1 Problem 1

1.1 Outline

To prove that $\mathsf{ZSP} = \mathsf{NP} \cap \mathsf{coNP}$, I will prove inclusion both ways.

1.2 $\mathsf{ZSP} \subseteq \mathsf{NP} \wedge \mathsf{ZSP} \subseteq \mathsf{coNP}$

First, let's prove $\mathsf{ZSP} \subseteq \mathsf{NP}$. Let's consider some language $L_{ZSP} \in \mathsf{ZSP}$. Then there exists M and K (M being deterministic Zosia Samosia machine and K being oracle) such, that $L(M^K) = L(M^*) = L_{ZSP}$. This means that some Zosia Samosia machine would recognize this language with any oracle ($L(M^*) = L_{ZSP}$), but for some given oracle K it would work in polynomial time in less than T steps. Now, to show that this language is in NP, we create a NP machine. Let's construct $M^{'}$ such, that $L(M^{'}) = L_{ZSP}$ and it simulates M with some oracle. This nondeterministic machine follows the same steps as deterministic M with oracle, with a slight difference: every time M would ask oracle something, $M^{'}$ selects the answer nondeterministically (caching previous answers to not contradict them) instead of asking the oracle. This nondeterministic machine accepts input when it enters accepting state as M with oracle would, otherwise after $T(|input|)$ steps (if M accepts input, then M^K accepts input in less than $T(|input|)$ steps) $M^{'}$ rejects (since M^K would accept in such number of steps, and clearly $M^{'}$ on some nondeterministic run have all the steps made by M with oracle K). If $M^{'}$ accepted, then M with oracle giving answers same as $M^{'}$ chose on its accepting run would also accept.

Now let's focus on $\mathsf{ZSP} \subseteq \mathsf{coNP}$. We've already proven that $\mathsf{ZSP} \subseteq \mathsf{NP}$. This implies that $\mathsf{coZSP} \subseteq \mathsf{coNP}$. And since Zosia Samosia is a deterministic machine, it always accepts or rejects, then it recognizes not only ZSP languages class but also its complement - coZSP. Then $\mathsf{ZSP} = \mathsf{coZSP}$. And thus $\mathsf{ZSP} \subseteq \mathsf{coNP}$.

1.3 $\mathsf{NP} \cap \mathsf{coNP} \subseteq \mathsf{ZSP}$

Now, let's consider some language $L_\#$, for which we know: $L_\# \in \mathsf{NP} \wedge L_\# \in \mathsf{coNP}$. Now let's focus on the fact that $L_\# \in \mathsf{NP}$ for a second. There exists a nondeterministic machine $M_{NP} \in \mathsf{NP}$ and $L(M_{NP}) = L_\#$. Let's switch to $L_\# \in \mathsf{coNP}$: there exists another nondeterministic machine $M_{coNP} \in \mathsf{NP}$ and $L(M_{coNP}) = \overline{L_\#}$. For such $L_\#$ let's construct an oracle Zosia Samosia machine M, that for any oracle recognizes $L_\#$ and for some oracle K recognizes $L_\#$ in polynomial time: $L(M^*) = L(M^K) = L_\#$. Oracle K would help us create run for checking word being in $L_\#$ or $\overline{L_\#}$. Machine M asks its oracle for subsequent steps in simulating M_{NP} and then after finishing run, for subsequent steps in simulating M_{coNP}. We can assume without loss of generality, that each step M makes, it has 2 choices. It simply writes "input # number of current step" on oracle tape, then oracle answers with yes/no, where yes means first choice and no means second one. After running both simulations machine M can answer if input word is in $L_\#$ or $\overline{L_\#}$. Checking its run (both for $L_\#$ and $\overline{L_\#}$) by Zosia Samosia machine M is done in polynomial time. If oracle that was used is K, then one of the runs must be correct and machine recognized word to be in $L_\#$ or $\overline{L_\#}$ (because $L_\# \in \mathsf{NP} \cap \mathsf{coNP}$). If both runs with oracle are incorrect (so it means the oracle wasn't K), then machine M checks if input belongs to $L_\#$ in $\mathsf{EXPTIME}$ since $\mathsf{NP} \subseteq \mathsf{EXPTIME}$.

2 Problem 2

2.1 Outline

First, I would prove that communication protocol described in hint is correct. Then I would show that using this protocol as a blackbox solution, Alice could answer always correctly when there is no triangle in $A \cup B$ and with probability $\geqslant \frac{1}{2}$ when such triangle exists.

2.2 Proof of hint

First let's notice that for every n, based on Bertrand's postulate (for every n exists at least one prime number p such that $n < p < 2n$), exists a prime number $4n \leqslant p \leqslant 8n$. We know that polynomial $\mathbb{Z}_p$ of degree k has at most k roots (zero points). When we interpret points in V as a square of points (if $\sqrt{n} \notin \mathbb{Z}$ then we can use padding to make this V representation a proper square), then we could say that polynomial Q_i describes i-th row of V in terms of which elements belong to set A. Same for R_i and belonging to set B. Now let's notice that when we multiply polynomials $Q_i * R_i$ we receive a polynomial that describes intersection of both sets A and B in row i. For example if $Q_i(j) * R_i(j) = 1$ then both $Q_i(j)$ and $R_i(j)$ are equal to 1, and thus element j of row i belongs to $A \cap B$. Otherwise, if element j doesn't belong to A or to B, the multiplication gives 0. We would use that property proving hint.

An honest Merlin is expected to compute both Q_i and R_i polynomials (using Lagrange Interpolating polynomial), then $P = \sum_{i=1}^{k} Q_i * R_i$ and to send coefficients of that polynomial (P) to Alice (he might actually be dishonest and send something different than $\sum_{i=1}^{k} Q_i * R_i$ so we need a way to check its correctness). Let's notice that since this polynomial has degree of at most $2(k-1)$ and it's a polynomial in $\mathbb{Z}_p$, then it would require $O(k \log n)$ bits. Since it has $O(k)$ coefficients and each one of them is $< 8n$ then each coefficient requires $O(\log n)$ bits to be written in binary. Since k is $O(\sqrt{n})$, then the bits required to send Merlin polynomial to Alice are $O(\sqrt{n} \log n)$.

Now, Bob draws $l \in \mathbb{Z}_p$ uniformly (and at random) and calculates polynomials R_i (also using Lagrange Interpolating Polynomial). Then he calculates value of each R_i in l. Then Bob sends l and k numbers: $R_i(l)$ (for each i) to Alice. (Each number is $O(\log n)$, so together it makes $O(k \log n) = O(\sqrt{n} \log n)$ bits to send)

Alice receives Bobs message and calculates coefficients for Q_i. Then, she calculates $Q_i(l)$ for each i based on the value of l she received from Bob. Now, Alice computes value of $T = \sum_{i=1}^{k} Q_i * R_i$ (the proper polynomial) in l and wants to check if Merlins polynomial is correct, in other words if $P = T$. So she also computes $P(l)$ and compares the results. If Merlin cheated (i.e. sent $P \neq T$), it stands that probability: $P(P(l) = T(l) \leqslant \frac{2(k-1)}{p} \leqslant \frac{1}{2}$. If $P(l) = T(l)$ (either Merlin sent $P = T$ or he sent $P \neq T$ but those polynomials have the same value in l) Alice assumes those polynomials are the same and checks values of $P(j)$ for each $0 < j \leqslant k$. If all those values are equal to 0 then the subsets are disjoint and Alice accepts. Otherwise, if for at least one j: $P(j) = 1$, Alice rejects. Alice also rejects if $P(l) \neq T(l)$.

2.3 Triangles solution

We are going to heavily use communication protocol described in hint. There is a slight difference though. Since Bob tosses a coin to draw consecutive bits for l, the number he draws from those tosses (treating each toss as a binary digit of a number) will be in range $[0, 2^t - 1]$ (if he tossed the coin t times). All polynomials used in communication protocol, on the other hand are in $\mathbb{Z}_p$, which means it's unavoidable that some numbers have higher chance of being drawn (at most 2 times more than any other). This is due to "overflow", that would happen when computing $l \mod p$. Because of the differrence described here, the probability of $P(l) = T(l)$ (if $P \neq T$) changes from $\frac{2(k-1)}{p}$ to $2 * \frac{2(k-1)}{p}$. After all, it's nothing to worry about, since it still holds: $2 * \frac{2(k-1)}{p} \leqslant \frac{1}{2}$

From now on let's look at the hint as a black box, that we define inputs to and interpret outputs from. Let's think about triangles problem. There are couple of possibilities:

- Set A contains a triangle. - Here it's simple, Alice would always answer correctly without any information from Merlin or Bob.
- Set B contains a triangle. - This is also pretty straightforward, Bob sends only one information to Alice, that she should accept, since he has a triangle in his set B.
- Neither A nor B contains a triangle. - Here is where things get more complex, I would describe solution below.

Let's notice, that a triangle that spans across sets A and B can:

- Have 2 nodes in A and 1 node in B.
- Have 1 node in A and 2 nodes in B.

To simplify description below, let's call a set of nodes that complement a triangle with 2 nodes in set X as "$missing(X)$".

Now let's describe which sets we want to intersect in our solution (since we are using hint as a blackbox). Let's notice, that when we intersect A with $missing(B)$ and the intersection is not-empty, then a triangle with 2 of its nodes in B and 1 node in A exists. Similarly, when we intersect B with $missing(A)$, non-empty intersections means there is a triangle with 2 nodes in A and 1 node in B. This is exactly what we would like to check. Let's check both those intersections at once: we can encode A and $missing(A)$ (padded to n elements each) as A', also we encode $missing(B)$ and B (in this order, also padded to n elements) as B'. Now, we simply want to check if intersection of A' and B' is non-empty. If it is, then a triangle in $A \cup B$ exists, otherwise it doesn't.

We should interpret Alice results differently than they are interpreted in hint. If Alice finds Merlin honest (i.e. $P(l) = T(l)$), then if she finds out there exists an intersection of sets A' and B', she accepts - a triangle exists. If Alice finds Merlin cheating (i.e. $P(l) \neq T(l)$) she also accepts (Merlin is trying to convince Alice and Bob that no triangle exists). The case when Alice rejects is when Alice finds Merlin honest and the intersection is empty - which means either Merlin is cheating and Alice with Bob didn't manage to find out about that or there is in fact no such triangle.