

## Solution Set for CPSC 468/568 Exam 2

### Question 1

- (a) Unknown. We know that  $\text{PH} \subseteq \text{P}^{\#\text{P}} \subseteq \text{PSPACE}$  (the first inclusion is Toda's Theorem) but not whether these inclusions are proper.
- (b) Unknown. We know that  $\text{BPP} \subseteq \sum_2^p \cap \prod_2^p$  (see Theorem 7.18 in Arora-Barak) but not whether it is contained in  $\text{NP} \cup \text{coNP}$ .
- (c) True, because  $\text{IP} = \text{PSPACE}$  and  $\text{coNP} \subseteq \text{PSPACE}$ .
- (d) True. Take the one-round, private-coin proof system for Graph Non-Isomorphism in Section 8.3 of Arora-Barak and apply the Goldwasser-Sipser Theorem.

### Question 2

- (a) For every  $x$  in  $L$ , there is an oracle  $O$  such that  $M^O$  accepts  $x$  with probability 1. For every  $x$  not in  $L$ , for all oracles  $O^*$ ,  $M^{O^*}$  accepts  $x$  with probability at most  $1/2$ . (Note that there is nothing special about  $1/2$  in this definition. One would get the same class of languages if it were anything between  $1/\text{poly}(|x|)$  and  $1 - 1/\text{poly}(|x|)$ .)
- (b) A prover is adaptive, but an oracle is non-adaptive. That is, in its interaction with the verifier on a given input  $x$ , a prover can put off deciding how it is going to answer a particular question until the verifier asks it; so the prover's answer in round  $i$  can depend on the first  $i-1$  rounds of questions and answers. By contrast, an oracle's answers to all questions that the verifier might ask on input  $x$  are fixed before the interaction begins.
- (c) See Example 18.3, part 1 in Arora-Barak (page 18.4). The "proof"  $\pi$  in that example is what we are calling an "oracle."

### Question 3

- (a) A probabilistic polynomial-time (ppt) Turing Machine is a nondeterministic, polynomial-time Turing Machine that, in each step at which it must choose between transition functions  $\delta_0$  and  $\delta_1$ , chooses each with probability exactly  $1/2$ . Language  $L$  is in  $\text{RP}$  if there is a ppt Turing Machine  $M$  such that, for all  $x$  in  $L$ ,  $M$  accepts  $x$  with probability at least  $3/4$ , and, for all  $x$  not in  $L$ ,  $M$  accepts  $x$  with probability 0. (Here, " $M$  accepts  $x$  with probability  $p$ " means that, if  $X$  is the number of computation paths that  $M$  may take on input  $x$ , then  $pX$  of them are accepting paths.) Language  $L$  is in  $\text{coRP}$  if there is a ppt Turing Machine  $M$  such that, for all  $x$  in  $L$ ,  $M$  accepts  $x$  with probability 0, and, for all  $x$  not in  $L$ ,  $M$  accepts  $x$  with probability at least  $3/4$  (i.e., if the complement of  $L$  is in  $\text{RP}$ ). Language  $L$  is in  $\text{BPP}$  if there is a ppt Turing Machine  $M$  such that, for all  $x$  in  $L$ ,  $M$  accepts  $x$  with probability at least  $2/3$ , and, for all  $x$  not in  $L$ ,  $M$  accepts  $x$  with probability at most  $1/3$ .
- (b) Let  $L$  be a language in  $\text{RP} \cap \text{coRP}$ , where  $M_1$  is a ppt Turing Machine that accepts  $L$  and  $M_2$  is a ppt Turing Machine that accepts the complement of  $L$ . Let  $q(n)$  be the larger of

the running times of  $M_1$  and  $M_2$ . The following ppt Turing Machine  $M$  clearly accepts  $L$ : On input  $x$  of length  $n$ , run each of  $M_1$  and  $M_2$   $s(n)$  times on input  $x$ , using independent coin tosses for each run. If on any of these runs  $M_1$  accepts  $x$ , then  $M$  accepts  $x$  and halts. Similarly, if on any of these runs  $M_2$  accepts  $x$ , then  $M$  rejects  $x$  and halts. If  $M$  finishes these  $s(n)$  independent runs of  $M_1$  and  $M_2$  without halting, then it simulates  $M_1$  on all possible coin-toss sequences (accepting  $x$  and halting if  $M_1$  ever accepts) and then simulates  $M_2$  on all possible coin-toss sequences (rejecting  $x$  and halting if  $M_2$  ever accepts). Note that  $M$  always outputs the correct answer, as required by the definition of ZPP. Suppose that the probability that  $M$  finishes the  $s(n)$  independent runs of  $M_1$  and  $M_2$  without halting is at most  $2^{-q(n)}$ . Then the expected running time of  $M$  on input  $x$  of length  $n$  is at most  $2s(n)q(n)(1 - 2^{-q(n)}) + 2(2^{q(n)})q(n)(2^{-q(n)}) = 2q(n)[s(n)(1 - 2^{-q(n)}) + 1]$ , and this is polynomially bounded, provided that  $s(n)$  is polynomially bounded. By applying the Chernoff Bound on the tails of the binomial distribution to amplify the correctness probability of  $M_1$  and  $M_2$ , we can indeed achieve error bound  $2^{-q(n)}$  with polynomially many runs  $s(n)$ . This shows that  $RP \cap \text{coRP}$  is contained in ZPP.

Now let  $L$  be a language in ZPP and  $M$  be a probabilistic TM that accepts  $L$  and has expected (polynomial) running time  $q(n)$ . Because ZPP is closed under complementation, it suffices to show that  $L$  is in RP. Let  $M'$  be a probabilistic TM that, on input  $x$  of length  $n$ , simulates  $M$  for time  $4q(n)$ . If, during this time,  $M$  accepts  $x$ , then  $M'$  accepts  $x$  and halts; if, during this time,  $M$  rejects  $x$ , or if  $M$  has neither accepted nor rejected during the first  $4q(n)$  steps of the simulation, then  $M'$  rejects  $x$  and halts. The running time of  $M'$  is bounded by (the polynomial)  $4q(n)$ . If  $x$  is not in  $L$ , then  $M'$  rejects  $x$ . If  $x$  is in  $L$ , then the probability that  $M'$  rejects  $x$  is exactly the probability that running  $M$  to completion on input  $x$  would take time more than  $4q(n)$ ; that is the probability that a random variable attains a value that is more than 4 times its expected value, which, by Markov's inequality, is at most  $1/4$ .

#### Question 4

Recall that the permanent of an  $n$ -by- $n$ , integer-valued matrix  $A$  is defined by the formula

$$\text{Perm}(A) = \sum_{\sigma \in S_n} \prod_{i=1}^n A_{i, \sigma(i)} \quad (*)$$

- (a) A general  $n$ -by- $n$ , integer-valued matrix  $A$  corresponds to a weighted digraph  $G_A$  as follows. The vertices of  $G_A$  are  $\{v_1, \dots, v_n\}$ . If  $A_{ij} = 0$ , then there is no directed edge  $(v_i, v_j)$  in  $G_A$ . If  $A_{ij} = k \neq 0$ , then there is a directed edge  $(v_i, v_j)$  in  $G_A$  of weight  $k$ . Self-loops are allowed, because the diagonal entries of  $A$  may be nonzero. A cycle cover of a digraph is a set  $\{C_1, \dots, C_m\}$  of cycles that "covers"  $G_A$  in the following sense: In the sub-digraph consisting of vertex set  $\{v_1, \dots, v_n\}$  and exactly those edges in  $C_1, \dots, C_m$ , each vertex has in-degree 1 and out-degree 1. For general integer matrices,  $\text{Perm}(A)$  is defined to be the sum, over all cycle covers  $\{C_1, \dots, C_m\}$  of the product of the weights on all the directed edges in  $\{C_1, \dots, C_m\}$ . To see why this is equivalent to  $(*)$ , note that there is a one-to-one correspondence between permutations  $\sigma \in S_n$  and *potential* cycle covers. Basic group theory tells us that each  $\sigma \in S_n$  has a unique decomposition into disjoint cycles  $(i_1 \dots i_{j_1})(i_{j_1+1} \dots i_{j_2}) \dots (i_{j_{l+1}} \dots i_n)$ ; this notation is interpreted to mean that  $\sigma$  maps

$i_1$  to  $i_2$ ,  $i_2$  to  $i_3$ , ...,  $i_{j_1}$  to  $i_1$ ,  $i_{j_1+1}$  to  $i_{j_1+2}$ , ...,  $i_{j_2}$  to  $i_{j_1+1}$ , *etc.*; cycles of size one are allowed, because  $\sigma$  may have fixed points. This cycle decomposition of  $\sigma$  corresponds to a cycle cover of  $G_A$  if and only if each pair of consecutive indices  $i, \sigma(i)$  in one of the cycles of the decomposition corresponds to a directed edge  $(v_i, v_{\sigma(i)})$  in  $G_A$ ; furthermore, these are the only  $\sigma$ 's for which the product of the  $A_{i\sigma(i)}$  is nonzero.

- (b) An  $n$ -by- $n$ , 0-1 matrix  $A$  corresponds to a bipartite graph  $G_A$  (and vice versa) as follows. The vertices of  $G_A$  are  $\{v_1, \dots, v_n, w_1, \dots, w_n\}$ , and all edges of  $G_A$  are of the form  $\{v_i, w_j\}$ ; if  $A_{ij} = 1$ , then the edge  $\{v_i, w_j\}$  is present in  $G_A$ , and, if  $A_{ij} = 0$ , then this edge is absent. For 0-1 matrices  $A$ ,  $\text{Perm}(A)$  is defined to be the number of perfect matchings in  $G_A$ , *i.e.*, the number of sets  $\{e_1, \dots, e_n\}$  of edges of  $G_A$  such that each  $v_i$  and each  $w_j$  is an endpoint of exactly one  $e_k$ . To see why this is equivalent to (\*), note that there is a one-to-one correspondence between permutations  $\sigma \in S_n$  and sets  $\{\{v_1, w_{\sigma(1)}\}, \dots, \{v_n, w_{\sigma(n)}\}\}$  of pairs of nodes in  $G_A$ , where all of the  $w_{\sigma(i)}$  are distinct, *i.e.*, between permutations and *potential* perfect matchings. A set  $\{\{v_1, w_{\sigma(1)}\}, \dots, \{v_n, w_{\sigma(n)}\}\}$  is a perfect matching if and only if the product of the  $A_{i\sigma(i)}$  is 1.
- (c) See the paragraph directly preceding Section 9.2.2 of Arora-Barak. The only part of that paragraph that is not clear is the explanation of how to replace edges  $(u,v)$  of weight  $k$ . If  $k > 0$ , then replace nodes  $\{u, v\}$  with nodes  $\{u, w_1, \dots, w_k, v\}$ , and replace the edge  $(u,v)$  with edges  $(u, w_1), \dots, (u, w_k), (w_1, w_1), \dots, (w_k, w_k), (w_1, v), \dots, (w_k, v)$ , all of weight 1. If  $k < 0$ , then do the same, except assign weight -1 to the self-loops  $(w_i, w_i)$ .

### Question 5

- (a) See Definition 8.10 in Arora-Barak. The collection of  $k$ -by- $n$  0-1 matrices is such a collection; if  $x$  is an  $n$ -bit vector (equivalently, an  $n$ -by-1 matrix), and  $H$  is a uniformly randomly chosen matrix in the collection, then  $H(x)$  is just the product of  $H$  by  $x$  over  $\text{GF}(2)$ .
- (b) Theorem 9.15 in Arora-Barak states one of the two versions of this result that we discussed in class. The statement in the original Valiant-Vazirani paper (also distributed in class) is as follows. Let  $x$  be a CNF formula. For any predicate  $Q$ , the function  $\text{USAT}_Q(x)$  is defined to be 0 if  $\#\text{SAT}(x) = 0$ , 1 if  $\#\text{SAT}(x) = 1$ , and  $Q(x)$  if  $\#\text{SAT}(x) > 1$ . For all  $Q$ , there is a ppt reduction from SAT to  $\text{USAT}_Q$ . (Thus, restricting attention to formulas that have either 0 or 1 satisfying assignment does not make the SAT problem appreciably easier.)
- (c) The formula  $g$  is defined to be  $f \wedge (x_{i_1} \oplus x_{i_2} \oplus \dots \oplus x_{i_j} \oplus 1)$ , where  $i_1, \dots, i_j$  are the indices of the bit positions in which  $w$  has a 1. The formula  $h$  is the CNF equivalent of  $f \wedge (y_1 \leftrightarrow (x_{i_1} \oplus x_{i_2})) \wedge (y_2 \leftrightarrow (y_1 \oplus x_{i_3})) \wedge \dots \wedge (y_{j-1} \leftrightarrow (y_{j-2} \oplus x_{i_j})) \wedge (y_{j-1} \oplus 1)$ .

### Question 6

- (a) The prime  $p$  must be greater than  $2^n$ ; it must also be bounded above by  $2^{r(n)}$ , for some fixed polynomial  $r(n)$ . Bertrand's Postulate (actually proven by Chebyshev) tells us that it suffices to fix any  $r(n) \geq n+1$ . The arithmetization over  $\text{GF}[p]$  of the 3CNF formula

$\varphi(x_1, \dots, x_n)$  is an  $n$ -variable multinomial  $q(X_1, \dots, X_n)$ . To construct  $q$  from  $\varphi$ , map the boolean variable  $x_i$  to the indeterminate  $X_i$ , the negation of  $x_i$  to  $(1-X_i)$ , the disjunction of boolean literals to the sum of the corresponding arithmetic expressions, and the conjunction of boolean clauses to the product of the corresponding arithmetic terms. Then the formula  $\varphi$  is unsatisfiable if and only if the sum of the values of  $q$  on the  $2^n$  vectors in  $\{0,1\}^n$  is 0; note that “0” and “1” denote arithmetic values in  $\text{GF}[p]$  when they are used as arguments to or values of  $q$ .

- (b) For each function  $f$  mapping  $\{0,1\}^*$  to  $\{0,1\}^*$ , let  $f_k$  be the (finite) function that agrees with  $f$  on inputs of length  $k$  or less (and is undefined for inputs of length greater than  $k$ ). Then  $f$  is *downward-self-reducible* if there is a polynomial-time oracle machine  $R$  such that, for each  $k$  and each  $x$  of length  $k$ ,  $R$  computes  $f(x)$  on input  $x$  when given access to an oracle for  $f_{k-1}$ . To see that the permanent function is downward-self-reducible, use the *co-factor expansion* of the permanent, which is the same as the (more familiar) co-factor expansion of the determinant, except that signs are all positive: If  $A$  is an  $n$ -by- $n$  integer matrix, let  $A_{ij}$  be the entry in the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column and  $M_{ij}$  be the  $(i, j)^{\text{th}}$  *minor* of  $A$  (i.e., the  $(n-1)$ -by- $(n-1)$  matrix obtained by eliminating the  $i^{\text{th}}$  row and  $j^{\text{th}}$  column of  $A$ ). Then, for any fixed row  $i$ ,  $\text{Perm}(A)$  can be expressed as the sum, over  $1 \leq j \leq n$ , of  $A_{ij}\text{Perm}(M_{ij})$ , and, for any fixed column  $j$ ,  $\text{Perm}(A)$  can be expressed as the sum, over  $1 \leq i \leq n$ , of  $A_{ij}\text{Perm}(M_{ij})$ .