connected to the source or the sink to one. This can only increase the max flow. Moreover, if we use one of the standard max flow algorithms, we can find an integral solution to the max flow in which each edge not incident to the source or sink has flow 0 or 1. This in turn gives us a solution to the reduced matrix problem where each entry is rounded up or down, and thus a solution to the original matrix rounding problem as well.

## 4. Combinatorial Optimization

Let  $P = \{x : Ax \leq b\}$  be a rational polyhedron in  $R^n$  and let  $w$  and  $c$  be rational vectors such that both

$$\max \{w^T x : x \in P\} \tag{3}$$

and

$$\max \{c^T x : x \in P\} \tag{4}$$

have optimal solutions.

Give a polynomial-time procedure for finding a linear system  $Bx \leq d$  that defines the set of vectors  $\bar{x}$  such that  $\bar{x}$  is optimal for (3) and  $\bar{x}$  is optimal for (4). The description of the system  $Bx \leq d$  should not involve  $w$  or  $c$ .

**Solution.** The set of optimal solutions to (3) is a face  $F$  of  $P$ . Using complementary slackness, we can write  $F = \{x : A'x = b', Ax \leq b\}$  where  $A'x \leq b'$  consists of the inequalities in  $Ax \leq b$  corresponding to non-zero variables in an optimal solution to the dual of (3). Similarly we can find a linear system  $A''x = b'', Ax \leq b$  describing the face of optimal solutions for (4). The combined system  $A'x = b', A''x = b'', Ax \leq b$  defines the set of common optimal solutions to (3) and (4).

### 5. Theory of Linear Inequalities

Say that a finite set  $H$  of rational vectors in  $R^n$  is a *super Hilbert basis* if for each  $J \subseteq \{e_1, \dots, e_n\}$  the set  $H \cup J$  is a Hilbert basis (where  $e_i$  denotes the  $i$ th unit vector in  $R^n$ ). Let  $P = \{x : Ax \leq b\}$  be a rational polyhedron in  $R^n$ . Show that  $Ax \leq b, x \leq u$  is totally dual integral for every rational vector  $u$  if and only if for every face  $F$  of  $P$  the set of active rows in  $Ax \leq b$  is a super Hilbert basis.

**Solution** We know (Schrijver, Theorem 22.5) that a rational system  $Ax \leq b$  is TDI if and only if for each face  $F$  of  $P = \{x : Ax \leq b\}$  the active rows in  $Ax \leq b$  form a Hilbert basis.

Suppose that  $Ax \leq b, x \leq u$  is TDI for every rational vector  $u$ . Let  $F$  be a face of  $P = \{x : Ax \leq b\}$  and let  $\bar{x} \in F$  be a vector such that an inequality  $a_i x \leq b_i$  in  $Ax \leq b$  is active for  $F$  if and only if  $a_i \bar{x} = b_i$ . Let  $H$  denote the active rows for  $F$  in  $Ax \leq b$  and let  $J \subseteq \{e_1, \dots, e_n\}$ . A vector  $u$  can be chosen such that the set of active rows of  $\{x\}$  in  $Ax \leq b, x \leq u$  is  $H \cup J$ . Since  $Ax \leq b, x \leq u$  is TDI, we know that  $H \cup J$  is a Hilbert basis.

Now suppose that for every face  $F$  of  $P$  the set of active rows in  $Ax \leq b$  is a super Hilbert basis. Let  $u$  be a rational vector and let  $Q$  be a face of  $\{x : Ax \leq b, x \leq u\}$ . The set of rows in  $Ax \leq b$  that are active for  $Q$  form a super Hilbert basis, and thus the active rows for  $Q$  in  $Ax \leq b, x \leq u$  are a Hilbert basis. It follows that  $Ax \leq b, x \leq u$  is TDI.

6. Algebra

Prove or disprove each of the following statements.

(a) Let  $p$  be a prime and let  $G$  be the group  $(\mathbb{Z}/p\mathbb{Z})^n$ . Let  $P$  be a  $p$ -Sylow subgroup of the group of automorphisms  $\text{Aut}(G)$ . Then, there exist sub-groups  $\{e\} = P_{n-1} \subset P_{n-2} \cdots \subset P_0 = P$  such that for each  $i, 1 \leq i \leq n-1$ ,  $P_i$  is a normal subgroup of  $P_{i-1}$  and the quotient  $P_{i-1}/P_i$  is abelian.

(b) Let  $G$  be a group,  $H \subset G$  a subgroup and  $g$  an element of  $G$  such that  $gHg^{-1} \subset H$ . Then,  $gHg^{-1} = H$ .

(c) Let  $k$  be a field with  $\text{char}(k) = p$  and  $E$  a finite algebraic extension of  $k$ . Then the number of distinct intermediate fields  $F, k \subset F \subset E$ , is finite.

**Solution:** (a) First note that since  $G$  is an  $n$ -dimensional vector space over the field  $\mathbb{Z}/p\mathbb{Z}$ ,  $\text{Aut}(G) \cong \text{Gl}(n, \mathbb{Z}/p\mathbb{Z})$ , and the order of  $\text{Aut}(Z)$  is  $(p^n - 1)(p^n - p) \cdots (p^n - p^{n-1})$ . The order of a  $p$ -Sylow subgroup of  $\text{Aut}(G)$  is  $p^{(n-1)+(n-2)+\cdots+1}$ . By a cardinality argument we see that the subgroup  $P$  is isomorphic to the subgroup  $T \subset \text{Gl}(n, \mathbb{Z}/p\mathbb{Z})$  of upper triangular matrices with 1's on the diagonal. It is easy to see that  $T$  is a subgroup since it is closed under multiplication and  $t \in T$  can be written as  $t = 1 + n$  where  $n$  is a nilpotent matrix. Thus,  $t^{-1} = 1 + n + n^2 + \cdots$  is in  $T$ .

For  $r = 1, \dots, n-1$  let  $T^{(r)} \subset T$  be the subgroup consisting of upper triangular matrices with 1's on the diagonal whose first  $r$  super-diagonals are zero.

Then,  $T^{(r+1)}$  is a normal subgroup of  $T^{(r)}$  since it is the kernel of the homomorphism sending  $T^{(r)}$  to its  $r$ -th super-diagonal considered as the additive group  $(\mathbb{Z}/p\mathbb{Z})^{n-r}$  which is abelian. Also,  $T^{(n-1)} = 1$ .

(b) The statement is false. A counter-example is as follows. Let  $G$  be the group with two generators  $a, b$  with one relation  $aba^{-1} = b^2$ . Let  $H$  be the subgroup of  $G$  generated by  $b$ . Then,  $aHa^{-1}$  is the subgroup generated by  $b^2$ . Clearly,  $b \notin aHa^{-1}$  and hence  $aHa^{-1} \subset H$  but  $aHa^{-1} \neq H$ .

(c) The statement is false. A counter-example is as follows. Let  $p$  be an odd prime.  $E = \mathbb{Z}/p\mathbb{Z}(X, Y)$  and let  $k$  be the subfield  $\mathbb{Z}/p\mathbb{Z}(X^p, Y^p)$ . Then,  $k \subset E$  are both infinite fields of characteristic  $p$ , and it is easy to verify that  $[E : k] = p^2$ . For  $c \in k$ , let  $F_c = \mathbb{Z}/p\mathbb{Z}(X + cY)$ . Then, each  $F_c$  is an intermediate field. Since,  $(X + cY)^p = X^p + c^pY^p \in k$ , we have that  $[F_c : k] = p$ . We claim that for  $c \neq d, c, d \in k$ ,  $F_c \neq F_d$ . Suppose that  $F_c = F_d = F$ . Then, since both  $X + cY$  and  $X + dY$  are in  $F$  by taking their sum and their difference we obtain that  $X$  and  $Y$  are in  $F$  too, and hence  $F = E$ . But this is impossible since  $[E : k] = p^2$  but  $[F : k] = p$ . Since  $k$  is infinite we have shown that there exists infinitely many distinct sub-extensions.

## 7. Randomized Algorithms

Consider the following random walk on the states  $\Omega = \{0, 1, 2, \dots, n-1\}$ . From state  $i$ :

- move to state  $i + 1 \bmod n$  with probability  $1/2$ ,
- move to state  $0$  with probability  $1/2$ .

More precisely, it is a Markov chain with transition matrix  $P$  defined as follows. For  $0 \leq i < n-1$ :

$$P_{i,i+1} = P_{i,0} = 1/2.$$

Also,  $P_{n-1,0} = 1$ .

Give a coupling argument to upper bound the mixing time of the chain (within a constant factor of optimal is fine).

Recall the mixing time is defined to be

$$T = \max_{x \in \Omega} T_x$$

where

$$T_x = \min \{t : d_{\text{TV}}(P^t(x, \cdot), \pi) \leq 1/4\},$$

where  $d_{\text{TV}}$  is the variation distance between the two distributions and  $\pi$  is the stationary distribution of the chain.

**Solution.** To analyze the mixing time we define a coupling. For two copies of the chain  $(X_t)$  and  $(Y_t)$ , the transitions  $X_t \rightarrow X_{t+1}$  and  $Y_t \rightarrow Y_{t+1}$  are coupled as follows. If  $X_t = Y_t$ , then we first randomly choose the transition  $X_t \rightarrow X_{t+1}$  and we set  $Y_t = X_t$ . If  $X_t \neq Y_t$ , let  $i = X_t$  and  $j = Y_t$ . Then, with probability  $1/2$  we set  $X_{t+1} = Y_{t+1} = 0$  and with probability  $1/2$  we set  $X_{t+1} = i + 1 \bmod n$  and  $Y_{t+1} = j + 1 \bmod n$ . It is clear that the transitions  $X_t \rightarrow X_{t+1}$  and  $Y_t \rightarrow Y_{t+1}$  follow the transition matrix.

Note, if  $X_t = Y_t$  then  $X_{t+1} = Y_{t+1}$ , and if  $X_t \neq Y_t$  then with probability  $1/2$  we have  $X_{t+1} = Y_{t+1}$ . Hence, for  $X_0 \neq Y_0$ ,

$$\Pr(X_4 \neq Y_4) = (1 - 1/2)^4 \leq \exp(-2) < 1/4.$$

Therefore, the mixing time is  $O(1)$ .

## 7. Approximation Algorithms

Consider the general uncapacitated facility location problem in which the connection costs are not required to satisfy the triangle inequality:

Let  $G$  be a bipartite graph with bipartition  $(F, C)$ , where  $F$  is the set of *facilities* and  $C$  is the set of *cities*. Let  $f_i$  be the cost of opening facility  $i$ , and  $c_{ij}$  be the cost of connecting city  $j$  to (opened) facility  $i$ . The problem is to find a subset  $I \subseteq F$  of facilities that should be opened, and a function  $\phi: C \rightarrow I$  assigning cities to open facilities in such a way that the total cost of opening facilities and connecting cities to open facilities is minimized.

Give a reduction from the set cover problem to show that approximating this problem is as hard as approximating set cover and therefore cannot be done better than  $O(\log n)$  factor unless  $\mathbf{NP} \subseteq \tilde{\mathbf{P}}$ . Also, give an  $O(\log n)$  factor algorithm for this problem.

**Solution.** Reduction. Encode elements of set cover instance as cities and sets as facilities. All connection costs are zero and opening cost of a facility is cost of set. The rest is obvious.

Algorithm. For a facility  $f$  and set  $S$  of cities, define the cost effectiveness of a covering  $S$  with  $f$  to be: (cost of opening  $f$  + connection costs of connecting cities in  $S$  to  $f$ )/ $|S|$ . The greedy algorithm covers using most cost effective combination in each iteration till all cities are covered. The argument for  $\log n$  factor is the same as for set cover.

## 7. Computational Complexity

Consider the FACTORING problem: Given a natural number  $N$ , express  $N$  as a product of its prime factors. To date no polynomial-time algorithm for FACTORING is known, and the conjectured hardness of FACTORING has been the basis of several public-key cryptosystems.

**1.** Prove that if  $\mathbf{P} = \mathbf{NP} \cap \mathbf{coNP}$ , then FACTORING can be solved by a polynomial-time algorithm. You may use a polynomial-time algorithm for deciding whether a given integer is prime.

**2.** Prove that unless  $\mathbf{NP} = \mathbf{coNP}$ , FACTORING is *not* **NP-Hard** under Cook reduction. Conclude that unless  $\mathbf{NP} = \mathbf{PH}$ , FACTORING is *not* **NP-Hard** under Cook reduction.

**Solution. Part 1.** Let the language  $L_{\text{Fac}}$  consist of pairs of the form  $(N, x)$  where  $N \in \mathbb{N}, x \in \{0, 1\}^*$ , such that  $(N, x) \in L_{\text{Fac}}$  if and only if there is a string  $y \in \{0, 1\}^*$  such that  $x \circ y$  is the binary representation of a prime factor of  $N$ , where  $x \circ y$  denotes the concatenation of strings  $x$  and  $y$ . That is,  $(N, x) \in L_{\text{Fac}}$  if and only if  $x$  is a prefix of a string representing a prime factor of  $N$ . We show that:

**Claim 1.** FACTORING Cook-reduces to  $L_{\text{Fac}}$ .

**Claim 2.**  $L_{\text{Fac}} \in \mathbf{NP} \cap \mathbf{coNP}$ .

Hence if  $\mathbf{P} = \mathbf{NP} \cap \mathbf{coNP}$ , then FACTORING can be solved by a polynomial-time algorithm.

**Proof of Claim 1:** We describe an algorithm that solves the FACTORING problem in polynomial time, given access to a membership oracle  $\mathcal{O}_{L_{\text{Fac}}}$  for  $L_{\text{Fac}}$ . The following algorithm  $A^{\mathcal{O}_{L_{\text{Fac}}}}(N)$  finds one prime factor of  $N$  in polynomial time, given access to  $\mathcal{O}_{L_{\text{Fac}}}$ . One can repeatedly use  $A^{\mathcal{O}_{L_{\text{Fac}}}}$  to completely factor  $N$ ; that is, compute  $p_1 = A^{\mathcal{O}_{L_{\text{Fac}}}}(N)$ , then  $p_2 = A^{\mathcal{O}_{L_{\text{Fac}}}}(N/p_1)$ ,  $\dots$ , until  $N$  is completely factored. Since  $N$  has at most  $\log_2 N$  prime factors, the resulting algorithm is still efficient.

**Algorithm**  $A^{\mathcal{O}_{L_{\text{Fac}}}}$ :

On input  $N \in \mathbb{N}$  (in the binary representation):

1. Test whether  $N$  is a prime. If so, output  $N$  and halt.
2. Let  $p \leftarrow \varepsilon$ , where  $\varepsilon$  is the empty string.
3. Repeat the following (until  $p$  represents a prime factor of  $N$ ):
  - (a) Let  $b \leftarrow 0$ .
  - (b) If  $\mathcal{O}_{L_{\text{Fac}}}(N, p \circ b) = 0$ , i.e. if  $(N, p \circ b) \notin L_{\text{Fac}}$ , let  $b \leftarrow 1$ .
  - (c) Let  $p \leftarrow p \circ b$ .
  - (d) Test whether  $p$  is the binary representation of a prime number. If so, output  $p$  and halt.

We first observe that  $A^{\mathcal{O}_{L_{\text{Fac}}}}(N)$  finds one prime factor of  $N$ : If  $N$  is prime then Phase (a) of the algorithm outputs  $N$ . If  $N$  is composite, then the main loop in Phase (c) finds a prime factor  $p$  of  $N$ , one bit at a time. We also observe that the running time of  $A^{\mathcal{O}_{L_{\text{Fac}}}}(N)$  is polynomial in  $n$ , which is the number of bits representing  $N$ . This is so because primality can be tested in polynomial time, and the number of iteration in the main loop in Phase (c) is at most  $n$ .

**Proof of Claim 2:** Note that the complete factorization  $(p_1^{e_1}, \dots, p_k^{e_k})$  of  $N$  serves both as a certificate of membership for  $(N, x)$  if  $(N, x) \in L_{\text{Fac}}$ , and as a certificate of non-membership for  $(N, x)$  if  $(N, x) \notin L_{\text{Fac}}$ .

To verify the membership or non-membership of a pair  $(N, x)$ , using  $(p_1^{e_1}, \dots, p_k^{e_k})$ , simply:

1. Verify that  $p_1, \dots, p_k$  are all primes.
2. Verify that  $N = \prod_{i=1}^k p_i^{e_i}$ .
3. Verify that  $x$  appears as a prefix of  $p_i$  for at least one  $i$ .

The correctness of the verification procedure follows from the unique factorization of integers, and its efficiency follows from that of the primality test. Therefore,  $L_{\text{Fac}} \in \mathbf{NP} \cap \mathbf{coNP}$ .

**Part 2.** If FACTORING were **NP-Hard** under Cook reduction, then the language  $L_{\text{Fac}}$  defined above would also be **NP-Hard** under Cook reduction, as FACTORING Cook-reduces to  $L_{\text{Fac}}$  and reductions are transitive.

**Claim 3.** If a language in  $\mathbf{NP} \cap \mathbf{coNP}$  is **NP-Hard** under Cook reduction, then  $\mathbf{NP} = \mathbf{coNP}$ .

Since  $L_{\text{Fac}} \in \mathbf{NP} \cap \mathbf{coNP}$ , it follows that if FACTORING were **NP-Hard**, then  $\mathbf{NP} = \mathbf{coNP}$ . The latter implies that  $\mathbf{NP} = \mathbf{PH}$ .

**Proof of Claim 3:** Let  $L \in \mathbf{NP} \cap \mathbf{coNP}$  and suppose that  $L$  is **NP-Hard** under Cook reduction. We show that this implies that  $\mathbf{NP} \subseteq \mathbf{NP} \cap \mathbf{coNP}$ , and thus  $\mathbf{NP} \subseteq \mathbf{coNP}$ . It is well known and can be easily shown that the latter holds if and only if  $\mathbf{NP} = \mathbf{coNP}$ .

Let  $\tilde{L} \in \mathbf{NP}$ . By assumption  $\tilde{L}$  Cook-reduces to  $L$ . That is, there is an algorithm  $A_L^{\mathcal{O}}$  that decides  $\tilde{L}$  in polynomial time given access to an oracle  $\mathcal{O}_L$  for the membership of  $L$ . We show that  $\tilde{L} \in \mathbf{NP} \cap \mathbf{coNP}$ . To do so, we show how to certify the membership and non-membership with respect to  $\tilde{L}$ .

Consider any string  $x$  and the execution of  $A_L^{\mathcal{O}}(x)$ . Let  $q_1, \dots, q_m$  be the queries made by  $A$  to  $\mathcal{O}_L$  and let  $a_1, \dots, a_m$  be the corresponding answers, i.e.  $a_i = \mathcal{O}_L(q_i)$  for each  $i$ . Since  $L \in \mathbf{NP} \cap \mathbf{coNP}$ , each member of  $L$  has a certificate of membership and each non-member of  $L$  has a certificate of non-membership with respect to  $L$ . Let  $\pi_1, \dots, \pi_m$  be the corresponding certificates of  $q_1, \dots, q_m$  respectively with respect to  $L$ . It follows that with respect to  $\tilde{L}$ ,  $\{(q_1, a_1, \pi_1), \dots, (q_m, a_m, \pi_m)\}$  serves both as a certificate of membership if  $x \in \tilde{L}$ , and as a certificate of non-membership if  $x \notin \tilde{L}$ .

To verify the membership or non-membership of  $x$  with respect to  $\tilde{L}$  using  $\{(q_1, a_1, \pi_1), \dots, (q_m, a_m, \pi_m)\}$ , simply:

1. Run  $A$  on  $x$ .
2. When  $A$  makes the  $i$ -th query, verify that it is  $q_i$ , and verify that  $a_i$  is the correct answer using  $\pi_i$ . If so, give  $a_i$  to  $A$ ; else reject.
3. Run  $A$  until it halts, and finally verify based on  $A$ 's output.

The verification procedure is clearly correct, and is efficient as  $A$  is.