競技プログラミングにおける生成AIの進展と課題

倪　　永茂

はじめに

競技プログラミングは、プログラマのプログラミング能力と問題解決能力を評価するための重要なベンチマークとして、長年にわたりその地位を確立してきた。この分野は、アルゴリズム的発想、数学的洞察、計算量に対する理解度、コーディングの熟練度、さらには自然言語の理解といった多様なスキルを試すものであり、競技会での成功は優れたプログラマの証と見なされることが多い。

実務的なソフトウェア開発と違い、競技プログラミングでは、短時間（数分から数時間）でプログラムを作成し、数秒以内の実行時間内に大量のテストデータに対して正しい出力を返さなければならない。個々の問題に正解があることは特徴的である（ちょうど日本の大学入学共通テストのような試験問題）。作成したプログラムの長さは数行から数百行と比較的短く、競技後に再利用されることは少ない。一方、実務的なソフトウェア開発では、顧客のニーズを分析するのに時間がかかることも多く、プログラムの完成までに数か月を要するのが一般的である。実行時間の制約はあるものの、それ以上にプログラムの可読性や保守性が重視される。プログラムの長さはソフトの規模によるが、数千行から数百万行と幅が広い。正解についても常識的許容範囲であれば、それほど厳しく判定されることは少ない。

それでも競技プログラミングがもてはやされる理由は、短時間でプログラマの能力を客観的かつ効率的に評価しやすいこと、ゲーム性があること、そして自身の技術力を示す達成感や、それを他者と共有できる手段になるからである。競技プログラミングに取り組むプログラマをスポーツ選手にたとえると、その本質がより理解しやすい。

競技プログラミングについての日本の実態を確認してみよう。AtCoderが競技プログラミングプラットフォームとして有名であり、プログラマの初心者から上級者まで幅広い層に支持されている。週末には定期的にコンテストが開催され、国内外から多くの参加者がインターネット経由で集まり、競い合う。AtCoderでは、参加者の実力をレーティング（Rating）で数値化し、色分けでグループ化している。2024年８月時点での参加者のレーティング分布はつぎのように推定される。

灰色（～399，71.8％）、茶色（400～799，12.49％）、緑色（800～1199，7.79％）、水色（1200～1599，4.16％）、青色（1600～1999，2.23％）、黄色（2000～2399，1.10％）、橙色（2400～2799，0.24％）、赤色（2800～，0.11％）。

実務的ソフトウェア開発プログラマと青色の競技者との違いについて、AtCoder社の公式な見解を借りて紹介すると、「青色所有者は普通のプログラマから見て、常軌を逸したコーディング速度を持ち、複雑なロジックにおいてもバグの少ない安定したロジック構築が可能となる。大抵の業務においてはここまでのレーティングは必要としないが、例えば、大きなデータを扱う・数理的な処理を扱う・研究開発に近い業務を行う・論文などをキャッチアップして実装までするような業務を行う場合は、大きな戦力となる。競技者としては、旧帝大や早慶などの、競技プログラミング強豪校を除いた、大学のエース選手がこの水準である。よほど関連した業務などを行っていない限りは、競技プログラミングに関する練習なしにこの水準のパフォーマンスが安定して出せるプログラマはいないと言っても良いだろう」。

ところで、近年、人工知能、特に大規模言語モデル（LLM）を含む生成AIの分野では目覚ましい進歩がみられ、さまざまな領域でその能力を発揮してきた。コード生成もその一例である。驚く

べきことに、生成AIは競技プログラミングにおいても優れた成績を収め始め、その競争力のあるレベルのパフォーマンスは注目を集めている。本文では、生成AIが競技プログラミングにおいて、これほどの成績を上げている理由と根拠を多角的な視点から分析し、プログラマと比較しながらその強みと弱みを明らかにする。また、技術の進歩や応用範囲の拡大といった観点から、生成AIの将来的な展望と限界についても考察し、倫理的課題や懸念事項に言及する。

近年、AIが競技プログラミングにおいて目覚ましい進歩を遂げている事実は、従来のAIによる問題解決が比較的単純な数学やプログラミングの問題、あるいは既存の解決策の検索とコピーに限定されていた状況からの大きな転換を示している。競技プログラミングにおける明確な課題文と客観的な評価基準は、AIがソリューションを理解し、考案し、実装する能力を測定するのに理想的な環境でもある。

Ⅰ 競技プログラミングにおける生成AIの成果

ここでは、競技プログラミングにおける生成AIの台頭と成果、生成AIモデルのアーキテクチャとその学習方法、生成AIが解答するのに得意とする問題とそうでない問題に分けて、それぞれみることにする。

（1）競技プログラミングにおける生成AIの台頭

生成AIが競技プログラミングにおいても注目されるようになった経緯について振り返る。

① DeepMindのAlphaCode

DeepMindのAlphaCodeは、公式ブログおよび論文公開を経て、2022年２月２日に正式に発表された。これは、生成AIを用いて競技プログラミングの問題を人間レベルで解くことを目指した初の大規模モデルであり、「プログラムコードを書くAIが競技プログラミングに挑戦できる」という新しいマイルストーンを打ち立てた。

2020年～2021年にかけて、OpenAIのCodex（GitHub Copilotの基盤）や、GoogleのCodey、FacebookのCodeGenなど、自然言語からプログラムを生成するモデルの研究が急速に進展した。AlphaCodeはその延長線上にありつつ、より厳密な競技環境（例：AtCoder、Codeforces）での性能を強く意識した構成になっている。その実力を確認したのはCodeforcesという世界的に有名なプログラミング競技プラットフォームにおいて、AlphaCodeは参加者の上位54％に相当する推定ランクを達成したことによる。これは、AIコード生成システムがこのような競技会で競争力のあるレベルに達した最初の事例とされる。AlphaCodeは、批判的思考、論理、アルゴリズム、コーディング、自然言語理解を組み合わせた新しい問題解決能力を示した。推定されるCodeforces Eloレーティングは1238であり、コンテストに参加したユーザーの上位28％に位置づけられる。また、Codeforcesにおいて評価1800の問題を含む、より難しい問題にも正解している。シミュレーション評価では、不正解と判定された後の修正作業等を含めた10回までの提出で約30％の問題を解決した。

AlphaCodeの示した意義として、「コードを書くAI」から「競技で戦えるAI」へというジャンプを見せたこと、従来のAIは「与えられた仕様に従うだけ」であったが、AlphaCodeは自然言語で与えられた複雑な課題文を理解してコードに変換することに成功したこと、「人間レベルのアルゴリズム的思考」を模倣する第一歩として、後の研究への礎となったことを挙げられる。

② CodiumAIのAlphaCodium

CodiumAI（現Qodo）が開発したAlphaCodiumは、競技プログラミングにおけるコード生成の精度と信頼性を向上させるために設計された、オープンソースのAIコード生成フレームワークである。2024年１月に発表されたこのツールは、従来のプロンプトエンジニアリングを超えた「フローエンジニアリング（Flow Engineering）」という新しいアプローチを採用し、AIが複雑なコード問題を効果的に解決できるよう支援することができるようになった。AlphaCodiumは、Code Contestsデータセットにおいて、GPT-4の正答率（pass@5）を直接的なプロンプトのみを使用した場合の19％から44％へと大幅に向上させた。AlphaCodeと比較して、LLMの呼び出し回数が大幅に少ないにもかかわらず、同等の性能を示し

た。

③ OpenAIのGPT-o1モデル

OpenAIのGPT-o1モデルは、2024年末～2025年初頭に登場した「新世代の小型モデル」のひとつで、特に軽量かつ応答性に優れた設計がなされたモデルである。このモデルは、Codeforcesの競技プログラミング問題において、89パーセンタイルにランクインしている。さらなるトレーニング後には、Codeforcesレーティング1673（89パーセンタイル）を達成した。さらに、GPT-o1-miniはコーディングに関するSEALリーダーボードで1271のレーティングを獲得している。ある報告によれば、Codeforcesのユーザーがo1-miniをライブコンテストで使用し、マスターレベルに近いパフォーマンスを達成したという。

④ OpenAIのGPT-o3モデル

OpenAIのGPT-o3モデルは、2024年12月20日に発表され、2025年４月16日に正式リリースされた。このモデルは、前身であるGPT-o1の後継として開発され、特に高度な推論能力と複雑な問題解決能力を強化することを目的としている。その設計には、強化学習を用いた「思考の連鎖（private chain of thought）」アプローチが採用されており、AIが回答を生成する前に内部で段階的な推論を行うことで、より正確で論理的な応答を実現している。GPT-o3は、Codeforcesで2724というレーティングを達成し、これは上位0.2％に相当する。2024年国際情報オリンピック（IOI）では395.64点を獲得し、金メダル獲得に必要な基準を上回った。

このように、AlphaCodeからGPT-o1、さらにGPT-o3へと進化する過程を通じて、生成AIが競技プログラミングの課題に取り組む能力を急速に向上させてきた。当初は中央値レベルの競争力にとどまっていたが、比較的短期間でトップレベルに近いパフォーマンスを示すに至った。また、AlphaCodeのサンプリングとフィルタリング、AlphaCodiumのフローエンジニアリングといった、異なるアプローチによる技術開発が進められており、競技プログラミングにおけるAIの問題解決能力向上に向け、多様な技術が存在することが明らかとなった。

（２）生成AIのアーキテクチャと学習方法

競技プログラミングで成功を収めている生成AIモデルは、特定のアーキテクチャと学習方法を採用している。

・Transformerベースのアーキテクチャ

特にエンコーダー・デコーダーモデルを含むTransformerネットワークの使用は、コード生成のようなシーケンス・ツー・シーケンスのタスクにおいてその有効性が証明されている。エンコーダーは問題記述を処理し、デコーダーはコードソリューションを生成する。

・大規模なコードデータセットでの事前学習

GitHubのような膨大なコードリポジトリでモデルを事前学習することは、様々なプログラミング言語の構文、意味論、および一般的なパターンを学習するために重要である。

・競技プログラミングデータセットでのファインチューニング

事前学習されたモデルを、CodeContestsのような競技プログラミングに特化したデータセットでファインチューニングする。CodeContestsには、Codeforces、CodeChef、HackerEarthなどのプラットフォームからの問題が含まれており、多くの場合、入力と出力のペア、および正誤両方の人間の解答が含まれている。

・大規模なサンプリングとフィルタリング

各問題に対して大量の候補ソリューションを生成し、提供されたテストケースに合格するかどうかに基づいてフィルタリングするなどの手法を採用している。

・ソリューションのクラスタリングと再ランキング

クラスタリングアルゴリズムを使用して意味的に類似したソリューションをグループ化し、これらのクラスターを再ランキングして、有望な候補の小さなセットを提出用に選択する方法を採用している。

・強化学習（RL）

OpenAIのGPT-o1やo3のようなモデルのトレーニングにおける強化学習では、モデルは問題解決の成功に対する報酬を通じて、推論プロセスと戦略を洗練させることを学習する。o1は思考の連鎖を使用し、RLを通じてこのプロセスを磨くこ

とを学習した。o3は、手作りの推論ヒューリスティックに頼らずに、大規模な強化学習とテスト時の計算能力をスケールアップすることで、最高のパフォーマンスを達成した。

・フローエンジニアリング（AlphaCodiumの場合）

巧妙なプロンプトエンジニアリングと反復テストを通じて、AIの問題解決プロセスを注意深く導くアプローチを採用している。

生成AIが競技プログラミングで成功しているのは、強力なTransformerアーキテクチャ、関連性の高いデータでの広範なトレーニング、およびソリューション空間を探索し、正しさを保証するための高度な手法の組み合わせに大きく依存している。事前学習されたTransformerから、洗練されたサンプリングおよびフィルタリングメカニズムを備えたファインチューニングされたモデル、そしてより最近では強化学習とフローエンジニアリングの統合への進化は、競技プログラミングの複雑な要求に取り組むための多面的なアプローチを示している。厳密なテストと正誤両方のソリューションを備えたCodeContestsのような特殊なデータセットの開発は、この分野におけるAIモデルのパフォーマンスをトレーニングおよび評価する上で非常に重要であった。競技プログラミングの課題を正確に反映し、包括的な評価基準を提供する高品質のデータセットは、この分野におけるAI能力の限界を突破するために不可欠であろう。

（3）生成AIが得意とする競技プログラミング問題の特徴

生成AIモデルが特に高い性能を発揮する競技プログラミングの問題には、以下に示すように、いくつかの共通する特徴が見られる。

① 明確かつ十分に定義された仕様と制約を持つ問題

生成AIは「言語モデル」であり、入力されたテキストから最も確からしい出力を予測するよう設計されている。仕様や制約が曖昧でない場合では、AIは問題の意図を誤解せずに論理的な手順を構築できる。例えば、数学の問題やフォーマルな仕様は、AIにとって「言語的に解釈が一意に定まる」ため、構文解析・パターン照合が容易で、正確な解法に導きやすい。

② より小さく、管理しやすい部分問題に分解できる問題

生成AIは複雑な課題に対して「一括で正解を導き出す」よりも、「段階的に小さなステップに分けて処理」する方を得意としている。問題を部分問題に分解すれば、AIは段階ごとに推論を重ねて、誤りを避けつつ正解にいたる。これはAIの自己完結的な一問一答スタイルにもマッチする。

③ レーニングデータに存在し、しかも標準的なアルゴリズムとデータ構造を使用する問題

AIは「既に見たことがある問題の構造」に強く依存する。標準的なアルゴリズム（例：二分探索、DFS、Union-Find、Dijkstraなど）は、大量の公開コードや教材に含まれており、モデルの学習過程で繰り返し登場する。記憶的再構成により、既知の手法に即して正確な解法を生成できるため、標準的な問題に対して特に高精度な回答が可能である。大学での情報系シラバスではこういう標準的なデータ構造や標準的なアルゴリズムが学修対象になるため、生成AIがそれらの練習問題にも得意である。

④ 多数の解答方法を効率的に生成およびテストできる問題

生成AIは「一度で最適解を出す」よりも、「候補を多く出し、その中から良さそうなものを選ぶ」という使い方に適している。答えの候補が多く存在するような問題（例：全探索、貪欲、構築系）では、AIが複数の選択肢を提案→検証という過程を高速に繰り返すため、精度が向上する。これは多様なサンプル生成能力を活かせるケースでもある。とくに、競技プログラミングでは正解があるため、正解と照合しながら、解答方法を逆に推定する技法もAIが活用している。

⑤ テキストによる説明に基づいてコードを合成および実装する必要がある問題

生成AIは言語モデルであり、自然言語→コード変換に特化した訓練を受けている。課題文の中に詳細な動作手順や条件が記載されている場合は、AIはそれを逐語的にコードへマッピングする。つまり、自然言語による「擬似コード」のような説明があればあるほど、AIはそのまま正しいコードを出力しやすくなる。

⑥ 一般的に知られていない高度なアルゴリズムを必要とする問題

一見するとAIが苦手そうに思われるが、実は「高度であっても有名なアルゴリズムであればトレーニングデータに含まれている」ことが多く、驚くほど正確にコードを出せる場合が多い（例：セグメント木、重み付きUnion-Find、2-SATなど）。また、こうした問題は定型的な解法に落とし込まれることが多く、課題文が長くても「変更すべきパターン」が少ないため、AIがパターン認識を通じて正解に到達できるのである。

このように、生成AIが得意な問題のタイプは

問題の構造（明確性、分割性）
解法の再利用性（標準アルゴリズムの有無）
訓練データとの親和性（頻出度）
AIの生成・評価能力（候補の並列生成）

といった観点から、AIの得意不得意を示唆してみた。

（4）生成AIが苦手とする競技プログラミング問題の特徴

一方で、現在の生成AIにとって依然として課題となる問題も以下のように存在する。

① 高度な創造性、直感、または型破りな発想を必要とする問題

生成AIは「過去の大量の事例（トレーニングデータ）に基づく統計的予測モデル」であり、創造的な飛躍や直観的な発見が求められる状況に弱い。競技プログラミングでは、意外な構造変換や制約条件の変形（例：非自明な状態遷移の発見など）が求められる問題があり、これらは過去のパターンをなぞるだけでは到達できないため、AIの限界が露呈しやすい。

具体的な問題例として、ICPC 2023 World Finals Problem A: Riddle of the Sphinx（https://icpc.global/world-finals/problems/2023-ICPC-World-Finals/icpc2023.pdf）を取り上げる。本問題は未知の生物３体がそれぞれのもつ脚の数を、５回の質問とその回答から正確に推定する内容である。ただし、５回回答のうち、１回は嘘である可能性がある。さて、未知生物の脚本数をそれぞれx, y, zとすると、最低３回の質問は必要であることが直ちにわかる。その３回の答えのうち１回は嘘であることを想定し、残りの２回の質問で嘘である回答を特定しないといけないが、AIは、嘘の可能性を考慮した推論や、限られた情報からの創造的な推測が苦手であり、解答するプログラムをうまく作成できないことが多い。

② 曖昧または十分に定義されていない問題

AIは言語の曖昧性を完全には解消できないため、仕様が明示されていない、または例外条件が曖昧な問題では誤解や誤実装が起こりやすい。とくに競技プログラミングでは、数文字の文言の違いが挙動を大きく左右するため、微妙な表現の差を読み取る力が問われるが、AIはこのような暗黙の仕様の読み取りが不得手である。

たとえばつぎの問題をみてみよう。AtCoder Beginner Contest 298 C - Cards Query Problem（https://atcoder.jp/contests/abc298/tasks/abc298_c）は複数の箱とカードに対する操作とクエリを処理するもので、操作の種類や出力形式に注意を必要とする。課題文では、「カードを１枚選んで数ｉを書き」とあるが、「同じ数ｉを書いたカードを複数作ることができるか」「同じ箱に複数入れてもよいか」は明示されていない。実際にはサンプルから「同じ数を複数枚のカードに書いて複数の箱に入れる」ことが許容されていると読み取れるが、これはサンプルからの類推に依存しており、AIは明示されない仕様を誤って一般化する危険がある。また、出力における並び順について「昇順で答える」という指示はあるが、たとえば「同じ数のカードが同じ箱に複数枚入っていた場合、出力には１つだけ出すのか複数回出すのか」、「箱番号の昇順とカード番号の昇順のような異なる基準がある場合、どちらを優先するか」といった判断は、用語の一語一句の違いに依存するため、AIにとって間違いやすい。つまり、一見すると単純なシミュレーションに見える問題でも、数と箱の双方向関係、多重性の扱い、出力の並び順の厳密さといった点で、AIが誤解しやすい「仕様の曖昧性」を内包しており、「人間が行間を読む力」と「細部の言葉の違いに敏感であること」が問われる好問である。

③ トレーニングデータにおける一般的なパターンから大きく逸脱する新規な問題

生成AIの強みは「よくある問題の再現」にあ

り、新奇性や独自構造を持つ問題には訓練されたパターンが存在しないため、対応が難しくなる。特に、AtCoderやCodeforcesのDiv.1後半問題などでは、これまでに見たことのない構造や制約が提示され、パターンマッチングが不可能になるため、AIは手がかりを見つけられず、誤った推論に陥りやすい。

AtCoder Beginner Contest 353 D - Another Sigma Problem（https://atcoder.jp/contests/abc353/tasks/abc353_d）を具体例として取り上げる。本問題は数列の要素を文字列として連結し、特定の演算を行うものであるが、通常の数値計算とは異なる処理が求められる。課題文のなかでは、関数 f（x, y）を「十進表記の x と y を文字列として連結し、それを整数として解釈した値」と定義しているが、関数は非対称性をもち、また、数値を文字列に変換し連結した後、再び数値に変換するという操作を含む。これは、通常の数値演算とは異なる処理であり、AIが数値演算のパターンに基づいて解こうとすると誤りを生じてしまう。つまり、本問題は生成AIが過去のパターンに基づいて解こうとする際に直面する課題を浮き彫りにするもので、非対称な関数定義、文字列と数値の変換、桁数に基づく処理など、従来のパターンに当てはまらない要素が多く含まれている。

④ 広範な現実世界の知識や常識的な推論を必要とする問題

生成AIは「言語情報に特化」しており、物理的な知識や人間の常識、日常的な感覚に基づく推論には限界がある。例えば、「確率を用いた現実的な意思決定」や「効率的なスケジューリング」など、抽象的でありながら常識的な最適化を求める問題では、明示的なルールが存在しないとAIは適切な推論を行うことが困難である。これは、直感的な判断をAIが代替できないことを意味する。

具体的な例として、ICPC 2021 World FinalsのProblem F: Islands from the Sky（https://icpc.global/worldfinals/problems/2021-ICPC-Worldfinals/icpc2021.pdf）を取りあげる。本問題では複数の島々と飛行経路が与えられる。各飛行経路は３次元空間上の直線で表され、飛行機に取り付けられたカメラは、飛行経路に垂直な方向に一定の角度 θ で地表を撮影する。本問題の出力として求めることはすべての島が少なくとも１つの飛行経路によって完全に撮影されるような最小の角度 θ を算出することである。

本問題ではカメラの視野角や飛行経路の高度、島の位置関係など、物理的な要素が複雑に絡み合っている。AIはこれらの要素を直感的に理解し、適切なモデルを構築することが難しい。さらに、飛行経路は３次元の直線、島は２次元平面上の多角形として表され、３次元空間での幾何学的な計算が求められる。AIがこれらの空間的な関係を正確に把握し、必要な計算を行うことは困難である。言い換えると、本問題は生成AIが苦手とする物理的直感と３次元幾何学の組み合わせを含み、人間の直感や経験に基づく判断が求められる。

⑤ 非常に厳しい時間またはメモリ制約があり、高度に最適化された、自明ではない解決策を必要とする問題

生成AIは「正しそうな解法」を出すのは得意でも、「最適で高速な解法」を設計する能力には限界がある。特に、指数時間・空間から多項式時間への削減（例：枝刈り、前処理の工夫）が必須となる問題では、AIはしばしば間に合わせの線形解法や無駄の多い実装を出力してしまう。こうした非自明な最適化戦略（計算量削減手法）の発見には、現時点のAIモデルはまだ対応しきれていない。

AtCoder Beginner Contest 294 G - Distance Queries on a Tree（https://atcoder.jp/contests/abc294/tasks/abc294_g）を具体例として確認すると、本問題は木構造に対する距離クエリと辺の重みの更新を効率的に処理するものである。効率的に解くためには、つぎのような高度なアルゴリズム、すなわちEuler Tour（木を線形に表現し、区間クエリを可能にするアルゴリズム）、セグメントツリーまたはFenwick Tree（区間の合計や更新を高速に処理するアルゴリズム）、Lowest Common Ancestor（LCA）（２つの頂点間の最小共通祖先を高速に求めるアルゴリズム）を組み合わせる必要がある。しかし、生成AIはこれらの最適化手法を自発的に選択することが難しく、しばしば単純な線形探索や再帰的な実装を提案して

しまい、計算量の制約を満たせないことになる。

⑥ 複数のコンポーネント間の複雑な相互作用を含む問題、または高レベルの設計視点を必要とする問題

AIは「単一の機能を記述する」のは得意であるが、「複数の構造や条件が絡み合う問題設計」になると、全体像の把握や相互依存の追跡が困難になる。たとえば、複数フェーズ（例：構築→検証→更新）の問題や、状態遷移が複雑に絡むマルチモジュール型の問題では、設計ミスや状態管理の漏れが起こりやすくなる。これは、AIがグローバルな設計力を欠いていることの現れでもある。

具体的な問題例として、ICPC 2020 World Finals Problem B: The Cost of Speed Limits（https://docs.icpc.global/icpcMoscow.pdf）を示す。本問題は道路網の速度制限標識の設置と速度制限の調整に関する最適化問題であり、全体の設計とコストのバランスを考慮することが求められる。道路の速度制限を改良することで、交差点での速度制限の違いが解消され、標識の設置が不要になる場合がある。しかし、どの道路を改良すべきか、どの程度改良すべきかを判断するには、全体の状態を把握し、最適な遷移を計画することが求められる。生成AIは、これらの状態遷移を適切に管理することが困難である。また、速度制限標識の設置コストと道路の改良コストのトレードオフを考慮し、総コストを最小化する必要があるが、生成AIはこれらのトレードオフを定量的に評価し、最適な選択を行うことが難しい。

⑦ 最適解ではなく、現実的近似解を優先する問題

計算機科学では、解くのに天文学的時間（何十億年や宇宙存在してきた以上の時間）を必要とする問題は無限に存在する。その代表格は巡回セールスマン問題（TSP、Traveling Salesman Problem，すべての都市を１回ずつ訪れて、最短で一周する経路を求める問題）、ナップサック問題（0-1 Knapsack Decision Problem、重さに制限がある袋に、価値と重さがある品物を選んで価値を目標値以上にできるかという問題）、グラフ彩色問題（Graph Coloring、隣接する頂点が異なる色になるように、最小の色数で頂点を塗り分ける問題）、充足可能性問題（SAT, Boolean Satisfiability Problem、真偽値を与えて、論理式が真になるようにできるかを判定する問題）、部分集合和問題（Subset Sum Problem、与えられた数の部分集合の和が、ちょうど指定された数になるかどうかという問題）、頂点被覆問題（Vertex Cover、すべての辺が少なくとも１つの選ばれた頂点と接するように、頂点集合を最小化する問題）、独立集合問題（Independent Set、互いに隣接していない頂点の集合のうち、最大のものを求める問題）、ハミルトン路問題（Hamiltonian Path / Cycle、すべての頂点を１度ずつ通るような経路（または閉路）が存在するかを問う問題）、集合被覆問題（Set Cover Problem、すべての要素をカバーするために、部分集合群から最小数の集合を選べるかどうかという問題）、ジョブスケジューリング問題（Job Scheduling Problem、複数の仕事を制約内で割り当て、全体の完了時間を最小にできるかどうかという問題）などなど、無限に作り出すことができる。これらの問題はいずれも、一般には非常に難しく、最適解の計算には膨大な時間がかかる。しかし、競技プログラミングでは、入力サイズや制約を工夫することで、数秒で解けるように調整された限定的なケースが出題される。ただし、こうした問題設定は現実の世界とは異なり、実用的な意思決定や現場の判断とはかけ離れていることも多い。たとえば巡回セールスマン問題では、現実にはセールスマンが数百の顧客を回ることもあり、最適経路を厳密に求めてから移動することはまずない。おおまかな順序や経験則に基づいて行動しているのが実情であろう。

つまり、NP完全・NP困難な問題では、最適解の探索に指数的な時間がかかるため、生成AIも全探索的なアプローチに陥ってしまい、計算時間やメモリ制約で実行不能になることが多い。一方、人間は制限時間内で現実的な近似解やヒューリスティクスを柔軟に選び、実用性を優先する判断を下す。生成AIにはこのような「妥協と納得」の発想が欠けやすく、問題構造を深く理解して適切に近似する能力が大きな課題となっている。

上記７つの項目は、生成AIの弱点をそれぞれ異なる視点から捉えたものであり、表１にまとめた。

表１　生成AIが苦手とする問題の分類

主な原因	特徴的な困難
想像力の欠如	パターン外の発想に弱い
曖昧さの解釈力不足	高度な読解力に対応できない
訓練外の知識欠如	新規構造に対応不能
常識知識の非内包	非数式的推論が困難
計算量意識の欠如	時間・空間効率の意識が甘い
設計統合力の弱さ	複雑系の全体設計に対応不能
妥協と納得の欠如	現実的近似解を出しにくい

以上の分析を整理すると、生成AIの強みは、膨大なデータセットから学習したパターンに基づいてコードを処理および生成する能力にあり、認識可能な構造と解決策を持つ問題に適している。AlphaCodeが、確立されたパターンを持つアルゴリズム問題が多数存在するCodeforcesのようなプラットフォームで成功していることは、AIがトレーニングデータを活用して既知の解決策またはその変形を特定し、実装する際に優れていることを示唆している。対照的に、競技プログラミングにおける現在の生成AIの限界は、多くの場合、真の創造性と、新規性または曖昧さを処理する能力の必要性から生じる。これらは、人間のプログラマが現在優位に立っている領域でもある。競技プログラミングでは、人間の創意工夫を特に試すように設計された問題が多く、標準的なアルゴリズムやパターンからすぐに明らかにならない解決策を必要とする。AIが既存のデータに依存すればするほど、そのような問題に苦手である。

（５）競技プログラミングにおける生成AIの実力検証

2025年５月現時点における生成AIの能力を検証するため、いくつかの競技プログラミングコンテスト、およびそれに類する問題環境において、AIに実際の問題を解かせる実験を実施した。本検証では、AIによる問題解答の正答率および所要時間に注目し、実用性や限界を評価することを目的とした。

誤答と判断された場合には、可能な限り時間の許す範囲でAIに対して再修正を指示し、再試行を行った。これにより、AIが自動的に自己修正できる範囲や限界も併せて観察した。

本検証では、OpenAIが提供する「ChatGPT Plus」プランのうち、o4-mini-hige モデルを主に使用した。理論上、o3やGPT-4.1など他のモデルも選択可能であったが、著者の過去の経験に基づくと、競技プログラミングのような構文精度と論理展開が重視される問題においては、o4-mini-high系列モデルが最も安定して高い精度を示すという主観的判断により、このモデルを選定した。

各課題文は、AIが正確に読み取れるようMarkdown記法に従って整形し、プロンプトに入力した。問題文に数式が含まれる場合には、LaTeX表記により明示的に記述した。さらに、グラフや画像を含む問題については、Microsoft Wordに課題文全体を入力し、対応する図や画像を貼り付けた上で、AIにWordファイルとして添付・読込させた。

AIが出力したプログラムは、原則として人手による修正を一切加えず、そのままコンテストの提出サイトにアップロードした。判定結果（正解／不正解）は、各サイトの自動ジャッジシステムに基づいて記録された。

また、本実験はリアルタイムでのAI使用が禁止されている多くのコンテストのルールを遵守するため、終了後の「バーチャルコンテスト」形式で実施した。これは、当該コンテストの実施終了後に提供される問題アーカイブを用い、あたかも本番中に参加したかのような仮想的条件下でAIに問題を解かせる形式である。

検証実験１　AtCoder Beginner Contest（ABC）

日本国内外で高い評価を受けているAtCoder社の競技プログラミングコンテストは、その出題形式に応じて大きく二つのカテゴリーに分類される。すなわち、正解が明確に定義された「アルゴリズムコンテスト」と、正解が存在しない、あるいは計算上求めるのが極めて困難な「ヒューリスティックコンテスト」である。アルゴリズムコンテストはさらに参加者の熟練度に応じてつぎの３つに分類されている。

AtCoder Beginner Contest（ABC）。初学者向けに設計され、比較的平易な問題が出題される。競技プログラミング未経験者や学習初期の参加者

を主な対象としている。

AtCoder Regular Contest（ARC）。中級者向けの難易度で、典型的なアルゴリズムだけでなく、応用力や発想力も問われる問題が出題される。

AtCoder Grand Contest（AGC）。上級者を対象とし、計算量・データ構造・アルゴリズム設計のすべてにおいて高度な知識と実装力が求められる。

これらはいずれも「正解が定義された問題」に対して、論理的かつ効率的な解法を構築する能力を競うという共通点を持つ。これに対し、ヒューリスティックコンテストでは最適解の計算が現実的に不可能な問題が出題され、参加者は限られた時間内で「より良い」解を探す必要がある。評価は絶対的な正解ではなく、提出された解のスコア（品質）によって行われるため、探索的・創発的アプローチが求められる。この分野は特にAIや機械学習技術の応用とも相性が良いが、検証目的としてはアルゴリズムコンテストの方が明確な判定がしやすい。

主要なアルゴリズムコンテストの開催情報はつぎの通りである。すなわち、ABCはほぼ毎週土曜21時に開催され、出題数７問、コンテスト時間は100分、参加者数は１万人規模である。ARCは日曜21時に開催されることが多く、出題数は４～５問、コンテスト時間は120～150分、参加者数は500～1500人規模である。AGCは不定期に開催され、出題数は５問、コンテスト時間は180分、参加者数は100～300人規模である。

このように、参加者のレベルに応じてコンテストの難易度や構成が明確に差別化されており、生成AIによる問題解決の検証においても、どのコンテストを対象とするかは重要な選択となる。

ABCは初級者向けとされているが、実際には非常に高い完成度が求められる問題も含まれている。特に、ABCにおいて最高レーティングである青色（2000点）に到達することは容易ではなく、ソフトウェア開発業務の即戦力として通用するレベルを超え、IT企業でも重宝される水準とされている。一方、ARCやAGCといった中上級者向けの問題は、現時点の生成AIにとっては問題の読解や解法戦略の構築が難解であり、不正解率が極めて高いという実態がある。したがって、本検証においては、AIの実力を最も妥当かつ客観的に測ることができる対象として、ABCに焦点を当てることにした。

本検証では、単発の成績では偶然性が排除できないため、過去問として公開されている直近５回分のABCコンテストに連続してバーチャル参加を行い、AIの安定した実力を評価する形式を採用した。参加方式は実際の開催時間と同様の制約条件のもとで再現し、出題内容をAIに提示してそのままコードを生成・提出する形で実施した。

各回の解答時間、正解数、推定順位などの参戦結果を表２に整理して提示しており、そこからAIモデルの実力を読み取ることができる。問題７問のうち、最後の１問は解けたり解けなかったりしていた。

表２　ABCの生成AI成績

ABC	使用時間、成績および推定順位
402回目	43分、**６問正解**、1975点、 推定順位65位
403回目	50分、７問全問正解、2675点満点、 推定順位10位
404回目	71分、７問全問正解、2675点満点、 推定順位37位
405回目	100分、**６問正解**、2000点、 推定順位250位
406回目	83分、**６問正解**、2000点、 推定順位270位

５回連続のAtCoder Beginner Contest（ABC）において、平均６問正解、２回は全問正解（７完）という結果は、非常に高いレベルのパフォーマンスといえよう。

注目すべきポイントとして、全問正解を２回達成したこと、ABC403・404では満点を獲得し、それぞれ推定順位10位と37位となったことである。これはAIが、典型問題からやや難易度の高いアルゴリズム問題まで安定して解けることを示している。また、未正解問題は一貫して最後の１問のみ、生成AIが高難度問題に若干苦戦する傾向はあるものの、それ以外は上位層に匹敵する安定感を持っている。

課題として、難問での詰まりが顕著であり、問題構造の把握や高度な考察が必要なプログラム設

計が必要なタイプにやや弱いことがあげられる。今回の検証対象はABCであり、より難度が高く思考力を問われるARCやAGCでは、新規アルゴリズム設計力や抽象化能力の壁がAIには依然残ると考えられる。

検証実験２　AtCoderアルゴリズム実技検定（PAST）

PAST（Programming Algorithm Skill Test）は、競技プログラミングプラットフォームを運営する株式会社AtCoderが実施している検定試験である。本検定は、アルゴリズムやプログラミングに関するスキルと知識の両面を測定し、実践的な問題解決力を可視化することを目的としており、日本国内で初めての本格的なアルゴリズム能力認定試験とされている。

試験時間は５時間と長時間にわたり、受験形式はオンラインであるため、受験者は自宅などの任意の環境から参加できる。これは通常の競技プログラミングコンテストに近い環境設定といえる。試験終了後、得点と認定ランクが即時にフィードバックされ、同時に公式認定証が発行される。これにより、受験者は自らのアルゴリズム能力を明確な形で証明することが可能となっている。

試験は15問構成で、各問題には個別の得点が割り当てられており、合計100点満点で評価される。取得得点に応じて、以下のような５段階のランクに分類される。エントリー（25〜39点）、初級（40〜59点）、中級（60〜79点）、上級（80〜89点）、エキスパートランク（90点以上）。

この評価方式は、受験者の絶対的な能力水準を示すものであり、他の受験者との相対的な順位ではなく、個人の技術レベルを客観的に判断することに重点が置かれている点が、通常のコンテストとの大きな違いである。

特に注目すべきは「上級」および「エキスパート」ランクの定義である。上級は、単にアルゴリズムの基本構造を理解しているだけではなく、実装対象の要件に応じて適切なアルゴリズムを選定し、安定してコードとして実装できる能力があることを示す。一方、エキスパートはさらに高度な能力を要する。すなわち、アルゴリズムの計算量を意識したうえで、時間効率や空間効率の最適化を図る能力を備えていることを意味しており、研究開発や業務応用にも通用する水準とされる。

なお、エキスパートランクの取得者は全体の数％にとどまるとされており、その取得難易度の高さがうかがえる。

筆者は、PASTの評価制度の有効性とAIの能力検証を兼ねて、第15回から第19回までの５回分の検定試験にバーチャル参加を行った。これらは全て公式に問題が公開された後の模擬実施によるものである。参加形式としては単回受験ではなく、連続して複数回に参加することで、より安定した実力を反映させることを重視した。

各回の受験結果、解答時間、および取得したランクは表３にまとめてあるが、いずれの回においてもすべてエキスパートランクを獲得しており、極めて安定した高成績を収めた。また、５回合計で解いた問題数75問中、不正解となったのはわずか１問のみであり、これはAIモデルの安定性と問題適応力の高さを裏付ける結果であろう。

表３　PASTにおける生成AIの成績

PAST	使用時間、成績およびランク
15回目	３時間47分、15問全問正解、100点満点、エキスパート獲得
16回目	２時間38分、15問全問正解、100点、エキスパート獲得
17回目	１時間45分、15問全問正解、100点、エキスパート獲得
18回目	３時間25分、**14問正解、94点**、エキスパート獲得
19回目	２時間46分、15問全問正解、100点満点、エキスパート獲得

Ⅱ　人間プログラマの強みと生成AIとの比較分析

（１）人間プログラマが持つ本質的な強みとは何か

生成AIの能力は近年目覚ましい発展を遂げており、コーディング支援や自動生成の分野で実用的な成果をあげている。しかしその一方で、人間のプログラマには依然として、AIには容易に置き換えられない本質的かつ核となる強みが存在している。以下では、人間がAIに対して優位性を保ち続けるいくつかの重要な能力について、その

根拠と共に検討していこう。

① 抽象的思考と創造性

人間のプログラマが持つ最も際立った特徴の一つが、「抽象的な思考」と「創造的な発想」に基づく問題解決能力である。現実のソフトウェア開発では、しばしば明確な仕様が存在せず、曖昧な要件を読み解き、それを構造化することが求められる。こうした状況においては、過去の経験や直感、感情に裏打ちされた判断が大きな役割を果たす。

AIは大量のデータからパターンを抽出し、それを新しい状況に応用する能力には優れているが、それは既存の知識の延長線上にとどまる。たとえば、全く新しいアルゴリズムをゼロから発明したり、斬新なアイデアを概念化したりする能力は、今のところ人間にしかできない芸当だと言われている。創造性とは、既存の知識の組み合わせだけでなく、それを飛び越えて新たな視点を見出す能力であり、それには想像力や独創性が不可欠であろう。

② 文脈理解と非定型問題への対応力

もう一つの大きな人間の強みは、複雑で文脈依存の問題に柔軟に対応できる点である。現実のプログラミング業務では、クライアントの曖昧な要望を読み取り、組織内の状況や業界特有の規範など、さまざまな文脈を踏まえて設計・実装を行うことが求められる。これに対して、AIは与えられた情報の範囲内では高精度な出力を行えるが、文脈の奥行きや人間関係、文化的・歴史的背景を深く理解し、それに応じた適切な対応を取ることはまだ困難であろう。

たとえば、ビジネス要件と技術的制約のバランスを取るような高度なトレードオフ判断、あるいは顧客の真の意図を推察しながらの要件定義といった工程は、人間の経験と共感能力に依存している。また、実世界のプログラミングでは「正解が一つとは限らない」ため、柔軟な判断力と創造的な問題解釈が不可欠である。

③ 倫理的判断と社会的責任

AIがいくら性能を向上させたとしても、「倫理的判断」を伴うタスクにおいては人間が欠かせない。ソフトウェア開発においては、セキュリティ、プライバシー、著作権といった倫理的・法的側面を意識した設計が求められる。AIは与えられたルールに従ってコードを生成することはできるが、たとえば「このコードは誰かを不利益にするのではないか？」といった社会的配慮までは自律的には行えない。

実際に、AIが生成したコードに脆弱性や他者の知的財産の不適切な利用が含まれることも少なくない。そうしたリスクを評価し、最終的に責任を持って対処できるのは、倫理的判断力を持った人間だけである。開発されたシステムが社会に与える影響を考慮し、その設計方針を調整できる存在として、プログラマは単なる技術者を超えた役割を担っているといえるであろう。

④ 協調性と対人コミュニケーション能力

ソフトウェア開発はチームで進めるプロジェクトである以上、他者と協力しながら作業を進める力も欠かせない。メンバー間の意思疎通、異なる立場の人々との合意形成、そして顧客との丁寧なコミュニケーションなど、人間関係の構築と維持はプロジェクトの成否を大きく左右する。

AIはドキュメントや会話の表面的なやり取りを模倣することはできても、相手の気持ちを汲んだ言葉の選び方や、誤解を避けるための丁寧な配慮といった、人間的なやり取りの本質にはまだ到達していない。特に、曖昧な要件を技術仕様に落とし込み、実装可能な形で再構成するような能力は、人間の知識と経験の結晶といえる。

AIは、大量の学習データに基づいて最適化された出力を高速に生成する能力に優れている。しかし、これはあくまで「既知の枠組みの中での最適化」に過ぎず、未知の状況や曖昧な前提のもとで本質的な理解を伴った問題解決を行う能力には限界があろう。

対照的に、人間のプログラマは、未定義の前提を補完したり、相手の意図を汲み取ったりといった「背景理解」や「目的意識」に基づいて行動することを得意とする。さらに、生成されたコードを批判的に評価し、品質や安全性、倫理面まで含めて責任を持つ判断を下せる点でも、人間の役割は依然として重要である。

このように考えると、プログラミングの未来はAIによる全面的な代替ではなく、人間とAIがそれぞれの強みを活かして協調する「協働的な創造

活動」に向かっているといえるかもしれない。人間のプログラマは、AIの出力を盲目的に受け入れるのではなく、それを批判的に吟味し、目的に照らして活用する能力を発揮することで、その価値を高めていくことが期待される。これは、プログラマが単なる「コードを書く人」から、「社会的・倫理的責任を担う知的専門職」へと進化していく過程でもある。

（2）囲碁AIとの比較（競技プログラミングの特異性）

囲碁AI「AlphaGo Zero」は、囲碁のルールのみを与えられた状態から、自らとの対局を通じて独自に学習を重ね、従来の人間の知見を大きく超える新たな戦略を見出すことに成功した。このAIの特徴的な点は、人間の対局データや定石などに依存せず、ゼロから試行錯誤を繰り返すことで、自律的な成長を遂げた点である。AlphaGo Zeroが編み出した着手は、当初は人間には理解し難いものであったが、後にその有効性が明らかになり、プロ棋士の戦略にも影響を与えた。これは、AIが人間の先入観に縛られず、新たな知の地平を切り拓く可能性を持つことを示す好例である。

囲碁AIは、「探索した世界」がそのまま「知の全体」となるような、閉じたルール体系の中での最適化を追求している。この点において、AIの学習可能性と限界が明確に分かれている。つまり、探索が及ばない範囲に関しては、本質的に未知のままであり、その先を想像することは難しい。

一方で、競技プログラミングもまた、明確なルールと評価基準の下で論理力と効率性を競う形式で行われるという点において、囲碁と類似の性質を有している。そのため、AIにとって非常に適したテストベッドであると考えられる。実際に、入力から出力までの関係が厳密に定義され、正答が自動的に判定される環境は、AIの学習アルゴリズムや推論能力を定量的に評価するのに理想的である。

しかし、こうした「明確に定義された問題空間」は、逆に言えばAIにとって「比較的扱いやすい問題」であるという見方もできる。特に強化学習や探索ベースの手法を用いるAIにとって、正誤がはっきりしている環境は試行錯誤の成果を即座に反映できるため、学習効率が高くなる。このため、競技プログラミングの分野ではAIが急速に性能を高めているのである。

実際、OpenAIの最新モデル（たとえばGPT-o3など）は、強化学習を通じて複雑なコード生成や問題解決において高い成果を示している。これらのAIは、人間が設計したヒューリスティックやテンプレートに頼らず、段階的な推論を繰り返しながら、自律的に解法を洗練させていく能力を備えている。この点において、囲碁AIと競技プログラミングAIは共通して、「人間の知識の延長線を超える可能性」を持つ存在であるといえるかもしれない。

ただし、両者の本質的な違いとして、囲碁が完結したルールベースの世界であるのに対し、競技プログラミングには「問題文の読解」「適切なアルゴリズムの発想」「コードとしての実装」「実行結果の検証と修正」といった、複数の段階を内包している点があげられる。特に、「問題の意図を読み取る力」や「新しいアルゴリズムを構想する創造性」は、現段階では人間の強みとして残されており、AIにとっては容易ではない課題となっている。

加えて、競技プログラミングのような閉じた問題空間と異なり、現実世界のプログラミングでは、要件の曖昧さ、ユーザーの多様性、実行環境の変化、そして倫理的配慮といった多様な要素が絡み合う。こうした「開かれた世界」では、AIが完全に人間を置き換えることは困難であると広く認識されている。

以上の点を踏まえると、AIが論理的に閉じたタスクにおいて人間を凌駕する可能性を見せている一方で、文脈理解や創造性、価値判断といった人間特有の能力が依然として重要であることが浮かび上がってくる。競技プログラミングにおけるAIの進歩は、人間の思考プロセスの一部を模倣・補完する力を持つことを示唆しているが、それが人間の役割の消失を意味するわけではない。

むしろ、AIと人間がそれぞれの強みを活かしながら協働することで、従来にはなかった創造的な解決策が生まれる可能性が開かれている。こうした観点から、囲碁AIや競技プログラミングAIの進展は、私たちがAIと共にどのように未来を

築いていくのかという問いを改めて投げかけていると言えるかもしれない。

Ⅲ　競技プログラミングにおける生成AIの倫理的課題と懸念事項

近年、生成AIの急速な進歩により、競技プログラミングの分野でもその活用が注目されつつある。しかしながら、AIの利用にはいくつかの倫理的な課題や懸念が伴う。これらを適切に理解し、対応することは、健全な競技環境を維持するうえで重要であろう。

・公平性と不正行為のリスク

AIを使って競技プログラミングの問題を解決する行為は、人間のみが参加することを前提とした大会においては、公平性を著しく損なうことになる。特に、AIが人間の代わりに問題を解いて提出するような行為は、参加者間の競技条件を大きく歪め、不正と見なされることにつながる。実際、AtCoderなどの主要なオンラインジャッジシステムでは、AIの利用に関するルールの見直しや厳格化が進められており、この問題への社会的関心の高さがうかがえる。

・人間のスキル習得への影響

生成AIに頼りすぎることは、人間の本来の学習プロセスに悪影響を与えることになる。競技プログラミングは、アルゴリズム的思考力やコーディングスキル、問題解決能力を鍛えることに主眼が置かれているが、AIがその役割を肩代わりしてしまうと、学習者が本質的な理解や応用力を身につけにくくなってしまう。AIは「答え」を提供する一方で、「考える力」を育てる機会を奪うことにもなりかねない。

・AIモデルに内在するバイアス

生成AIは大量の既存データをもとに学習されているが、そのデータに偏り（バイアス）が含まれている場合、生成されるコードにもその偏りが反映されてしまう。たとえば、特定のアルゴリズム設計やコーディングスタイルばかりを提案し、他の選択肢を無視することがあり、これが学習者の視野を狭めたり、特定の問題タイプで不公平な結果を招いたりする要因になる。こうしたバイアスの存在は、AIの透明性や信頼性を問う重要な論点にもなっている。

・透明性と説明責任の問題

現在の多くのAIモデルは「ブラックボックス」として機能しており、どのような論理や根拠に基づいて出力が得られたのかが分かりにくいという特徴がある。この透明性の欠如は、エラーが発生した場合に原因を特定しにくくし、さらにはその責任が誰にあるのかを曖昧にする。競技環境においては、提出されたコードの正当性や信頼性を確認するが、AIが生成したコードに対してそのプロセスが不明瞭であることは、大きな課題である。

・知的財産権の曖昧さ

AIが生成したコードの著作権や所有権は、現在の法律体系では十分に整理されていない分野のひとつである。たとえば、GitHub Copilotのようなサービスを通じて生成されたコードが、過去に公開されたオープンソースコードに類似している場合、それが著作権の侵害に該当するか否かは法的にも曖昧である。また、AI自身は法律上の著作者と見なされないため、そのコードの使用や再配布において誰がどのような権利を持つのかが不明瞭である。競技プログラミングにおいても、生成されたコードの再利用や商用利用の場面では、こうした知的財産の問題が避けて通れない課題となっている。

・セキュリティ上の懸念

AIが生成するコードは、必ずしも安全性が保証されているとは限らない。モデルが安全なコーディング手法や脆弱性への対策を十分に学習していない場合、結果としてセキュリティホールを含んだコードが出力されるリスクにつながる。特に、セキュリティに関する高度な知識が求められる競技や実務的なプログラムでは、AIによる生成物を無批判に利用することは危険を伴う。

以上のように、競技プログラミングにおけるAIの利用には多面的な倫理的課題が存在する。AIの能力がますます向上するなかで、「正当な支援」と「不正な優位性」の境界線は一層曖昧になりつつある。したがって、競技の精神や教育的価値を損なわないようにするためには、明確かつ実効性のあるガイドラインの整備が求められる。

特に知的財産権に関しては、現行法が想定していないケースが増えており、AIによるコード生成の現実に対応できるよう、法的整備や指針の見

直しが必要であろう。今後、AIが競技と実務の両面で広く利用されるようになるにつれて、所有権、著作権、ライセンスに関する問題を早期に明確化し、関係者間の認識を共有することが不可欠であろう。

おわりに

本文では、生成AIが競技プログラミングにおいて目覚ましい成功を収めている現状、その背景にある理由と根拠、人間のプログラマと比較した際の強みと弱み、生成AIの倫理的課題について多角的に分析してきた。

AlphaCode、AlphaCodium、GPT o1、o3といったAIモデルが競技プログラミングの分野で達成した顕著な成果は、AI技術の急速な進化を明確に示している。これらのモデルは、問題解決能力、コーディング効率、そして時には人間のプログラマを凌駕するパフォーマンスを発揮しており、競技プログラミングの風景を大きく変えつつある。

AI技術は今後も進化を続け、推論能力、創造性、複雑な問題への対応力などを向上させることで、競技プログラミングにおけるその可能性をさらに広げるであろう。AIと人間のプログラマの協調、そしてAIが新たな競技プログラミングの形式を生み出す可能性も視野に入れる必要がある。

しかしながら、現在の生成AIには、真に独創的な問題への対処、コードの信頼性や効率性、文脈理解の限界、そして計算リソースの必要性といった課題も存在する。これらの課題を克服し、AIの潜在能力を最大限に引き出すためには、今後の研究開発が不可欠である。

生成AIの競技プログラミングへの応用は、教育、研究、実務など様々な側面に影響を与える。プログラミング教育の個別化、AIによる問題解決能力の評価、現実世界のソフトウェア開発における生産性向上など、その恩恵は多岐にわたる。一方で、公平性、人間のスキル開発への影響、知的財産、セキュリティといった倫理的な課題にも真摯に向き合い、適切なガイドラインを策定していく必要がある。

競技プログラミングにおける生成AIの進化は、単にゲームのルールを変えるだけでなく、人間の知性とAIの協調の可能性を探るための重要な機会を提供する。今後、AIと人間がそれぞれの強みを活かし、どのように協力してより高度な問題解決に取り組んでいくのか、その動向を注視していく必要がある。

参考文献

A. El-Kishky, A. Wei, A. Saraiva et al. (2025) 'Competitive Programming with Large Reason-ing Models' arXiv preprint arXiv:2502.06807.

Antti Laaksonen (2020) “Guide to Competitive Programming: Learning and Improving Algorithms Through Contests” Springer.

Michael R. Garey, David S. Johnson (1979) “Computers and Intractability: A Guide to the Theory of Np-Completeness” W H Freeman & Co.

S. Lertbanjongngam, B. Chinthanet, T. Ishio, RG. Kula, P. Leelaprute and B. Manaskasemsak (2022) 'An empirical evaluation of competitive programming ai: A case study of alphacode', 2022 IEEE 16th International Workshop on Software Clones (IWSC), 10-15.

秋葉拓哉，岩田陽一，北川宜稔（2012）『プログラミングコンテストチャレンジブック［第2版］』マイナビ出版。

大槻兼資，AtCoder株式会社（2022）『アルゴリズム的思考力が身につく！ プログラミングコンテストAtCoder入門』KADOKAWA。

大槻兼資，杉江祐哉，中村謙弘（2023）『アルゴリズム実技検定　公式テキスト［上級］〜［エキスパート］編』マイナビ出版。

米田昌悟（2022）『競技プログラミングの鉄則〜アルゴリズム力と思考力を高める77の技術〜』マイナビ出版。

渡部有隆，Ozy，秋葉拓哉（2015）『プログラミングコンテスト攻略のためのアルゴリズムとデータ構造』マイナビ出版。

The Rise and Issues of Generative AI in Competitive Programming

NI Yongmao

Abstract

In recent years, the field of artificial intelligence (AI), especially generative AI powered by large language models (LLMs), has achieved remarkable progress, demonstrating capabilities across a wide range of domains. A notable application is automatic code generation. Remarkably, generative AI has also begun to achieve strong results in competitive programming, even reaching levels comparable to human participants.

This paper analyzes, from multiple perspectives, how generative AI has emerged in competitive programming, the mechanisms underpinning its performance, and the reasons behind its impressive results. As part of this analysis, we designed and conducted two experiments where generative AI participated in five consecutive mock sessions of the AtCoder Beginner Contest (ABC) and the AtCoder Programming Skill Test (PAST), with its performance tracked and recorded.

Furthermore, by comparing the abilities of generative AI with those of human programmers, we clarify their respective strengths and weaknesses. Finally, we explore the future outlook and limitations of generative AI in light of technological advancements and expanding applications, while also addressing associated ethical issues and concerns.

（2025年 6 月 1 日受理）