P1

First, if we have a subset of at least k students, we can go through each two students check whether they take two same courses. Thus the Diverse Subset Problem is in **NP**.

We can consider reduct independent set problem to the Diverse Subset Problem. For every graph $G = (V, E)$. We can convert each $v \in V$ to a student. And if $v_1, v_2 \in V \wedge (v_1, v_2) \in E$. We can create a new courses j for (v_1, v_2) and set
$A[v_1, j] = 1, A[v_2, j] = 1 \ and \ \forall(i \neq v_1 \wedge i \neq v_2) A[i, j] = 0$

In this reduction, the number of courses is $|E|$ and the number of students is $|V|$. Thus the method is a polynomial reduction. And if $(v_1, v_2) \in E$, they can not be both chosen in the Diverse Subset Problem because $A[v_1, j] = A[v_2, j] = 1$. And if and only if we have a subset of at least k of students that is diverse, we have an independent set in G whose size is $\geq k$.

Since the independent set problem is a **NP-Complete** problem, the Diverse Subset Problem is a **NP-Complete** problem.

P2

Consider we construct a new graph $G' = (V', E')$. $V' = V \cup \{S\}$, $E' = E \cup (\cup_{v \in V} \{(S, v)\})$. Which means we add a new vertex S to the original G, and connect S to every original node in G.

Then we apply the **4-coloring** to the new graph G'. Suppose we are able to do the **4-coloring**. As the new node S is connected to any other nodes in G', so node S must has a distinct color that not same as any other nodes in the graph. If we delete the node S in G' and then the rest of the node are colored by less or equal to 3 colors. If the graph G' is not able to be colored in 4 colors, the original graph also cannot be colored in 3 colors because S consume only 1 color. Thus, any instance graph of **3-coloring** problem can perform such method to become a **4-coloring** one. Thus this is a polynomial reduction from **3-coloring** problem to **4-coloring** problem.

P3

Consider try to select every triples randomly. When the new one does not overlap any other triples we selected we choose to select the new one.

Proof:

Suppose the subset we selected is M, which we have $M \subseteq T$, and the elements in M triples form 3 sets $A \subseteq X, B \subseteq Y, C \subseteq Z$.

For any triple $t \in T - M$, suppose $t = (x, y, z)$, we have
$\neg(x \in X - A \wedge y \in Y - B \wedge z \in Z - C)$, which means at least 1 of its 3 elements is contained by the sets we selected. (*Note that if the condition is not fulfilled which means we can still choose t to our subset*).

On the other words, for each triple $t = (x, y, z)$ we has selected, we can at most replace it by 3 triples which has the property

$$(x_1 \in A \land y_1 \in Y - B \land z_1 \in Z - C)$$
$$(x_2 \in X - A \land y_2 \in B \land z_2 \in Z - C)$$
$$(x_3 \in X - A \land y_3 \in Y - B \land z_3 \in C)$$

For each t in the best case we can do such replace, which means $|OPT| = 3|ALG|$

$\frac{|ALG|}{|OPT|} = \frac{1}{3}$

Thus we can prove that this algorithm is at lease $1/3$ times the maximum possible size.

P4

(1)

If we have a pack arrangement plan for all the objects, we can check the size of all the bins whether the size is exceeded in polynomial time. Thus the decision version of the Bin Packing Problem is in **NP**.

We choose the **Partition Problem** to reduct to this problem. For a Partition Problem instance, we have integers $v_1, \ldots, v_n$, then we construct a new sequence of numbers.

$W = \frac{\sum_{i=1}^{n} v_i}{2}, s_i = \frac{v_i}{W}$

Then let $K = 2$, which means we are try to put all s_i in 2 bins. And the solution of the final state of objects in two bins represent the partition method.

Then the solution of the decision version of **Bin Packing problem** can represent the solution of the **Partition** Problem.

As the **Partition** is a **NP-complete** problem, and this is a polynomial reduction from the **Partition** to the decision version of the **Bin Packing problem**. We can conclude that the decision version of the **Bin Packing problem** is **NP-complete**.

(2)

We consider each object according to their sequence. We trying to put a s_i to current bins. If it cannot be put in the current bins, we create a new bin and put the object in it. The algorithm polynomial because the number of bins will not exceed the number of objects.

Assume that if we can put rational object into a bin, the minimal bin we need to consume is:

$|OPT| = \lceil \sum_{i=1}^{n} s_i \rceil$. Assume that each time we waste some space of each bin, according to our algorithm, the empty space of each bin will not exceed 0.5. (*Note that except the last bin, the empty space of the last bin will not exceed* 1). We can prove this because we can only have at most 1 bin which empty space exceed 0.5. When we meet a new object, if $s_i > 0.5$, we may put it in a new bin, and the new bin's empty space is $1 - s_i < 0.5$. If $s_i \leq 0.5$, if we current have a bin whose empty space exceed 0.5, we can put the object in that bin, otherwise, we put the object in a new bin and then that bin is the only bin whose empty space exceed 0.5. Suppose the empty space of i_{th} bin is e_i

Thus,
$|ALG| = |OPT| + \sum_{i=1}^{|ALG|} e_i < |OPT| + 0.5 \times (|ALG| - 1) + 1 = |OPT| + 0.5 \times |ALG| + 0.5$
.

Then we have $|ALG| < 2 \times |OPT| + 1 \Rightarrow |ALG| \leq 2 \times |OPT|$ because $|ALG|$ and $|OPT|$ are integers.

Thus the approximation ratio of this algorithm is $\frac{|ALG|}{|OPT|} \leq 2$. Thus this is a **2-approximation** algorithm.