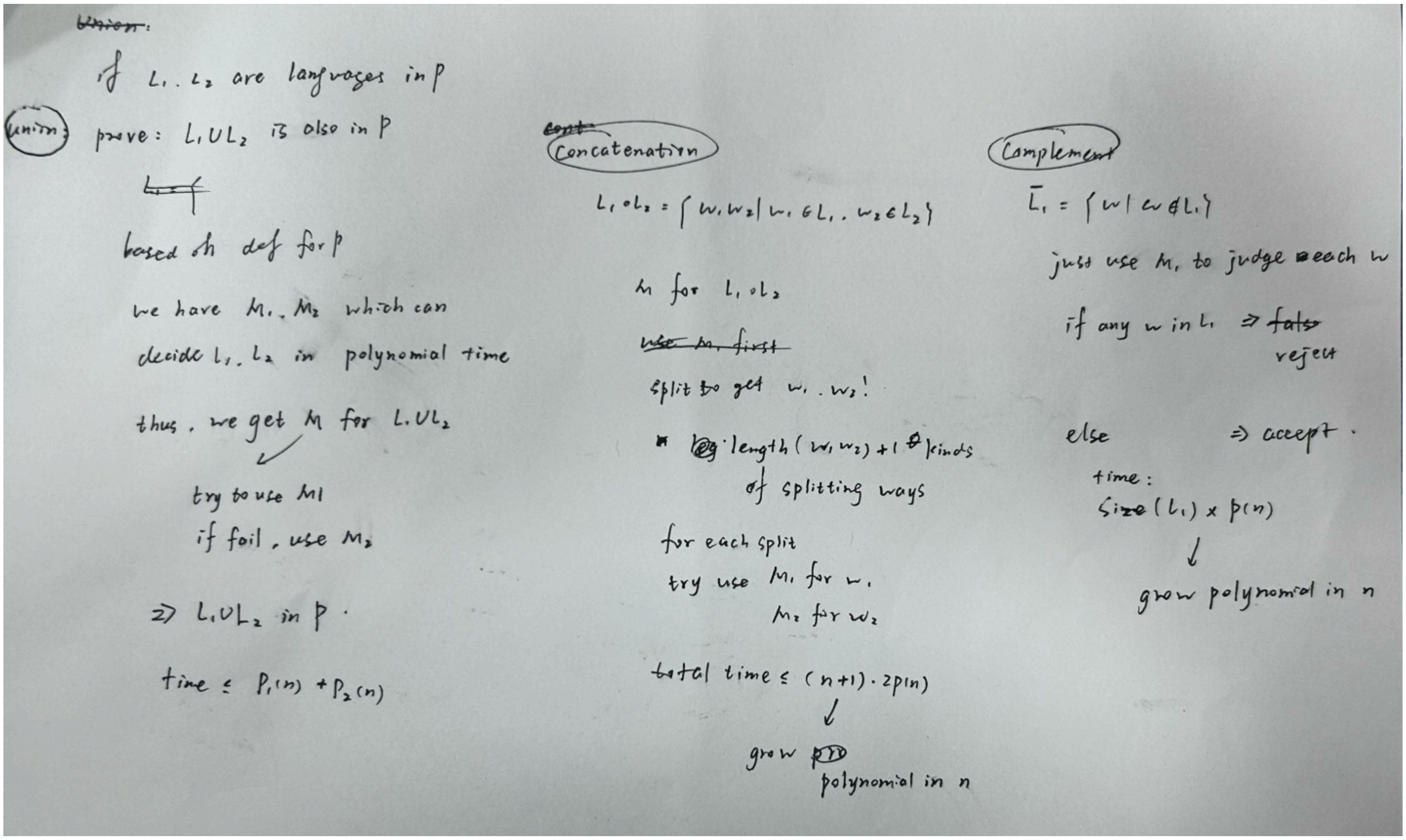
Closure Properties of P

alt text



1. Closed under Union

If $L_1, L_2 \in \mathrm{P}$, then $L_1 \cup L_2 \in \mathrm{P}$.

Key Steps:

- Let M_1 (decides L_1 in poly time $p_1(n)$) and M_2 (decides L_2 in poly time $p_2(n)$) be DTMs.
- Construct M: Run M_1 on input w. If M_1 accepts, M accepts. Else, run M_2; if M_2 accepts, M accepts; else, M rejects.
- Time complexity: $p_1(n) + p_2(n)$ (polynomial, as sum of polynomials is polynomial).

2. Closed under Concatenation

If $L_1, L_2 \in \mathrm{P}$, then $L_1 \circ L_2 = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\} \in \mathrm{P}$.

Key Steps:

- Let M_1 (decides L_1 in poly time $p_1(n)$) and M_2 (decides L_2 in poly time $p_2(n)$) be DTMs.
- Construct M: For input w, check all splits $w = w_1 w_2$. For each split, run M_1 on w_1; if M_1 accepts, run M_2 on w_2; if M_2 accepts, M accepts.
- Time complexity: $(n+1) \cdot O(p(n))$ (polynomial, as $n+1$ (number of splits) times a polynomial is polynomial).

3. Closed under Complement

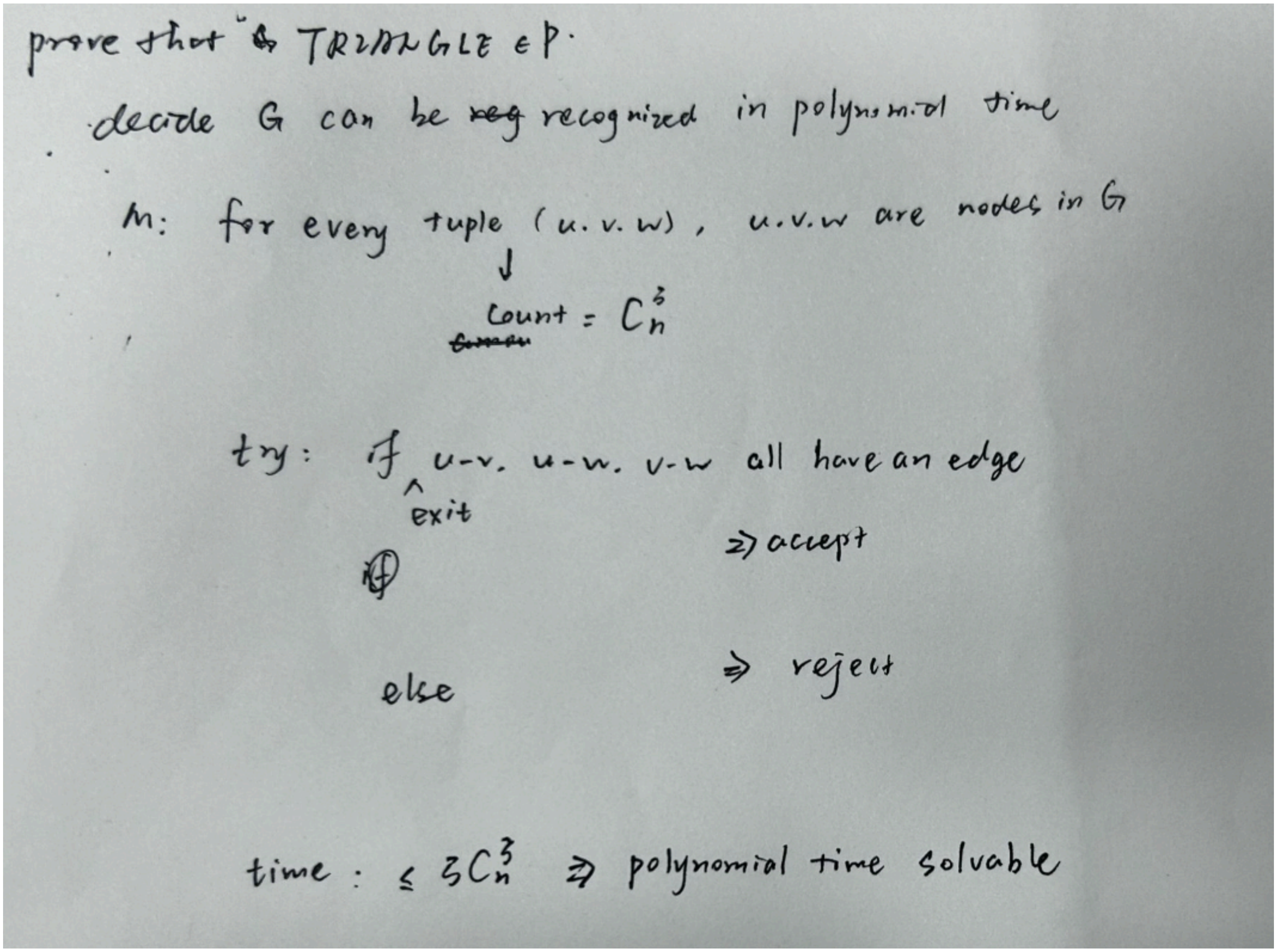
If $L \in \mathrm{P}$, then $\overline{L} = \{w \mid w \notin L\} \in \mathrm{P}$.

Key Steps:

- Let M (decides L in poly time $p(n)$) be a DTM.
- Construct $\overline{M}$: Run M on input w; flip the accept/reject outcome (accept if M rejects, reject if M accepts).
- Time complexity: $O(p(n))$ (polynomial, as only a constant - time flip is added to M's poly - time run).

Prove that $\mathrm{TRIANGLE} \in \mathrm{P}$

alt text



prove that "TRIANGLE ∈ P"

decide G can be ~~reg~~ recognized in polynomial time

M: for every tuple (u.v.w), u.v.w are nodes in G

↓

count = C_n^3

try: if exit u-v. u-w. v-w all have an edge ⇒ accept

else ⇒ reject

time: $\leq 3C_n^3$ ⇒ polynomial time solvable

1. Problem Definition

- **Triangle**: A 3-clique in an undirected graph, meaning a set of 3 vertices where each pair of vertices is connected by an edge (forming a complete subgraph K_3).
- **Language** TRIANGLE: $\text{TRIANGLE} = \{\langle G \rangle \mid G \text{ contains a triangle}\}$, where $\langle G \rangle$ is the encoding of graph G. We need to show there exists a polynomial-time deterministic algorithm to determine if a graph belongs to this language.

2. Polynomial-Time Algorithm Design

Step 1: Enumerate All Triples of Vertices

For a graph G with n vertices, enumerate all possible triples of vertices (u, v, w) (where u, v, w are distinct vertices in the graph).

Step 2: Check if a Triple Forms a Triangle

For each triple (u, v, w), check if all three of the following edges exist:

- There is an edge between u and v;
- There is an edge between u and w;
- There is an edge between v and w.
  If such a triple exists, then G contains a triangle; otherwise, it does not.

3. Time Complexity Analysis

- **Number of Triples Enumerated**: The number of ways to choose 3 vertices from n is $\binom{n}{3} = \frac{n(n-1)(n-2)}{6}$, which has a time complexity of $O(n^3)$ (the highest degree term is n^3).
- **Time to Check Edges**: When the graph is represented by an adjacency matrix, checking if there is an edge between two vertices takes constant time. For each triple, checking 3 edges also takes constant time.

Thus, the overall time complexity of the algorithm is $O(n^3)$, which is **polynomial time**.

Conclusion

Since there exists a polynomial-time deterministic algorithm to determine if a graph contains a triangle, $\text{TRIANGLE} \in \text{P}$.