解説

応般

プログラミングコンテスト奮戦記

―アルゴリズム・パズルの面白さと奥深さ―

秋葉 拓哉　東京大学 情報理工学系研究科

本稿のきっかけと目的

プログラミングコンテストが世界一強い大学がどこだか，ご存知でしょうか？　1つの尺度における現在の正解は，なんと東京大学です．数年前にはトップ10にも入っていなかった東京大学は，現在，世界ランキング1位に輝いています（**表-1**）．このように，近年，日本はプログラミングコンテストの世界において大躍進を遂げています．その一方で，他の強豪国に比べ，日本でのプログラミングコンテストへの関心は，いまだにかなり低いままと言えます．

今回，筆者を含むチームが世界最大のプログラミングコンテストであるACM-ICPC（International Collegiate Programming Contest）で銅メダルを獲得し，このような記事を書かせていただく貴重な機会をいただきました．この記事をきっかけに，プログラミングコンテストに関心を持っていただければ嬉しいです．学生の方には，できればコンテストに参加してもらいたいと思います．また，先生方が，コンテストで成果を出す学生に高い評価を与えてくださったり，学生のコンテスト出場を促していただけると，学生がチャレンジしやすくなると思います．

	大学	国	レーティング
1	東京大学	日本	2625.89
2	精華大学	中国	2493.79
3	サンクトペテルブルク大学	ロシア	2452.14
4	ワルシャワ大学	ポーランド	2449.68
5	サンクトペテルブルク情報技術・機械・光学大学	ロシア	2385.11

表-1　TopCoder アルゴリズム部門 大学ランキング☆1

プログラミングコンテストとは

プログラミングコンテストでは，開始とともに，問題が何問か提示され，そこで指示されたプログラムを作成します．たとえば，以下は非常に簡単な問題の例です．

【問題 A+B】
標準入力に，100以下の2つの正整数が与えられます．その和を標準出力に出力してください．

もちろん，本当のコンテストでは，より複雑な問題が出題されます（**図-1**）．問題数はコンテストによりますが，短時間（75分～2時間程度）のコンテストなら3～5問程度，長時間（5時間）のコンテストなら10問程度が標準的です．

コンテストが開始したら，参加者は，問題を読み，その指示に従ったプログラムを作成し，ソースコードを提出します（**図-2**）．すると，正答・誤答の判定が行われます．時間中に何問解けるかや，どれだ

☆1 2012年8月19日現在のもの，http://community.topcoder.com/stat?c=school_avg_rating

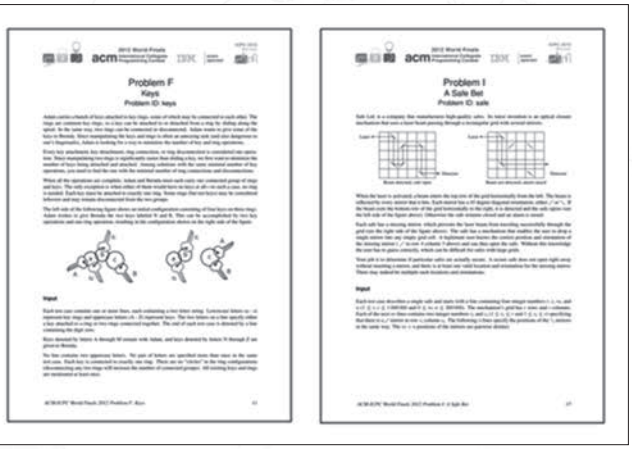
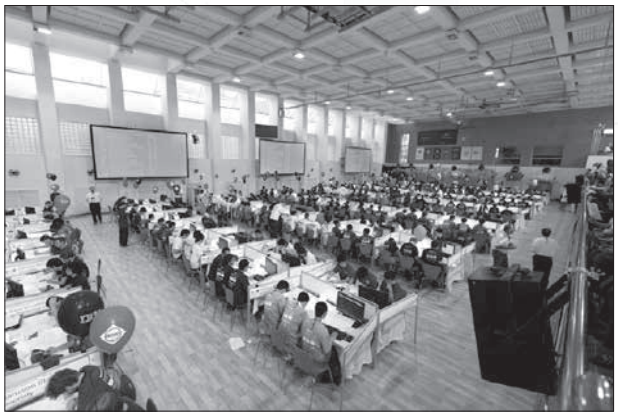


図 -1　実際の問題の例

```
#include <iostream>
int main() {
  int a, b;
  std::cin >> a >> b;
  std::cout << a + b << std::endl;
  return 0;
}
```

図 -2　問題 A+B の解答例



ICPC Photo より http://icpc.baylor.edu

図 -3　ACM-ICPC のコンテスト中の風景

け早く正解したか等で順位がつきます．

プログラミングコンテストの大きな特徴として，採点が自動で行われるという点があります．「こういった入力に対し，こういった出力をせよ」という問題であるため，採点データとして，入力データとそれに対する正しい出力データをあらかじめ用意しておくことができます．ソースコードの提出が行われたら，システムが自動でコンパイルし，それらの入力に対し実行し，出力をチェックします．

世の中には，大きく分けて 3 種類のプログラミングコンテストがあり，ここで説明した問題を解くコンテストはそのうちの 1 種類となります．ほかには，数日間〜数カ月間の間で 1 つの問題に対してどれだけ性能の高いプログラムを作れるかという種類のコンテストや，お題に沿ってどれだけ面白いソフトウェアを作れるかという種類のコンテストがありますが，今回は触れません．

この種の世界的なコンテストには，最も歴史があり有名な ACM-ICPC（**図 -3**），最も盛んに開催される TopCoder，トップ企業が開催する Google Code Jam，Facebook Hacker Cup 等があります．

プログラミングコンテストで問われるもの

名前の通り，プログラミングコンテストはプログラミングのコンテストなのだから，ソースコードをどれだけ早く記述するかを競うだけなのではないかと思われるかもしれません．確かに，それも一部ですが，プログラミングコンテストはもっと遥かに奥の深い競技です．

▶ アルゴリズムを設計する力

まず，最初に問われるのが，アルゴリズムを設計する力です．コンテストの問題は，「○○を計算せよ」などという形式であり，自分で計算する方法を考えなければなりません．プログラミングコンテストをよく知らない人には意外な事実だと思いますが，実はこの，アルゴリズムを設計する部分こそが，プログラミングコンテストにおいて最も差がつく部分になります．問題を解くためには，アルゴリズムを思いつけなければ，実装にとりかかることができず，手も足も出ない状態になってしまいます．また，実装の難易度はそこまで大きな差をつけにくい一方，アルゴリズムの難易度は幅が非常に大きいため，コンテストにおける実力はアルゴリズム設計力が占め

る部分がかなり大きいと言えます．

設計するアルゴリズムは，正しく計算できればどのような方法でもよいわけではありません．プログラムの実行時間には制限があるため，指定されたサイズの入力を実行時間制限の中で処理できる速度のアルゴリズムを考える必要があります．たとえば，計算量が $O(2^n)$ の全探索のアルゴリズムならばすぐ思いつくとしても，n が 1,000 になる入力もあると指定されている場合，そのようなアルゴリズムでは到底実行時間制限に間に合わないため，動的計画法等に基づくより効率的なアルゴリズムを考える必要があります．

正解を得るためには，まず十分に高速なアルゴリズムを考える必要があります．しかし，良い成績を納めるためには，それでもまだ十分ではありません．アルゴリズムを考える時点で，できるだけコードがシンプルで短くなるようなものを考えられると，より良い成績を得られるようになります．たとえば，問題の制約が緩ければ，無理に最良のアルゴリズムを選択せず，計算量や精度を悪くしてでもシンプルなアルゴリズムにすることが有利になったりします．

• アルゴリズムを設計する力の分析

アルゴリズムを設計するために必要なものは大きく分けて 2 つです．1 つは，アルゴリズムの知識，そしてもう 1 つはそれを活用する柔軟な思考力です．ただし，これらは独立ではありません．

まず，さまざまなアルゴリズムの知識がしばしば要求されます．たとえば，グラフにおいて最短路や最小全域木を求めるアルゴリズム，集合や数列を管理するためのデータ構造，線形連立方程式を解くアルゴリズムなどです．知識さえあればすぐ解けてしまう問題は面白くなく出題もされにくいため，基礎的で応用の効くアルゴリズムの知識が重要になってきます．これらは，問題を解くためのアルゴリズムのパーツとして活かされていきます．

また，各アルゴリズムの知識以上に，アルゴリズム設計技法についてよく知っておく必要があります．アルゴリズム設計技法とは，貪欲法，動的計画法，分割統治法など，アルゴリズムを設計する際のアプローチの仕方です．プログラミングコンテストでは，自分でアルゴリズムを設計する必要があるので，しばしばそれらの技法に沿ったアルゴリズムを設計することになります．技法の考え方を知っておくだけでなく，実例を多く知っておくと，それらからの類推でアルゴリズムが設計しやすくなります．

次に必要なものが，それらの知識を活用して実際にアルゴリズムを設計する思考力です．知識さえあれば解けるような問題や，アルゴリズムがすぐに分かってしまうような問題は，非常に簡単な問題としてのみ出題されます．難しい問題は，その問題特有の性質をじっくりと考察してゆき，部分問題に分けたりしながら，いろいろなアイディアを試行錯誤し，設計技法やアルゴリズムを組み合わせて 1 つのアルゴリズムを設計できるようになっています．頭の柔らかさのようなパズル的な能力があると有利ですが，どれだけアルゴリズムを設計してきたかというような経験も重要です．

• アルゴリズムを設計する力を得るには

以上のような能力を身に付けるにはどうすればよいでしょうか．残念ながら，大学で行われるアルゴリズムの授業や，「アルゴリズム C」「アルゴリズムイントロダクション」のような多くの有名な教科書は不十分と言わざるを得ません．現状，それらはしばしばアルゴリズムを紹介することに徹しており，アルゴリズムを設計することや，活用することについては手薄になってしまっています．

発想力を身に付けたり経験を積んだりするためには，実際にアルゴリズムを考える機会を持つことも不可欠です．一方，知識やアプローチを把握するためには，ある程度勉強をする必要があります．アルゴリズム設計に焦点を当てた本としては，比較的新しい「アルゴリズムデザイン」という本が存在します．しかし，それでもアルゴリズムの活用法や組合せ方，少し進んだアルゴリズムについて等のまとまった資料は不足していました．そのため，たとえば筆者らは他人のソースコードを解読し解法やテクニックを学んだり，中国語の Web ページを機械翻訳により読んだり，より専門的な本を読み解いたりし

ながら情報交換を盛んに行っていました．

そのような状況を反映し，筆者を含むプログラミングコンテスト経験者 3 人による書籍「プログラミングコンテストチャレンジブック」が 2010 年に発売されました（**図 -4**）．アルゴリズム自身だけでなくアルゴリズム設計法・活用法・実装法のすべてに重点を置いた初めての書籍として，プログラミングコンテストに参加しない層からも支持を受けており，2012 年には第 2 版が出版され，合計して 1 万部以上を売り上げたほか，韓国語・中国語に翻訳もされています．

図 -4　プログラミングコンテストチャレンジブック（第 2 版）および翻訳版

▶ プログラムを素早く正しく実装する力

アルゴリズムを考えついたら，それをプログラムにしなければなりません．アルゴリズムが思いつかなければ問題に手も足も出ない一方，実装はただ実装するだけじゃないかと思われるかもしれません．しかし，実際には実装の時間もしばしば何倍も差がつきます．どのような点で差がつくのでしょうか．

まず，どれだけスラスラと実装できるかという点があります．1 行書くごとに悩んだり，削除や前に戻っての修正を何度もやったりしていると，短いソースコードでも思った以上に時間がかかってしまいます．次に，どれだけコンパクトに実装できるかという点があります．抽象化を上手く行ったり，言語の標準ライブラリを活用したりして，できるだけ楽に実装するようにします．最後に，どれだけ正確に実装できるかという点があります．バグの修正は非常に時間がかかりますし，誤答のペナルティを受けるのも望ましくありません．間違いやすい部分には気をつけてゆっくり実装したり，間違いにくいように無理な抽象化を避けたり，見やすくコーディングすることを心がけたりします．

• 実装力を身に付けるには?

実装の力を付けるには，基本的には実際にコーディングすることがまず重要です．コードをスラスラと書いたり間違えずに書いたりする能力は，基本的にはコーディングをすることにより得られると思います．

一方，ライブラリ関数を活用したりデータの持ち方を工夫したりしてコードをコンパクトにしたりすることは，1 人での練習だけでは限界があります．そこで，プログラミング言語についてしっかり知ることと，優れた人が書くコードを読むことが必要になります．一部のコンテストでは，他人のソースコードが終了後読めるようになるため，上位者のソースコードを読みさまざまな発見をすることができます．また，先述の「プログラミングコンテストチャレンジブック」でも，すべての問題に対してソースコードをつけてあり，実際にアルゴリズムを実装するところにまで重点を置いています．

▶ 戦略的に問題に取り組む力

これは，実力があってこそではあるものの，しばしば明暗を分ける力です．特に，次に進出すればよいようなラウンドでなく，重要な勝負で優勝や入賞を狙っていくときには，「運が悪かった」で済まさないためにも，戦略的に行動する必要が出てきます．

戦略において最も重要なのは，どの問題に取り組むかということです．たとえば，順位表を見ると，簡単な問題を把握することができるだけでなく，他チームの誤答具合などから落とし穴の有無等を推測することもできます．また，勝利に必要なスコアとの兼ね合いで，安全を取るかリスクを取るかといった判断が必要になることもあります．このようなとき，自分が実装・デバッグにかかる時間をなるべく正確に見積もれる能力が必要になってきます．

図 -5　Aizu Online Judge の画面[☆2]

プログラミングコンテストで活躍するために

▶実践的な練習

プログラミングコンテストの問題を解いて経験を積むことは，実力の核になりますが，プログラミングコンテストに出場する以外でも，日常的に問題を解くことができるようになっています．

その仕組みは，「オンラインジャッジ」と呼ばれる Web サイトです．過去のコンテストの問題が大量に収録されており，問題の検索・閲覧ができるだけでなく，先述したプログラムの自動正誤判定が 24 時間行えるようになっています．そのため，いつでもどこでもインターネットさえあれば問題を解くことができるようになっています．ほかに何人その問題を解いているか等も分かるため，自分に適した難易度の問題を探すこともできるほか，解いた問題数等でランキングがついたりライバルと比較したりできるようになっています．問題を解くこと自体が楽しいこともあり，しばしばオンラインジャッジで解いた問題数で競っている人たちがいます．

オンラインジャッジシステムはいろいろな特徴を持つものが世界中に多数あります．日本のオンラインジャッジシステムとしては，会津大学の Aizu Online Judge（AOJ）（**図 -5**）[☆2] が有名で，多くの日本人が日々問題を解いています．日本語の問題が数多く収録されていることが特徴で，入門者のための非常に簡単な問題のコースも用意されています．

[☆2] http://judge.u-aizu.ac.jp/onlinejudge/

▶トップ競技者はどのように育つか

実際のコンテストにおける参加者の実力の差はかなり大きく，トップ層の参加者は安定して常に上位を取り続けます．そのような世界的なトップ層はどのようにしてその実力を身に付けるのでしょうか．

基本的には，まず，プログラミングコンテストに大きな時間をかけることが一番の近道です．オンラインジャッジ等で解いた問題の数が 1 つの目安になりますが，最低でも数千問は問題を解いています．筆者の場合は恐らく 2,000 ～ 3,000 問程度で，TopCoder で現在最も高いレーティングを持つ Gennady Korotkevich は 10,000 問を超えると言っていました．たとえば筆者は比較的余裕のある大学 1，2 年の頃にほとんどの暇をオンラインジャッジに費やしていました．ほかのトッププレイヤも似たような生活を経験しています．その程度に生活を捧げる覚悟は必要だと思われます．

また，闇雲に問題を解けばよいというものではありません．自分にとってちょうどよい難易度かつ質の良い問題を解いたり，解いた問題について最大限の知見を得たりすることが実力向上に繋がります．そこで重要になってくると思われるのが，仲間の存在です．たとえば，1 人で問題を解くよりも，他人と共通した問題を解くと，その後で解き方や実装方法について議論することができ，得られる知見が大きく向上します．教えてくれる先輩的存在でもよいですが，筆者の意見としては，似たような実力のライバルのような存在が重要だと思います．その理由は，似たような実力のため，自明だと思ったり非自明だと思ったりする部分が共通しており，お互いのレベルにあった議論になり，得られるものが大きくなるからです．

なお，日本でも外国でも，数学オリンピック経験者がプログラミングコンテストに転向し好成績を納める例が多いです．彼らが数学を通じて得たパズル的な思考力はプログラミングコンテストでも活きるのだということが分かります．一方，プログラミングコンテストで優秀な成績を納める人間は必ずしもそのような数学で秀でた成績を納める人間というこ

とはありません．

▶ 日本はなぜ一気に強豪国となったか

筆者がプログラミングコンテストに本格的に取り組み始めた 5 年前，TopCoder の国別ランキングにおいてトップ 10 にも入っていなかった日本は，現在国別ランキング 4 位に位置しています．特に，冒頭で述べた通り，大学別では東京大学が世界 1 位になっています．日本に何が起こったのでしょう．

一番大きいのは，コミュニティが形成・拡大し，盛り上がっていることだと思います．個人の努力や書籍・Web ページのみでも実力を身に付けることができるとはいえ，身近にコンテストに取り組む人間がいると状況は随分と変わってきます．特に，トップ層はかなりの時間をコンテストに捧げるようなかなり割り切った生活へ勇気を持って踏み出さなければなりませんが，同じような生活をする仲間がいることや，同じような生活を経験し成功している先輩がいることは大変心強いと言えます．あるいは，実力の拮抗するライバルがいたり，到達したい・超えたい先輩が身近にいたりすることも大きな刺激となります．特に興味を持ったばかりのころは，勉強すべきものやトレーニング方法，ほかのコンテストの情報などがまったく分からず，そういった情報は自分から調べようとするのは難しいため，コミュニティがしっかりしていると圧倒的に有利になります．近年の日本では Twitter がかなり流行しており，Twitter 上で四六時中プログラミングコンテストに関する議論が行われています．

特に大きな影響を与えているのが，情報オリンピックの存在だと思います．情報オリンピックは，中高生向けのプログラミングコンテストで，10 年の休止を経て 2005 年に再開しています．中高生向けながら非常に難易度の高い問題が出題されることが大きな特徴です．再開直後はあまり知名度も高くなく，トレーニング方法等も知られておらず，競争も激しくなかったため，筆者を含むほとんどの学生はぶっつけ本番のようなスタイルで挑んでいました．一方現在は，コンピュータ好きの中高生における知名度は高く，オンラインジャッジ等のトレーニング方法なども知れ渡るようになり，かなりエスカレートした勝負が行われています．中高生は自由な時間を多く持っていることもあり，日本代表に選ばれるレベルの中高生は大学生顔負けの実力を持っていることも少なくなく，そのまま世界トップレベルの競技者になることがあります．

プログラミングコンテストに関する議論

▶ プログラミングコンテストの面白さ

プログラミングコンテストはなぜこれだけの人を惹きつけここまでエスカレートしているのでしょうか．

まず 1 つには，それ自体でも魅力のあるプログラミングがゲームのようになっているということがあると思います．制限時間の中で集中してプログラムを書くという行為はとても興奮します．あるいは得意なプログラミングで他人と勝負することが楽しいと思うかもしれません．

次に，手軽ながら大規模であるということがあると思います．特にインターネット上のコンテストでは，パソコンさえあればどこでも参加ができる手軽さの一方で，世界中のライバルの数は時には 1 万人にも登ります．また，自動採点システム等の影響でコンテスト開催のコストが非常に低いため，コンテストが頻繁に開催されています．

しかし一番大きいのは，やはりアルゴリズムを考えることがパズルのようで奥深く面白いという点があると思います．数学オリンピック経験者がわざわざプログラミングを勉強してプログラミングコンテストに進出してくるのも，こういった魅力が原因だと思います．

▶ プログラミングコンテストの意義

プログラミングコンテストは楽しいだけでなくそれを通じて何かを得られるのでしょうか．

まず，言わずもがな，アルゴリズムとプログラム

実装の能力が付きます．アルゴリズムに関しては，授業や教科書で身に付く有名アルゴリズムの知識だけでなく，アルゴリズムを設計する能力というほかで得られにくい能力が得られます．また，実装に関しては，特に複雑で間違えやすいロジックを冷静に正確に実装する能力がぐんと伸びます．

それ以外では，自分をアピールする道具になるという点があると思います．特に，プログラミングが好きな大学生であっても，趣味のソフトウェア開発等に取り組んでいなければ，研究を始める前の学年において自分をアピールする道具は皆無です．また，信頼できる仲間が得られるということもあると思います．コンテストで普段から競り合っている相手の実力というのは手に取るように分かるものです．あるいは，プログラミングコンテストを通じて世界を知ったり世界を相手にする自信を得たりすることで良い刺激が受けられるということもあると思います．

▶ プログラミングコンテストに対する批判

一方で，世間ではプログラミングコンテストへの批判もあります．その一部を取り上げます．

まず，一番よく聞く批判としては，プログラミングをするならもっと有意義なものを作れ，勝負などしていないで皆でいろいろ違うプログラムを作るべきだ，というものがあります．しかし，少なくとも筆者個人としては，これには賛同しかねます．面白いアイディアを自分で考え新しいソフトウェアを作ることは重要な社会貢献ですが，それがすべてではありません．たとえば大学におけるコンピュータ科学の研究においても，そのようなソフトウェア系の研究はあくまで 1 分野であり，ほかのジャンルでは，既存の手法より優れた手法を設計するような研究が基本になります．そういった場所では，しばしばアルゴリズムを設計することとなるため，プログラミングコンテストはその練習として絶好の場だと思います．こういった練習を否定することは，数学を勉強するために練習問題を解くことを否定していることと同じだと思います．また，実際のソフトウェア開発でも，有用なアイディアを出すだけでなく，アルゴリズムを工夫して高速にすることが重要になる部分もあります．

一方，筆者自身もそうだと思う批判として，身に付く能力が偏っているということが挙げられます．たとえば，プログラミングコンテストでは大規模なプログラムを書かないという点があります．1 問に対して，大抵 100 行前後のソースコードを 1 ファイルに記述し，正解となればそれをほぼ捨てて次に進みます．そのため，複数ファイルに渡る大規模なプログラムを実装し管理するようなことは練習できません．また，使うアルゴリズムもプログラミングコンテストの形式で出題しやすいものばかりになっているきらいがあります．しかし，これらに関しては，1 つそれだけをやっているだけですべてが学べるような万能のトレーニングなど世の中に存在せず，プログラミングコンテストに限らず 1 つのことだけをやっているのでは不十分であるということに尽きるのだと思います．

参考文献

1）Sedgewick, R.（著），野下浩平，他（翻訳）：アルゴリズム C・新版―基礎・データ構造・整列・探索，近代科学社（2004）．
2）Cormen, T. H., Stein, C., Rivest R. L. and Leiserso, C. E.（著），浅野哲夫，他（翻訳）：アルゴリズムイントロダクション第 3 版第 1 巻 数学的基礎とデータ構造，近代科学社（2012）．
3）Cormen, T. H., Stein, C., Rivest R. L. and Leiserso, C. E.（著），浅野哲夫，他（翻訳）：アルゴリズムイントロダクション第 2 版第 2 巻 アルゴリズムの設計と解析手法, 近代科学社（2007）．
4）Cormen, T. H., Rivest R. L. and Leiserso, C. E.（著），浅野哲夫，他（翻訳）：アルゴリズムイントロダクション第 3 巻精選トピックス，近代科学社（1995）．
5）Kleinberg, J. and Tardos, E.（著），浅野孝夫，他（翻訳）：アルゴリズムデザイン，共立出版（2008）．
6）秋葉拓哉，岩田陽一，北川宜稔：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（2012）．

（2012 年 8 月 29 日受付）

秋葉 拓哉　t.akiba@is.s.u-tokyo.ac.jp

東京大学情報理工学系研究科修士 2 年．ACM-ICPC をはじめとする世界的なプログラミングコンテストにおいて 10 回以上の世界大会出場を経験．2010 年に「プログラミングコンテストチャレンジブック」を共著．大学では大規模グラフ向けのアルゴリズムを研究．