알고리즘 연구회
전남대학교 게임개발동아리 PIMM

대회 제출 소스코드 AI 평가 결과 보고서

전남대학교 PIMM 알고리즘 파티,
2025 상반기

INFORMATION

작성자 — 전남대학교 게임개발동아리 PIMM
평가 기간 — 2025-03-30 (일) → 2025-04-05 (토)
평가 대상 — <2025 상반기 전남대학교 PIMM 알고리즘 파티> 대회 중 제출 코드 1,035건

들어가는 말

AI의 발전이 매섭습니다. 불과 한두 해 사이에 AI는 우리 생활 깊숙한 곳에 찾아와, 우리의 일상, 학교, 직장, 사회를 완전히 바꿔내고 있습니다.

특히 컴퓨터 분야는 AI의 영향을 강하게 받는 분야 중 하나입니다. 많은 컴퓨터공학과 학부생, 개발자, 연구원들이 AI와 함께 지내고 있습니다. 프로그래밍은 AI의 성능 지표 중 하나가 될 정도로 AI의 주요한 발전 목표가 되었습니다. ChatGPT를 시작으로, 다양한 생성형 AI 제공자가 자신들의 모델 성능을 코드포스 레이팅으로 표현하고 있습니다.

이제 코드포스 대회에 AI 제출을 어렵지 않게 찾아볼 수 있습니다. 이는 비단 코드포스만의 상황인 것이 아니라, 백준 온라인 저지를 포함하여 모든 알고리즘 경쟁 대회 개최지에서 벌어지고 있습니다.

혹자는 AI의 등장으로 컴퓨팅 알고리즘을 공부하는 것이 불필요하다고 주장합니다.

그럼에도 불구하고, 우리 알고리즘 대회 팀은 아직 사람이 작성한 프로그램과 AI 없는 경쟁 프로그래밍 대회가 가치를 잃지 않았다고 생각합니다. 그래서 대회에 AI 답안 제출을 금지했고, 꽤 많은 시간을 투자하여 모든 제출을 분석했습니다.

늦어서 죄송합니다.
여기 분석 결과를 공유합니다.

박종현 belline0124, 전남대학교 게임개발동아리 PIMM.

이 보고서는 https://github.com/pimm-dev/2025-first-half-algorithm-party-editorial 에서도 확인하실 수 있습니다.

평가 결과

다음의 참여자를 생성형 AI를 이용하여 답안을 제출한 것으로 판단합니다:

• 사용자 #159 • 사용자 #161 • 사용자 #175

- 위 참여자들은 다양한 AI 생성 코드 검출 방법에서 지속적으로 AI가 생성한 것으로 의심되는 답안을 제출하였거나, AI가 생성한 코드임이 충분한 설명력을 갖는 답안을 제출하였습니다.

- 위 참여자들을 스코어보드와 상품 지급 대상에서 제외하였습니다.
- 위 참여자들의 전체 제출을 스타트링크와 솔브드에 생성형 AI 사용 의심군으로 제출하였습니다.
- AI를 이용하여 생성한 답안 중 위 참여자들의 답안과 유사한 제출을 스타트링크와 솔브드에 제출하였습니다.

다음의 참여자는 생성형 AI를 이용하여 답안을 제출했을 가능성이 있습니다:

• 사용자 #179 • 사용자 #077 • 사용자 #157

- 위 참여자들의 전체 제출을 스타트링크와 솔브드에 생성형 AI 사용 의심군으로 제출하였습니다.
- AI를 이용하여 생성한 답안 중 위 참여자들의 답안과 유사한 제출을 스타트링크와 솔브드에 제출하였습니다.

- 참여자의 마스킹 처리된 아이디에 할당된 번호는 솔트 처리한 백준 온라인 저지 핸들의 MD5 해시를 사전 순으로 정렬한 값입니다. (구체적인 솔트 처리 식은 비공개입니다.)
- 참여자 본인의 마스킹 처리된 아이디를 확인하고 싶으신 분은 동아리 연락창구로 연락해주시기 바랍니다.

이의제기

이 보고서에 대한 이의제기는 공개일로부터 30일 내에 동아리 연락창구를 통하여주시기 바랍니다.

동아리 연락창구

이 보고서에 대한 문의사항은 아래의 연락처를 통해 연락해주시기 바랍니다.

- 이메일 —— pimm.jnu@gmail.com
- 인스타그램 —— @pimm.jnu
- 깃허브 —— @pimm-dev

이렇게 분석했습니다

방법 1. AI-Code-Detector 사용

방법 2. 다양한 AI로 대회 문제 풀이 답안 생성 후, 유사도 비교

방법 3. 각 제출 데이터를 산출식에 대입하여 의심 수준 평가

방법 4. AI가 생성하는 것으로 널리 알려진 주석 확인

방법 5. <방법 1.>, <방법 2.>, <방법 3.>과 <방법 4.>의 결과로 AI가 생성한 것으로 의심되는 답안 직접 검토

방법 1. AI-Code-Detector 사용

수행 방법

- 이 방법에서는 AI 생성 코드 감지용으로 파인튜닝된 LLM 모델을 사용하여 코드가 AI에 의해 생성된 것인지 판단합니다.
  - 이 평가에는 Poe를 통해 제공되는 AI-Code-Detector 모델을 사용하였습니다: https://poe.com/AI-Code-Detector
- AI로 작성된 코드일 가능성이 70% 이상인 제출을 6회 이상 제출한 참여자에 대해, AI로 작성된 코드일 가능성이 70% 이상인 제출들의 평균값이 75% 이상이라면 AI로 코드를 생성하여 제출하였을 가능성이 높은 참여자로 판단하였습니다.
  - 모델의 정확성을 고려하여 350자 미만의 제출은 평가에서 제외하였습니다.
- 모델에는 소스코드와 함께 다음과 같은 프롬프트를 사용하였습니다:

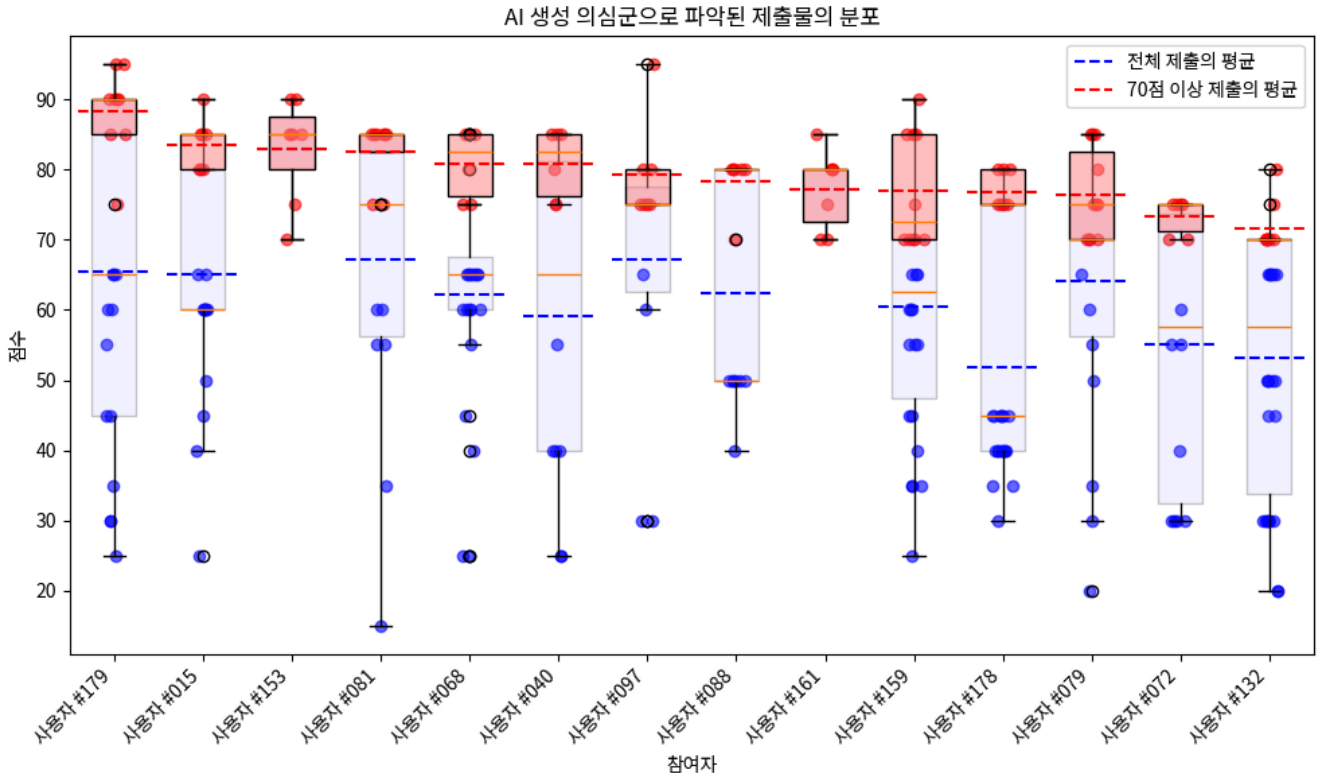
```
Evaluate how much AI-generated probability they have using given source code files.
The result must be in CSV format and contain the filename and the calculated score.
If there are any significant viewpoints when evaluating code, give me a simple summary referring to the actual given source code.
```

- 이 평가는 참고자료로 활용됩니다: 이 평가에 AI-Code-Detector가 알고리즘 문제 해결 전용으로 파인튜닝되지 않았고, 설명력 있게 AI 생성 가능성을 평가하였는지 검증되지 않았습니다.

수행 결과

참여자	평균 점수	계	제출
사용자 #088	78.33%	6	9225***5, 9225***7, 9225***3, 9225***7, 9225***4, 9225***2
사용자 #159	77.00%	10	9225***6, 9225***6, 9225***8, 9225***0, 9225***0, 9225***8, 9225***3, 9225***1, 9226***0, 9226***0
사용자 #179	88.33%	9	9225***9, 9225***9, 9225***0, 9225***2, 9225***0, 9225***8, 9225***4, 9225***4, 9226***7
사용자 #068	80.83%	6	9225***2, 9225***2, 9225***4, 9225***7, 9225***2, 9225***7
사용자 #040	80.83%	6	9225***9, 9225***7, 9225***4, 9225***2, 9226***0, 9226***9

사용자 #081	82.50%	8	9225***4, 9225***5, 9225***4, 9225***5, 9225***3, 9225***1, 9225***2, 9225***6
사용자 #072	73.33%	6	9225***4, 9225***0, 9225***8, 9225***3, 9225***2, 9225***6
사용자 #153	82.86%	7	9225***5, 9225***1, 9225***0, 9225***5, 9225***6, 9225***7, 9225***8
사용자 #097	79.29%	7	9225***4, 9225***9, 9225***8, 9225***7, 9225***3, 9225***9, 9225***2
사용자 #132	71.67%	9	9225***4, 9225***1, 9225***6, 9225***6, 9225***6, 9225***7, 9225***7, 9225***7, 9225***5
사용자 #161	77.14%	7	9225***0, 9225***0, 9225***2, 9225***9, 9225***5, 9225***5, 9225***1
사용자 #178	76.88%	8	9225***9, 9225***6, 9225***5, 9225***6, 9225***3, 9225***6, 9225***2, 9225***2
사용자 #015	83.57%	7	9225***7, 9225***7, 9225***0, 9225***2, 9225***0, 9225***0, 9225***2
사용자 #079	76.36%	11	9225***2, 9225***1, 9225***1, 9225***5, 9225***9, 9225***8, 9225***6, 9225***3, 9226***5, 9226***6, 9226***7



이 표에서 몇몇 참여자가 개인적인 템플릿 코드, 일관된 포매팅 등을 이유로 위양성으로 오진단되었습니다.

AI-Code-Detector가 AI로 생성되었을 가능성이 높은 것으로 진단한 코드 유형입니다:

- 광범위하고 일관된 예외처리
- 복잡한 알고리즘 구현†
- 잘 구성된 문서화 패턴과 주석
- 표준적인 코드 구조†
- 일관된 포매팅†
- 체계적인 변수 명명†
- 템플릿 코드†
- 정규표현식의 사용
- 고차원의 데이터 구조
- 포괄적인 모듈 구성
- 한줄 코딩†
- 라이브러리 사용 패턴

†: 알고리즘 문제 풀이에서 직접 작성한 코드에서도 일반적으로 확인 가능한 유형

AI-Code-Detector가 AI로 생성되었을 가능성이 낮은 것으로 진단한 코드 유형입니다:

- 더 불규칙한 포매팅
- 일관되지 않은 명명 규칙
- 덜 구조화된 코드
- 더 간단한 논리 흐름
- 더 직접적/절차적인 접근 방식†
- 기본적인 예외 처리†
- 더 적은 주석

†: 알고리즘 문제 풀이에서 AI 코드에서도 확인 가능한 유형

- AI에 의해 생성된 한줄 코딩의 대표적인 유형으로 9225***8, 9225***1, 9225***0, 9225***9 번 제출이 지목되었습니다.
- AI에 의해 생성된 정규표현식 사용 코드의 대표적인 유형으로 9225***8, 9225***1, 9225***1 번 제출이 지목되었습니다.
- 그 외에 (A) 9225***2번 제출이 정규표현식으로 작성된 전형적인 한줄 코드, (B) 9225***5번 제출이 복잡한 시스템 구현이 포함된 러스트 코드, (C) 9225***0번 제출이 확장 가능한 람다식과 import 구문으로 AI가 작성한 것으로 지목되었습니다.
  - (B), (C): 모든 제출에서 일관되게 목격된 개인 템플릿 코드가 원인이 되어 위양성 판단을 받은 것으로 확인하였습니다.

방법 2. 다양한 AI로 대회 문제 풀이 답안 생성 후, 유사도 비교

수행 방법

1. 다양한 AI 모델을 사용하여 답안 비교군을 생성합니다.
2. 답안 비교군과 대회 제출을 AI 모델의 임베딩 벡터로 인코딩합니다.
3. 인코딩한 임베딩 벡터를 비교합니다.
4. 유사도가 97% 이상인 제출을 AI가 생성한 것으로 의심합니다.

1. 단계 1에서는 다음과 같이 비교군을 생성하였습니다.
   - AI로 답안 비교군을 생성할 때 사용한 프롬프트는 다음과 같습니다:

`(대회 지문 전문)`
 	
`---`
 	
`Solve problem using C`

이후에 연달아 `use cpp`, `use java`, `use python`를 더 입력하여 총 네 개 언어의 답안 생성

   - 답안 비교군 생성에는 다음의 AI를 사용하였습니다:
     - ChatGPT 4o
     - o1 mini
     - o3 mini
     - Claude 3.5 Haiku
     - Claude 3.5 Sonnet
     - Claude 3.7 Sonnet
     - Gemini 1.5 Flash
     - Gemini 1.5 Pro
     - Gemini 2.0 Flash
     - Deepseek V3 FW
     - Deepseek R1
     - Llama 3 70b Groq
     - Llama 3.3 70b FW
     - Mistral Medium
     - Grok 2
2. 단계 2에서는 GraphCodeBERT 모델과 HuggingFace에 게시된 사전 학습된 파라미터`microsoft/graphcodebert-base`를 사용하였습니다.
3. 단계 3에서는 코사인 유사도 평가로 두 임베딩 벡터의 유사도를 평가하였습니다.

- 이 평가는 참고자료로 활용됩니다: 이 평가에 GraphCodeBERT 모델이 알고리즘 문제 해결 전용으로 파인튜닝되지 않았고, 유사도 검사 방식이 적절한 것인지 충분히 검증되지 않았습니다.

수행 결과

참여자	제출	비교군	유사도	처리
사용자 #077	92259***0	f/llama-3.3-70b-fw	0.984	AI 미사용으로 인정[1,2]
	92259***3	f/llama-3.3-70b-fw	0.984	
	92260***2	f/llama-3.3-70b-fw	0.9835	
	92259***7	f/llama-3.3-70b-fw	0.984	
	92259***4	f/llama-3.3-70b-fw	0.984	
사용자 #053	92258***9	f/llama-3.3-70b-fw	0.9727	위양성[3]
	92259***5	f/llama-3.3-70b-fw	0.9707	
	92258***3	f/llama-3.3-70b-fw	0.9709	
	92259***0	f/llama-3.3-70b-fw	0.9709	
	92253***7	c/llama-3-70b-groq	0.9726	
	92253***7	c/mistral-medium	0.973	
	92253***7	c/gemini-2.0-flash	0.9702	
사용자 #016	92256***0	c/llama-3-70b-groq	0.97	위양성[3]
사용자 #066	92255***7	c/deepseek-r1	0.9764	위양성[3]
사용자 #132	92256***7	c/deepseek-r1	0.9705	위양성[3]
	92256***7	c/gemini-1.5-flash	0.971	
	92256***7	c/gemini-1.5-pro	0.9736	AI 미사용으로 인정[1]
	92255***6	c/deepseek-r1	0.9738	
	92255***6	c/gemini-1.5-flash	0.9727	위양성[3]
	92255***6	c/gemini-1.5-pro	0.9766	
사용자 #157	92252***7	c/o1-mini	0.9754	위양성[3]
	92251***7	a/llama-3.3-70b-fw	0.9742	AI에 의해 작성됨
	92251***7	a/deepseek-v3-fw	0.9715	AI 미사용으로 인정[1]
사용자 #023	92256***5	c/deepseek-r1	0.9769	위양성[3]
	92256***5	c/gemini-1.5-flash	0.974	
	92256***5	c/gemini-1.5-pro	0.9742	
	92254***7	c/deepseek-r1	0.9762	

	92254***7	c/gemini-1.5-flash	0.9702	
	92254***5	c/deepseek-r1	0.9767	
	92254***5	c/gemini-1.5-flash	0.9717	
	92254***5	c/gemini-1.5-pro	0.972	
사용자 #187	92258***4	c/deepseek-r1	0.9778	위양성[3]
	92258***4	c/gemini-1.5-flash	0.9717	
	92254***2	b/grok-2	0.9728	
사용자 #179	92251***9	c/gemini-1.5-flash	0.9727	위양성[3]
사용자 #188	92257***1	b/gpt-4o	0.9711	위양성[3]
	92257***1	a/llama-3.3-70b-fw	0.9718	
사용자 #131	92252***1	b/gpt-4o	0.9715	위양성[3]
	92252***8	b/gpt-4o	0.973	
사용자 #094	92256***9	b/mistral-medium	0.9707	위양성[3]
사용자 #105	92260***5	b/claude-3.5-sonnet	0.9713	위양성[3]
	92260***5	b/gpt-4o	0.972	
사용자 #046	92261***5	a/llama-3.3-70b-fw	0.97	위양성[3]
사용자 #058	92259***7	a/deepseek-v3-fw	0.9716	위양성[3]
사용자 #047	92257***7	a/llama-3.3-70b-fw	0.9727	AI에 의해 작성됨
사용자 #029	92255***5	a/gemini-2.0-flash	0.9749	AI 미사용으로 인정[1,4]
	92255***5	a/deepseek-v3-fw	0.9746	위양성[3]
사용자 #140	92255***1	a/llama-3.3-70b-fw	0.9713	AI에 의해 작성됨
	92255***1	a/deepseek-v3-fw	0.9818	AI에 의해 작성됨
사용자 #124	92254***6	a/llama-3.3-70b-fw	0.98	위양성[3]
	92254***6	a/deepseek-v3-fw	0.9703	위양성[3]
사용자 #159	92251***6	a/deepseek-v3-fw	0.9811	AI에 의해 작성됨

1. AI 생성 코드의 가능성이 있으나 확실함을 보장하지 않음.

   AI 사용을 확신할 추가적인 단서가 없으므로 참여자가 직접 작성한 것으로 인정함.

2. 아래 패턴이 llama-3.3-70b-fw 모델이 생성한 답안과 매우 유사하고, 프로그램 진입점 선언에 알고리즘 문제 풀이에서는 보기 드문 `throws IOException {}`을 함께 사용했다는 점에서 AI 생성 코드의 가능성이 있음.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.*;
```

```java
public class Main {
static StringTokenizer st;
static StringBuilder sb = new StringBuilder();

public static void main(String[] args) throws IOException {
```

3. 두 코드 간 연관성 없음.

   단순히 GraphCodeBERT에 의해 생성된 임베딩 벡터가 유사한 것으로 보임.

4. 변수명, 코드 구조, 함수 사용 등에서 두 코드가 유사하나, 코드 길이가 짧아 우연에 의해 비슷하게 작성되었을 가능성이 있음.

Gemini 2.0 Flash	제출 답안

`#include <iostream>`
`#include <string>`
 	
`using namespace std;`
 	
`int main() {`
`    int n;`
`    cin >> n;`
`    cin.ignore(); // Consume the newline character`
 	
`    string slogan;`
`    getline(cin, slogan);`
 	
`    int total_count = 0;`
`    string current_number = "";`
 	
`    for (char c : slogan) {`
`        if (c == '.' || c == '|' || c == ':' || c == '#') {`
`            total_count += stoi(current_number);`
`            current_number = "";`
`        } else {`
`            current_number += c;`
`        }`
`    }`
 	
`    total_count += stoi(current_number);`
 	
`    cout << total_count << endl;`
 	
`    return 0;`
`}`
	`#include <iostream>`
	`#include <string>`
	`using namespace std;`
	`int main() {`
	`    int n;`
	`    string slogan;`
	`    cin >> n;`
	`    cin >> slogan;`
	`    int sum = 0;`
	`    string current_number = "";`
	`    for (char c : slogan) {`
	`        if (c == '.' || c == '|' || c == ':' || c == '#') {`
	`            if (!current_number.empty()) {`
	`                sum += stoi(current_number);`
	`                current_number = "";`
	`            }`
	`        } else {`
	`            current_number += c;`
	`        }`
	`    }`
	`    if (!current_number.empty()) {`
	`        sum += stoi(current_number);`
	`    }`
	`    cout << sum;`
	`}`

방법 3. 제출 데이터에 산출식을 적용하여 의심 수준 평가

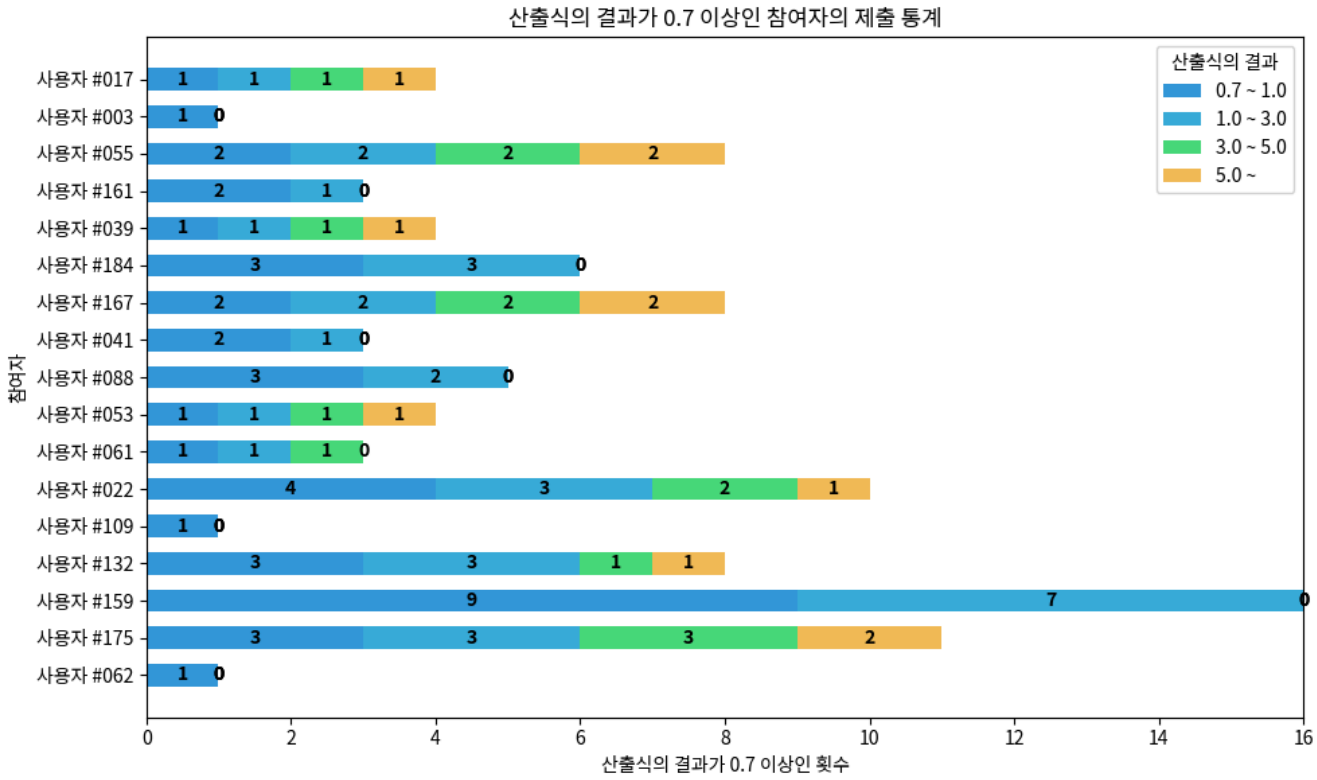
$$\text{score}(c_1, c_2) = \frac{\text{w}_{\Delta c}\ \text{len}(\Delta c)}{\text{w}_{\Delta t}\Delta t \cdot \left(w_l(c_1, c_2)^{1.5}\right)}$$

$$\begin{aligned}
c_1, c_2 &= \text{어떤 문제에 대한 연이은 두 제출} \\
\text{len}(\Delta c) &= \text{diff } c_1,\ c_2 \text{ 의 코드 길이} \\
l(c) &= \text{제출 } c\text{의 언어} \\
\Delta t &= \text{unix time diff of } c_1, c_2 \\
w_l(c_1, c_2) &= \begin{cases} 0.05 & \text{if } l(c_1) \neq l(c_2) \\ 1.2 & \text{if } l(c_1) = \text{Java} \\ 0.8 & \text{if } l(c_1) = \text{Python} \\ 1 & \text{else} \end{cases} \\
\text{w}_{\Delta c} &= 10 \\
\text{w}_{\Delta t} &= 6
\end{aligned}$$

- 이 산출식은 다음의 경우에 높은 점수를 냅니다:
  - 단일한 문제를 다양한 언어로 제출
  - 짧은 시간 안에 변경 내용이 큰 제출
- 이 산출식은 각 참여자의 문제별 제출 기록에 적용합니다.
  - 참여자 #001의 A번 제출이 제출 #1, 제출 #2, 제출 #3으로 세 건이라면 score(제출 #1, 제출 #2), score(제출 #2, 제출 #3)를 계산합니다.
  - 참여자 #002가 A번 제출 1건 제출 #10, B번 제출 1건 제출 #11로 대회 중 단 두 건의 답안만을 제출하였다면, 이 산출식을 적용하여 평가할 수 없습니다.

- 이 평가는 참고자료로 활용됩니다: 이 산출식은 가중치와 각 연산자의 적절성을 충분히 검토하지 않았습니다. 따라서 계산 결과가 높은 답안을 많이 제출하였다고 하더라도 AI를 사용한 참여자임을 보장하지 못합니다.

수행 결과



분석 결과 계산 결과가 $[0.7, 3.0]$ 구간에 위치할 경우, 그 바깥 구간의 제출보다 AI를 사용한 제출일 가능성이 조금 더 높았습니다.

이 산출식을 적용한 결과가 높은 제출 중 AI가 작성하지 않은 제출에는 이러한 유형도 있었습니다:

전 (사용자 #109, 9225***9번 제출)

```
//이거설마브루트포스~~~~?

var [n,s] = require('fs').readFileSync(0,'utf8').trim().split('\n')
n = +n.split(' ')[0]
var min = n

// †: (이하 답안 제출)

console.log(min)
```

후 (사용자 #109, 9225***2번 제출)

```
//이거설마브루트포스~~~~?
//아니 1시간동안 고민한 결과가 오타 + 입력 형식 잘못 봄 이라니.. 말이 됩니까 ㅠㅠㅠ
/*
```

```
// †: (이하 이어지는 아스키 아트와 답안)
```

†: 제출 원본에는 없으나, 일부 발췌하는 과정에서 보고서 작성자가 추가한 주석입니다.

방법 4. AI가 생성하는 것으로 널리 알려진 주석 및 함수 확인

수행 방법

- 알고리즘 문제 해결 답안 중 주로 AI가 생성하는 것으로 널리 알려진 주석이 존재하는지 확인합니다.
- 자연어 문장 주석에서 다음의 어휘가 존재한다면 AI가 생성한 것으로 의심합니다:

  1. 출력(= Print, Output)
  2. 단계(= Step)
  3. 계산
  4. (경어체 사용)
  5. 정규표현식 `(//|#)(.*?):`을 만족 (i.e. `"예:"`, `"참고:"`)

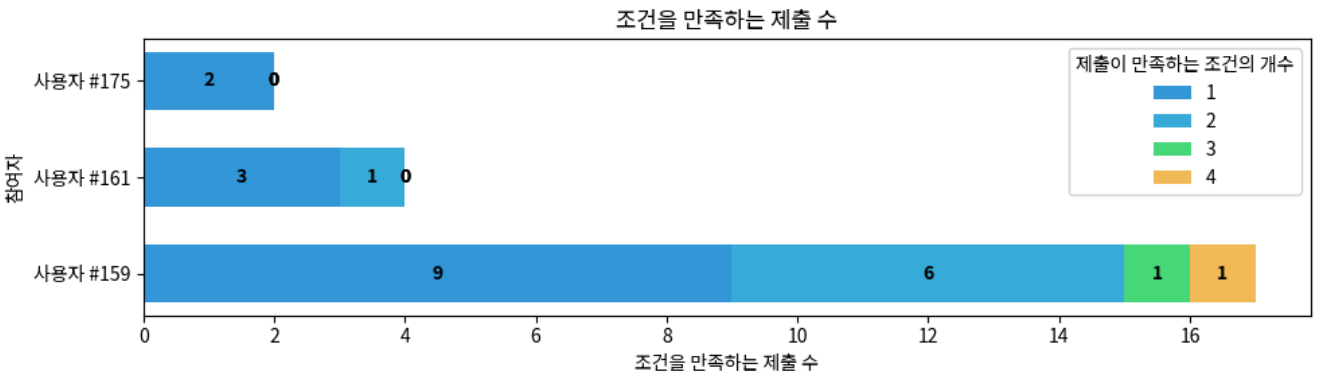
- 다음의 함수, 모듈을 사용한다면 AI가 생성한 것으로 의심합니다:

  6. C/C++: `free()`
  7. ~~Python: `re`~~†

  †: 이번 대회에서 AI가 생성한 코드의 전형적인 유형 중 하나로 파이썬의 `re` 모듈을 사용하는 경우가 많았음. 하지만 <방법 1.>에서 이미 확인하였고, 숏코딩에 널리 사용되므로 이 방법에서는 제외함.

수행 결과

참여자	계	조건을 만족하는 제출(만족하는 조건 수)
사용자 #175	2	9226***5(1), 9226***3(1)
사용자 #161	4	9225***0(1), 9225***2(1), 9225***1(2), 9225***9(1)
사용자 #159	17	9225***1(4), 9225***0(3), 9226***4(2), 9226***6(2), 9226***1(2), 9226***2(2), 9225***6(2), 9226***0(2), 9225***5(1), 9225***0(1), 9225***8(1), 9225***0(1), 9225***4(1), 9225***3(1), 9225***1(1), 9226***0(1), 9226***2(1)



이 단계에서 개인용 템플릿 코드에 포함된 자연어 주석으로 위양성으로 의심군에 포함된 제출에는 9225***0번 제출과 같은 경우가 있었습니다:

사용자 #050, 9225***0번 제출

```
// †: (윗 줄까지 답안 코드)
/* read it as if you were wrong once. --> "why is this wrong??"

 * basic strategy:
  * don't be obsessed with speed or memory when the input is small compared
to limit
  * internalization of problem statements
  * simplify. a step-by-step approach
  * readability is important
  * Do I have to solve like this?

 * stuff you should look for
  * 0-based or 1-based?
  * off-by-one error
  * int overflow, array bounds (habituation of assert and debug)
  * special cases (n=1?)
  * do smth instead of nothing and stay organized
  * WRITE STUFF DOWN
  * DON'T GET STUCK ON ONE APPROACH (feat. BFS)

 * after solving the problem
  * consider whether there is another way.
  * reduce memory, time, codes, ...
  * what is my weakness that need to be addressed by solving this problem?
*/
```

- 이 주석은 사용자 #050의 일상적인 제출에서도 지속적으로 확인되는 것으로, 개인용 템플릿 코드로 판단하고 의심군에서 제외하였습니다.