情報オリンピックの問題

基般

保坂和宏（東京大学理学部数学科）

JJOOII（JOI 2011／2012 本選 1）

■ 問題

問題文は次ページの**図 -1** を参照のこと．

時間制限 1 秒，メモリ制限 128MB

■ 解説

本問は，2012 年 2 月に行われた第 11 回日本情報オリンピック（JOI）本選の最初の問題として出題された．本選の第 1 問は例年比較的簡単であることが多いが，簡単な問題でもアルゴリズムの効率の良し悪しを区別する工夫がなされており，素朴な解法では満点は獲得できないのである．

問題文を要約すると**表 -1** のようになる．入力として文字列 S（長さ $1 \leq N \leq 10^6$）が与えられたとき，表のような「JOI 列」のうち，S に一部として含まれるもので最もレベルが高い（すなわち，長い）ものはどれか求めることが問われている．

単純な解法

単純な解法として，ほぼ問題の指示通りのアルゴリズムを設計するというものが考えられるだろう．すなわち，各 k $(0 \leq k \leq N/3)$ に対してレベル k の JOI 列を考え，それが S に含まれているか判定するのである．判定するには，S の中でレベル k の JOI 列が始まる位置の候補（$N-3k+1$ 個ある）のそれぞれについて，そこからの k 文字が J であるか，続く k 文字が O であるか，さらに続く k 文字が I であるか，を調べればよい．

この方法の時間計算量を考えてみよう．かかるステップ数（S の各文字が調べられる回数）は $\sum_{k=0}^{\lfloor N/3 \rfloor} (N-3k+1) \cdot 3k = O(N^3)$ となっている．これは，部分点が設定されている $N \leq 100$ に対しては，実行時間制限である 1 秒に余裕をもって間に合うが，$N=10^6$ のような大きいデータに対しては間に合わない．

高速化のためには「ある S の部分文字列が JOI 列になっているかどうか高速に判定する」「JOI 列を探す場所の候補を減らす」という 2 点での工夫が考えられる．

なお，情報オリンピックの標準的課題では，すべてのデータに対してプログラムが時間・メモリ制限内で動作し正解することが求められるので，我々が興味があるのはアルゴリズムの平均的な性能ではなく，最悪ケースの計算量であることに注意されたい．

やや効率の良い解法

ここでは，「ある S の部分文字列について，それが JOI 列になっているかを判定する」という部分の高速化を検討しよう．そのためには，「S の a 文字目から b 文字目まではすべて J であるか」という形の質問に高速に答えられればよい（O, I についても同様である）．

実は，「すべて J であるか」より強く「J は何個現れるか」に答えることができる．J の個数の先頭からの累計を事前に求めておけばよいのである．

S が文字列 OJJOOIIOJOI の場合の例を**表 -2** に示す．

S の 1 文字目から x 文字目までにある J の個数を J_x とおくと（$J_0=0$ とする），a 文字目から b 文字目までにある J の個数は $J_b - J_{a-1}$ として求められるのである．たとえば上の例では 3 文字目から 10 文字目までにある J の個数は $J_{10} - J_2 = 3 - 1 = 2$ となっている．

以上により，$J_0, J_1, \ldots, J_N$ は前計算で $O(N)$ 時間

第 11 回日本情報オリンピック 2011/2012 本選
2012 年 2 月 12 日（東京）

1 JJOOII（JJOOII）

JOI（日本情報オリンピック）の本選に向けてプログラミングの練習をしていたあなたは，今年度の JOI の予選の問題には数値を扱う問題ばかりが出題され，文字列を扱う問題がなかったことに気がついた．そこであなたは，こっそり文字列の問題に強くなってライバルたちに差をつけることにした．

JOI の過去問を眺めていると，J,O,I の 3 種類の文字からなる文字列に慣れておく必要がありそうなことが分かった．そこで，そのような文字列について考えよう．あなたは「与えられた文字列が JOI という部分文字列を持つかどうかを答えよ」という問題を思いついたものの，これはすぐに解けてしまった．もっとレベルの高い問題を解きたいあなたは，以下のような問題を作った．

文字列 t が文字列 s の**部分文字列**であるとは，t の先頭および末尾に何文字か（0 文字でもよい）を付け足すと s になることである．たとえば，JJOOII は OJJOOIIOJOI の部分文字列である．一方，JOI は JOOI の部分文字列ではない．

また，0 以上の整数 k に対し，**レベル k の JOI 列**とは，k 個の文字 J，k 個の文字 O，k 個の文字 I をこの順に並べた文字列のことであるとする．たとえば，JJOOII はレベル 2 の JOI 列である．

与えられた文字列の部分文字列である JOI 列のうち，レベルが最大のものを求めたい．

課題

J,O,I の 3 種類の文字からなる長さ N の文字列 S が与えられたとき，レベル k の JOI 列が S の部分文字列であるような最大の k の値を求めるプログラムを作成せよ．

制限

$1 \leqq N \leqq 1000000$ $(= 10^6)$ S の長さ

入力

標準入力から以下のデータを読み込め．

- 1 行目には J,O,I の 3 種類の文字からなる文字列 S が書かれている．

出力

標準出力に，レベル k の JOI 列が S の部分文字列であるような最大の k の値を表す整数を 1 行で出力せよ．

採点基準

採点用データのうち，配点の 20% 分については，$N \leqq 100$ を満たす．

入出力例

入力例 1	出力例 1
OJJOOIIOJOI	2

OJJOOIIOJOI はレベル 2 の JOI 列である JJOOII を部分文字列として含んでおり，レベル 3 以上の JOI 列は部分文字列として含まない．

入力例 2	出力例 2
IJJIIJJJ	0

レベル 0 の JOI 列は長さ 0 の文字列である．

入力例 3	出力例 3
JOIJOIJOIJOIJOI	1

入力例 4	出力例 4
OOJJJJJJJOOOOIIIII	4

図 -1 「JJOOII」問題文

で求めることができ，JOI 列かの判定を $O(1)$ 時間で行えるようになった．全体の時間計算量は $O(N^2)$ である（これでも $N = 10^6$ に対しては時間超過となる）．

効率の良い解法

S の部分文字列は $O(N^2)$ 個あるので，さらなる高速化のためには調べる候補をうまく減らす必要がある．JOI 列に現れる特徴的な部分を探すのがポイントである．

S のある部分に JOOOI という部分が現れたとしよう．この場所の O を含む JOI 列を考えると，JJOOII や JJJJOOOOIIII などは現れることができず，あり得るものはレベル 3 の JJJOOOIII のみ

レベル	JOI 列
0	（空文字列）
1	JOI
2	JJOOII
3	JJJOOOIII
4	JJJJOOOOIIII

表 -1 「JOI 列」の例

S	O	J	J	O	O	I	I	O	J	O	I
J の個数	0	1	2	2	2	2	2	2	3	3	3
O の個数	1	1	1	2	3	3	3	4	4	5	5
I の個数	0	0	0	0	0	1	2	2	2	2	3

表 -2 各文字の個数の累計

であることが分かる．一般に，Oがk個並んでいて左右がOでないとき，そのOを含むJOI列としてはレベルkのもののみがSに含まれ得る．

このことを用いると，以下のアルゴリズムが考えられる．

1. Sの中でOが連続している極大な範囲をすべて取り出す
2. それぞれについて，JOI列となり得る部分文字列の位置が定まるので，それが実際にJOI列になっているか調べる．今まで見つかったものよりレベルの値が大きければ，答えを更新する．

以上により，$O(N)$時間・$O(N)$メモリで本問を解くことができた．これで満点を獲得することができる．

なおこの解法では，部分文字列がJOI列かどうかを判定する際に愚直に調べても実は問題ない．なぜなら，調べることになる部分文字列の長さの合計は，Sに含まれるOの個数の高々3倍なので$O(N)$になっているからである．

別解

上記の解法と本質的に同じではあるが，別のすっきりした言い換えがある．それは，文字列のrun-length encoding（RLE）を用いる方法である．

文字列のrun-length encodingとは，同じ文字の連続をその文字と長さによって表す方法である．たとえば，文字列OOJJJJJJJOOOOIIIIIからは

O2　J7　O4　I5

という列が得られる．レベルkのJOI列は

Jk　Ok　Ik

と表される．

run-length encodingを施すと，JOI列が現れる部分は，

Ja　Ob　Ic

という形の部分に限られ，

- $a \geq b, b \leq c$のとき，レベルbのJOI列が現れる
- そうでないとき，この部分にJOI列は現れない

ということが分かる．よって，Sにrun-length encodingを施し（これは$O(N)$時間でできる），連続する3ブロックに対して上の条件を満たすものを確かめていけば，$O(N)$ですべてのJOI列を見つけることができるのである．

```
int i0, i1, i2, i3;
int answer = 0;
for (i0 = 0; i0 < N; i0 = i3) {
  for (i1 = i0; S[i1] == 'J'; ++i1) ;
  for (i2 = i1; S[i2] == 'O'; ++i2) ;
  for (i3 = i2; S[i3] == 'I'; ++i3) ;
  if (i1 - i0 >= i2 - i1 && i2 - i1 <= i3 - i2) {
    if (answer < i2 - i1) {
      answer = i2 - i1;
    }
  }
}
```

図-2 「JJOOII」の満点解法の実装例

この解法の発想を基にすると，

J0　O2　I0　J7　O4　I5

のように長さ0のブロックがあるとみなしてJ,O,Iがこの順に繰り返し現れていると考えることができる．実装は簡潔になり，入出力を除いた部分のコードはC++などでは**図-2**に示すようになる．

Friend（IOI 2014）

■問題

問題文は次ページの**図-3**を参照のこと．

時間制限1秒，メモリ制限16 MB

■解説

本問は，2014年7月に行われた国際情報オリンピック（IOI）台湾大会の第2日の課題の1つとして出題された．金メダルを獲得した選手も多くは苦戦した難問で，満点を得たのは参加311人中13人であった．

本問は，人を頂点，互いに友達であるという関係を辺とするグラフの問題として記述することができる．情報オリンピックの問題では，グラフ理論の用語が問題文で使われることはほぼないが，このように何らかの現実的なモデルで記述され，背後ではグラフ理論的思考やグラフアルゴリズムに関する知識が要求されることも多い．

さて，本問の課題は，各頂点に「信頼度」が定ま

International Olympiad in Informatics 2014
13-20th July 2014
Taipei, Taiwan
Day-2 tasks

friend
Language: ja-JP

友達 (Friend)

0 から $n-1$ までの番号がついた n 人からなるソーシャルネットワークを作ろう．ネットワーク内のいくつかの 2 人組が友達となる．人 x が人 y の友達になるならば，人 y は人 x の友達にもなる．

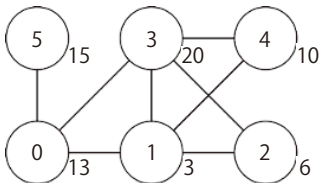
n 人は，0 から $n-1$ までの番号がついた n 回の操作によってネットワークに加えられていく．人 i は操作 i で加えられる．操作 0 では人 0 が加わり，ネットワークのただ 1 人となる．続く $n-1$ 回の操作のそれぞれでは，すでにネットワーク内にいる誰か 1 人が**ホスト** (host) となる．操作 i ($0 < i < n$) では，以下の 3 種類の**方式** (protocol) のいずれかを用いてホストは人 i をネットワークに加える：

- IAmYourFriend：人 i は，ホストのみと友達になる．
- MyFriendsAreYourFriends：人 i は，その時点でのホストの友達**それぞれ**と友達になる．この方式では人 i はホストとは友達に**ならない**ことに注意せよ．
- WeAreYourFriends：人 i は，ホストと友達になる．また，その時点でのホストの友達**それぞれ**とも友達になる．

ネットワークを作ったのち，あるアンケート調査の**サンプル** (sample) を選ぶことになった．ネットワークから何人かをサンプルとして選びたい．友達どうしは関心が似ている場合が多いため，サンプルには友達であるような 2 人組を含めてはならない．n 人のそれぞれには，調査の**信頼度** (confidence) と呼ばれる正の整数が定まっている．信頼度の合計が最大であるようなサンプルを求めたい．

例 (Example)

操作	ホスト	方式	増えた友達関係
1	0	IAmYourFriend	(1, 0)
2	0	MyFriendsAreYourFriends	(2, 1)
3	1	WeAreYourFriends	(3, 1), (3, 0), (3,2)
4	2	MyFriendsAreYourFriends	(4, 1), (4, 3)
5	0	IAmYourFriend	(5, 0)



はじめ，ネットワークは人 0 のみを含む．

1. 操作 1 のホスト（人 0）は人 1 を IAmYourFriend によって加える．人 0 と人 1 は友達となる．
2. 操作 2 のホスト（人 0）は人 2 を MyFriendsAreYourFriends によって加える．ホストの友達である人 1 のみが，人 2 と友達となる．
3. 操作 3 のホスト（人 1）は人 3 を WeAreYourFriends によって加える．人 1（ホスト）および，人 0 と人 2（ホストの友達）が，人 3 と友達になる．

操作 4 と操作 5 も同様である．最終的なネットワークが上の図で示されている（円の中の数は人の番号を，円の隣の数は信頼度を表す）．人 3 と人 5 からなるサンプルを考えると，信頼度の総和は 20 + 15 = 35 となり，これが信頼度の総和の最大値である．

課題 (Task)

各操作の説明と各人の信頼度が与えられたとき，サンプルの信頼度の総和の最大値を求めよ．関数 `findSample` を実装せよ：

- `findSample(n, confidence, host, protocol)`
  - `n`：人数．
  - `confidence`：サイズ n の配列であり，`confidence[i]` は人 i の信頼度を表す．
  - `host`：サイズ n の配列であり，`host[i]` は操作 i のホストの番号を表す ($0<i<n$)．
  - `protocol`：サイズ n の配列であり，`protocol[i]` は操作 i で用いられる方式を表す ($0<i<n$)．
    0 は IAmYourFriend を，1 は MyFriendsAreYourFriends を，2 は WeAreYourFriends を表す．
  - 操作 0 にはホストがいないので，`host[0]` と `protocol[0]` は未定義であり，あなたのプログラムでアクセスしてはならない．
  - この関数は，サンプルの信頼度の総和の最大値を返すこと．

小課題 (Subtasks)

以下の表に示す通り，いくつかの小課題では，操作において方式のうちのいくつかだけが用いられる．

小課題	得点	n	信頼度 (confidence)	用いられ得る方式 (protocol)
1	11	$2 \leqq n \leqq 10$	1 ≦信頼度≦ 1,000,000	3 つすべて
2	8	$2 \leqq n \leqq 1{,}000$	1 ≦信頼度≦ 1,000,000	MyFriendsAreYourFriends のみ
3	8	$2 \leqq n \leqq 1{,}000$	1 ≦信頼度≦ 1,000,000	WeAreYourFriends のみ
4	19	$2 \leqq n \leqq 1{,}000$	1 ≦信頼度≦ 1,000,000	IAmYourFriend のみ
5	23	$2 \leqq n \leqq 1{,}000$	信頼度はすべて 1 である	MyFriendsAreYourFriends と IAmYourFriend
6	31	$2 \leqq n \leqq 100{,}000$	1 ≦信頼度≦ 10,000	3 つすべて

実装の詳細 (Implementation details)

1 つのファイルを提出せよ．提出するファイルの名前は `friend.c`，`friend.cpp`，`friend.pas` のいずれかである．このファイルには課題で指定されたサブプログラムを以下のシグネチャを用いて実装すること．C/C++ のプログラムにおいては，`friend.h` をインクルード (include) する必要がある．

C/C++ プログラム (C/C++ program)

```
int findSample(int n, int confidence[], int host[], int protocol[]);
```

Pascal プログラム (Pascal program)

```
function findSample(n: longint, confidence: array of longint, host: array
of longint; protocol: array of longint): longint;
```

採点プログラムのサンプル (Sample grader)

採点プログラムのサンプルは，以下のフォーマットで入力を読み込む：

- 1 行目：`n`
- 2 行目：`confidence[0], ..., confidence[n-1]`
- 3 行目：`host[1], protocol[1], host[2], protocol[2], ..., host[n-1], protocol[n-1]`

採点プログラムのサンプルは，`findSample` の戻り値を出力する．

図 -3 「Friend」問題文

っているグラフにおいて，頂点の集合であってどの2点も辺で結ばれていないもの（独立集合と呼ばれる）のうち，信頼度の合計の最大値を求めることである．この問題──最大重み独立集合問題──は一般のグラフに対してはNP困難問題であることが知られており，大きいサイズのデータに対して効率的に解くためには，問題文のようにして作られるグラフの特殊性を利用したアルゴリズムが必要である．

まずは，小課題として部分点が設定されている，「MyFriendsAreYourFriends のみ」「WeAreYourFriends のみ」「IAmYourFriend のみ」の3つの場合，つまりグラフの作り方をさらに制限した場合を検討していく．

MyFriendsAreYourFriends のみの場合

方式としてMyFriendsAreYourFriends のみが用いられると，友達関係は一切生じない，すなわち辺がまったくないグラフが作られる．この場合，すべての頂点を選ぶことができるので，信頼度すべての和が答えとなる．

WeAreYourFriends のみの場合

方式としてWeAreYourFriends のみが用いられると，どの2人も友達となる，すなわち完全グラフが作られる．この場合，頂点を1個までしか選ぶことができないので，信頼度すべての最大値が答えとなる．

IAmYourFriend のみの場合

方式としてIAmYourFriend のみが用いられると，生じる友達関係は，i=1, …, $n-1$ に対する人 i と人 i のホスト，という $n-1$ 組である．すなわち，グラフは木になり，根は頂点0，頂点1, …, $n-1$ の親は各々のホストと考えることができる．

木についての最大重み独立集合問題は，以下に解説するように，部分木に対する動的計画法によって効率的に解くことができる．

頂点 i を根とする部分木（頂点 i およびその子孫全体からなる木）を T_i とする．T_i に含まれない頂点を独立集合に含められるか否かに関係するのは，T_i 内の頂点では頂点 i だけである．このことを考慮し，次のように変数を定める：

- T_i の独立集合であって頂点 i を含むものについての信頼度の和の最大値を a_i とおく，
- T_i の独立集合であって頂点 i を含まないものについての信頼度の和の最大値を b_i とおく．

図 -4 に例を示す．

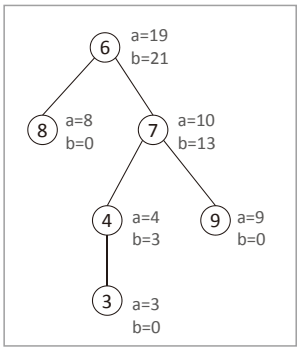


図 -4　木での動的計画法（頂点の値は信頼度を表す）

頂点 i の子が頂点 j_1, …, j_r である（つまり，人 i は人 j_1, …, j_r のホストである）とき，a_i, b_i は a_{j_1}, …, a_{j_r}, b_{j_1}, …, b_{j_r} から以下のように求めることができる：

- 頂点 i を独立集合に含めるとき，頂点 j_1, …, j_r は含めることができない．よって，

$$a_i = （人\ i\ の信頼度） + \sum_{k=1}^{r} b_{j_k}$$

となる．

- 頂点 i を独立集合に含めないとき，頂点 j_1, …, j_r は含めても含めなくてもよい．よって，

$$b_i = \sum_{k=1}^{r} \max\left\{a_{j_k}, b_{j_k}\right\}$$

となる．

特に，頂点 i が葉である場合，a_i=（人 i の信頼度），$b_i = 0$ である．

上述の式を用いて葉から根に向かって a_i, b_i を計算していくことができ，最終的な答えは $\max\{a_0, b_0\}$ となる．頂点0から再帰的に計算してもよいし，本問の場合は頂点の番号が大きい順に計算することができる．この解法は時間計算量・空間計算量ともに $O(N)$ である．

一般の場合

満点解法への鍵は，IAmYourFriend のみの場合，すなわち木の場合の解法を少し違った見方で考えることである：各頂点には「選ぶ場合に得られる信頼度 a_i」と「選ばない場合に得られる信頼度 b_i」が定まっていると考える．

はじめはすべての頂点 i について $a_i \leftarrow$（人 i の信頼度），$b_i \leftarrow 0$ とする．そして，頂点 i が葉であり

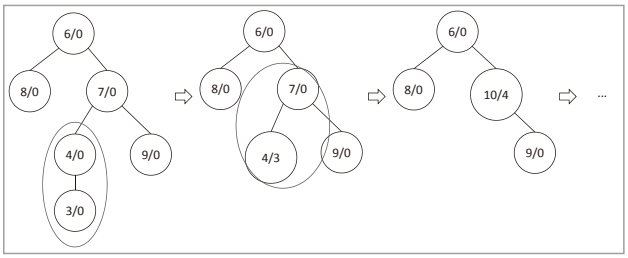


図 -5　頂点をまとめる

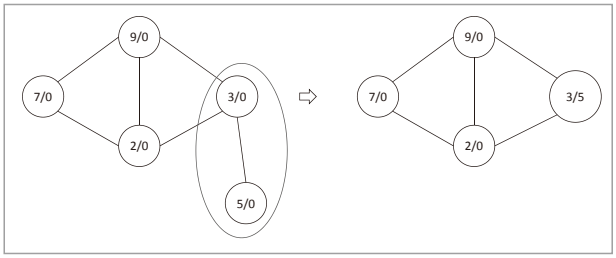


図 -6　IAmYourFriend の場合

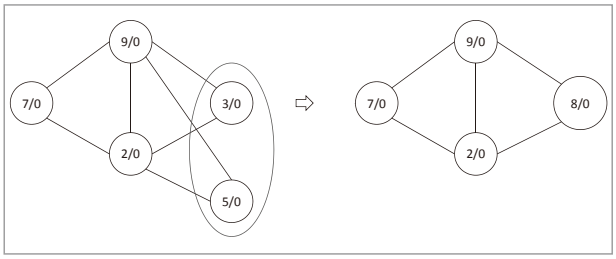


図 -7　MyFriendsAreYourFriends の場合

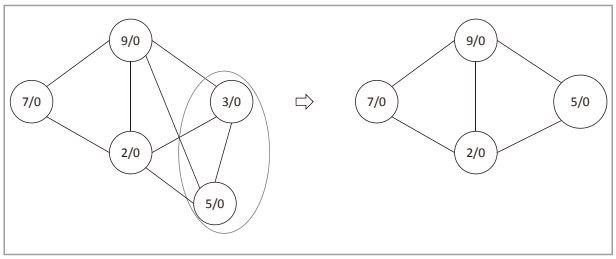


図 -8　WeAreYourFriends の場合

親が頂点 p であるとき，

$a_p \leftarrow a_p + b_i$,

$b_p \leftarrow b_p + \max\{a_i, b_i\}$

という操作によって，頂点 i を消して頂点 p に「まとめる」ことができると考える．ある頂点 i が葉になったとき，a_i, b_i は先ほどの解法のものと同じ値となることが分かる．**図 -4** の木の例では**図 -5** のようになる．

この「頂点を 1 つずつ消してまとめていく」という考えを用いると，方式が 3 つすべて使われ得る場合にも対応することができる．

残っている頂点のうち番号が最大のものを i とし，人 i のホストを p とする．

- 人 i が加わった方式が IAmYourFriend のとき（**図 -6** 参照）

  頂点 i は頂点 p のみと辺で結ばれているので，上で見たように，

  $(a_p, b_p) \leftarrow (a_p + b_i, b_p + \max\{a_i, b_i\})$

  として頂点 i を消して頂点 p にまとめることができる．

- 人 i が加わった方式が MyFriendsAreYourFriends のとき（**図 -7** 参照）

  頂点 p と頂点 i は他の頂点からとの接続関係はまったく同じであるから，頂点 i を頂点 p にまとめることができる．p と i の両方または一方を選んだ場合はまとめた頂点を選んだことに，p も i も選ばない場合はまとめた頂点を選んでいないことになるので，

  $(a_p, b_p) \leftarrow (\max\{a_p + a_i, a_p + b_i, b_p + a_i\}, b_p + b_i)$

  とすればよい．

- 人 i が加わった方式が WeAreYourFriends のとき（**図 -8** 参照）

  この場合も，頂点 p と頂点 i は他の頂点からとの接続関係はまったく同じであるから，頂点 i を頂点 p にまとめることができる．p と i は辺で結ばれているので，両方を選ぶことはできない．よって，

  $(a_p, b_p) \leftarrow (\max\{a_p + b_i, b_p + a_i\}, b_p + b_i)$

  とすればよい．

上記の手順で頂点を番号が大きい順に消していき，答えを $\max\{a_0, b_0\}$ として得られる．時間計算量・空間計算量ともに $O(N)$ である．

（2014 年 11 月 4 日受付）

保坂和宏　hos@hos.ac

東京大学理学部数学科 4 年．国際情報オリンピックエジプト大会（2008）・ブルガリア大会（2009）金メダル，ACM-ICPC 2013 金メダルなど，多くの世界的なプログラミングコンテストに参加している．