Quiz 2 - COMS E6261: Advanced Cryptography

Question 1

Recall the PPA-complete problem LONELY from class: given a circuit $C : \{0,1\}^n \to \{0,1\}^n$ s.t. x,y are paired (aka "matched") iff $C(x) = y \ \wedge \ C(y) = x$, either demonstrate that 0^n is paired, or find another element $x \neq 0^n$ that is unpaired.

What if we represented pairings a different way? Given a circuit $C : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, let x and y be paired iff $C(x,y) = C(y,x) = 1$.

Then consider the same search problem as above: Given such a C, either demonstrate that 0^n is paired, or return another element $x \neq 0^n$ that is unpaired.

Is this problem in TFNP?

○ Yes

◉ No

Explanation: This problem is not in TFNP for several reasons. For one, it's not total – the way we defined pairing does not necessarily correspond to a perfect matching (e.g., some x could be matched to more than one y) and thus it's not necessary that, even if 0^n is unpaired, there's another unpaired element. Moreover, it's not clear how to verify in polynomial time that a given x is unmatched with any other element.

Question 2

Let us slightly change the representation in the previous problem. x is defined to be unpaired if $C(x,x) = 1$. That is, x and y are paired iff $C(x,y) = C(y,x) = 1 \ \wedge \ C(x,x) = C(y,y) = 0$.

Consider the following search problem: Given C output one of:

(1) $\perp$ if 0^n is not unpaired.

(2) A violating element x s.t. $C(x,x) = 1$ but x satisfies the other conditions to be paired with another element.

(3) A violating element x s.t. $C(x,x) = 0$ but x is not paired with any other element.

(4) A violating element x s.t. $C(x,x) = 0$ but x is paired to more than one element.

(5) Another unpaired element $w \neq 0^n$.

Question 2.1

This problem is total.

- ◉ True
- ○ False

Explanation: (2), (3) and (4) ensure that C encodes a valid matching; in particular, an element can only be paired with at most one other element, solving the issue in the previous problem. Thus, if there is no solution of type (1)–(4) that ensures that 0^n is unpaired and that there is another unpaired element (since C is a matching on an odd-sized domain), so there must be a solution of type (5).

Question 2.2

Which of the allowed solution types must have efficiently verifiable NP certificates? (1)-(5) are the same as above:

- ☑ (1) $\perp$ if 0^n is not unpaired.
- ☑ (2) A violating element x s.t. $C(x, x) = 1$ but x satisfies the other conditions to be paired with another element.
- ☐ (3) A violating element x s.t. $C(x, x) = 0$ but x is not paired with any other element.
- ☑ (4) A violating element x s.t. $C(x, x) = 0$ but x is paired to more than one element.
- ☑ (5) Another unpaired element $w \neq 0^n$.

Explanation: (1) $\perp$ is correct if $C(0^n, 0^n) = 0$, which can be checked directly. For (2) the certificate is another y s.t. $C(x, y) = C(y, x) = 1$ and $C(y, y) = 0$; the condition $C(x, x) = 1$ can be checked directly. For (3) we don't know how to certify that there is no y such that $C(x, y) = C(y, x) = 1 \wedge C(y, y) = 0$. For (4) the certificate are two y, z such that x is paired with both y and z. For (5) we can check that $C(w, w) = 1$.

Remark: The point of the last two questions was to demonstrate that it's not always easy to come up with other ways to define matching that is still in TFNP. We also note (not part of what we studied so far) that the way defined in this problem, while not (known to be) in TFNP, is higher in the hierarchy, in $\mathrm{TF}\Sigma_2$ (where we can also define analogues of classes like PPA, PPAD, and so on).

Question 3

Let $C : \{0,1\}^{n+1} \to \{0,1\}^n$ be a an instance of WEAK-PIGEON (where as usual, C is of polynomial size in n). This instance has exponentially many solutions.

- ◉ True
- ○ False

Explanation: We mentioned in class that a circuit $C : \{0,1\}^{n+1} \to \{0,1\}^n$ has at least 2^n colliding pairs – this follows by a simple counting argument, as the domain includes 2^n more elements than the co-domain.

Question 4

Let R be a search problem in PWPP. Then any instance of R must necessarily have exponentially many solutions.

- ○ True
- ◉ False

Explanation: This is not necessarily true; R in PWPP means that instances of R can be reduced to instances of WEAK-PIGEON, such that given a solution to the WEAK-PIGEON instance, we can compute a solution to the R instance. It is true that **all** the solutions to the WEAK-PIGEON instance must yield a solution to the original problem, but many can map to the same problem. For instance we mentioned in class that Factoring reduces to PWPP (with randomness), but the original problem (Factoring) has polynomially many solutions (unique factors).

Question 5

Let N be an integer of length n and suppose it were proven that there is always a quadratic non-residue mod N in the range $[1, \sqrt{N}]$. This would mean that that Good Integer Factoring GIF can be reduced to PPP, using the result that we saw in class (recall we saw that if you can find a quadratic non-residue for a good integer, you can reduce to PPP).

- ○ True
- ◉ False

Explanation: This is false because $\sqrt{N}$ is super-polynomial in the input representation size n, so looping through the range to find a non-residue and apply to the reduction to PPP would not be efficient. We do not know how to otherwise find the non-residue in that range. In fact, the problem of factoring itself must have a solution in that range, but does not mean factoring is in FP.

Question 6

Let N be an integer of length n and suppose it were proven that there is always a quadratic non-residue mod N in the range $[N - n^5, N]$. This would mean that that Good Integer Factoring GIF can be reduced to PPP, using the result that we saw in class (recall we saw that if you can find a quadratic non-residue for a good integer, you can reduce to PPP).

- ◉ True
- ○ False

Explanation: This is true, because the range of n^5 numbers can be efficiently looped through.

Remark: Note that this looping would give a reduction that calls the PPP oracle multiple times, while to be in PPP, by the way we define subclasses of TFNP, a problem has to be reducible by a reduction that calls the oracle just once (aka as many-to-one reduction or Turing reduction). This is why we used the wording "reduces to PPP" rather than "in PPP." However, in this case the distinction turns out to not matter, as one can prove that $\text{FP}^{\text{PPP}} = \text{PPP}$ (you're welcome to try to prove this as a non-required homework problem).