競技プログラミングを始めよう

終に鮭

電子計算機研究会

December 2, 2020

自己紹介

- 終に鮭
- 情報系学科 1 回生
- 競プロ歴 8ヶ月 (最初のコンテストは 4/4 の ABC161)
  - AtCoder しかやってない
  - 現在のレートは 877
  - ごくまれに yukicoder の問題も解く
- プログラミング歴は 4〜5 年 (?)
  - 現在メインで使っている言語は Python, C++, JavaScript
  - 使ったことのある言語は C, Java, Visual Basic, PHP

今日の内容

1. 競技プログラミングとは？
   - 競プロのいいところ
   - コンテストの形式
   - オンラインジャッジ
2. AtCoder をやってみよう
   - 対応言語
   - 書き方と注意点
   - 提出の練習
   - 典型問題を解いてみよう
   - 解説
3. 本番のコンテストについて
   - 参加方法
   - おすすめの言語
   - 注意点
   - 役に立つツール

Section 1

競技プログラミングとは？

競技プログラミングとは

プログラミング早解きコンテストのこと !!

競プロのいいところ

- プログラムが書けるようになる
  - 化生の友人 (未経験) に AtCoder を教えたらプログラミング 1 が楽単になった
- アルゴリズム・データ構造を学べる
  - 次回があればこの部分を掘り下げて勉強会を開くかも
- 計算量を意識して書くようになる
  - 逆に、$1 \leq N \leq 10^6$ なら $O(N \log N)$ で 2 秒に間に合うな、とメタ読みできる
- 数学力がつく
- 就活に役立つことがある (らしい)
  - レートが実力の証明になる

コンテストの形式

参加者全員に同じ課題が与えられ、より早く要求を満たすプログラムを記述することを競う

- アルゴ：短時間、最適解が出せる
- マラソン：中長時間、最適解を目指す
- コードゴルフ：より短いコードを記述することを競う

オンラインジャッジ

- AtCoder(日本)：週 1,2 回、全問日本語に対応
- CodeForces(ロシア)：開催頻度がすごい
- TopCoder(アメリカ)：マラソンが充実してる
- Aizu Online Judge(日本)：JOI や ICPC の過去問が解ける

Section 2

AtCoder をやってみよう

対応言語

多すぎて TEX がクラッシュするので一部のみ抜粋 (独断で選びました)

主な対応言語（ABC183 時点）

- C / C++
- Java
- C#
- Rust
- Python / PyPy
- Cython
- JavaScript (Node.js)
- PHP
- Ruby
- Perl

延べ 57 言語が使用可能 [a]

[a] https://atcoder.jp/contests/abc182/rules

プログラムの書き方と注意点

- 「入力」は標準入力から受け取れる、「出力」は標準出力に出す
  - コマンドライン引数は使わない
  - 標準入出力を備えていない言語の場合はちょっと特殊（practice contest 参照）
  - 例えば JavaScript では `/dev/stdin` からファイル入力、`console.log` で出力
- (B 問題ぐらいから) 制約に注意する
  - 算術オーバーフロー
    - 黒魔法 `#define int long long`
  - 実行時間制限超過 (TLE：Time Limit Exceeded)
    - 1 秒間に処理できる量は 10^7 から 10^8 ステップ程度
  - スタックオーバーフロー
    - 再帰の制限は緩められる場合がある
  - `out_of_range`

提出の練習

- 常設中のコンテストから practice contest に飛ぶ
- 問題タブから「A - Welcome to AtCoder」を選んで解いてみよう
  - 自前のエディタで書くのがおすすめ
  - 事前調査で回答いただいた言語のサンプルをここに用意してあります

Figure: AtCoder のコンテストトップページ

提出の練習

- 必ずコードテストする
  - 手元の実行環境を使う
  - AtCoder のコードテスト機能を使う
- 問題ページ下部から提出

「いかにも競プロ」な問題を解いてみよう

- どんな問題？
  - 文字列操作系問題
  - 数学の問題
    - 小数の計算、幾何、方程式、数え上げ
  - データ構造を活かす問題
    - 配列、集合、スタック・キュー、Union-Find、全域木
  - 典型アルゴリズムを使う問題
    - ソート、探索、累積和、しゃくとり法、いもす法、動的計画法、貪欲法、最短経路問題、最大流問題

初心者におすすめの問題

- AtCoder Problems(有志サイト) にバーチャルコンテストを用意したのでやってみよう
- 制限時間は 1 時間
- 全問解ききることは想定していないので解けそうな問題から解いてください
- インターネットや本で調べても OK
- 全問解説すると時間がかかるので、全員解けた問題と聞いても誰もわからなさそうな問題は解説しません

解説

解説を始めます

配点基準

Table: 今回採用した配点の基準

配点	問題数	問題の要素
50	3	if 文のみで解ける 数学的考察が不要
100	4	if 文のみで解けるが、場合分けが難しい 文字列操作が必要 簡単な数学的要素が含まれる
200	7	配列を扱う ループまたは再帰関数が必要 単純な多重ループが必要 小数を扱う 数学的考察が含まれる

配点基準

Table: 今回採用した配点の基準

配点	問題数	問題の要素
300	3	ナイーブな実装が通用しない コーナーケースへの対応が必要
400	3	特殊なデータ構造を用いる 比較的複雑な典型アルゴリズムを用いる $\mathrm{mod}\ 10^9 + 7,998244353$ 複合的なスキルが必要

50 点問題

ABC Swap

Table: 箱とその中身

A	X
B	Y
C	Z

A	Y
B	X
C	Z

A	Z
B	X
C	Y

箱 A, B, C の中身 X, Y, Z について、A と B、A と C を順に入れ替えたときの中身を求めよ

- `swap` 関数を使ってシミュレーションする
- 順番を決め打ちして出力する

AC or WA / Sheep and Wolves

それぞれ $N = M$ か、$W \geq S$ か判定せよ

- 条件演算子 `cond ? a : b` を使うと短く書ける

100 点問題

Accepted...?

長さ 6 の文字列 S に '1' はいくつ含まれるか

- 6 回 if 文書く卍
- for 文を回す
  - 文字列のまま判別 (おすすめ)
    - 範囲 for 文でもよい
  - 10 進整数の各桁を判別
- `count_if` のような関数を使う
  - イテレータとラムダ式の知識を要するので初心者にはおすすめしない

Repeat ACL

文字列 ACL を K 回繰り返した文字列を出力せよ

- 5 回 if 文書く卍
- K 回 "ACL" を出力する
- 空文字列に K 回 "ACL" を連結した文字列を出力する
  - Python では 'ACL' * K でよい

CODE FESTIVAL 2015

文字列の末尾の 2014 を 2015 に置き換えよ

- 末尾 4 文字が ’2014’ であることが確定しているので、末尾 1 文字を ’5’ に書き換えれば良い
  - C++の場合末尾 1 文字は `S[S.length() - 1]`
  - 文字列をスタック的に扱って `S.pop_back(); S.push_back('5');` するという方法もある

Number of Multiples

1	2	3	4	5	6	7	8	9	10	11	12	···

Figure: 自然数の列

$[L, R]$ の範囲にいくつ d の倍数があるか

- L, R, d から計算 (時間計算量 $O(1)$)
  - L 以上の最小の d の倍数は $\lceil L/d \rceil \times d = \lfloor (L + d - 1)/d \rfloor \times d$
  - R 以下の最大の d の倍数は $\lfloor R/d \rfloor \times d$
  - 求める数は $\lfloor R/d \rfloor - \lfloor (L + d - 1)/d \rfloor + 1$
- $i (L \leq i \leq R)$ について、d の倍数か判定 $(O(R - L))$
- d の倍数 $n = di (1 \leq n \leq 100)$ について、 $L \leq n \leq R$ か判定 $(O(\frac{1}{d}))$

200 点問題

Fourtune Cookies

A, B, C, D を 2 つグループに分けて、それぞれの総和が等しくなることがあるか

- $A \leq B \leq C \leq D$ としても一般性を失わない (これはソートすれば簡単に得られる)
  - 食べるクッキーと残るクッキーの美味しさの総和が等しくなることがある
    $\iff A + D = B + C$ or $D = A + B + C$
- bit 全探索で食べるクッキーと食べないクッキーの全ての組み合わせを検証する
  - 4 要素なので組み合わせは $2^4 = 16$ 通りだけ

天下一プログラマーコンテスト 1998

$a_0 = 100, a_1 = 100, a_2 = 200, a_n = a_{n-1} + a_{n-2} + a_{n-3}$ が与えられるので a_{19} を計算せよ

- いわゆる動的計画法 (DP:Dynamic Programming) を使う

動的計画法

- 帰納的な関係を利用
- 部分問題の計算結果を記録

  - 配列を使って $a[i] = a[i-3] + a[i-2] + a[i-1]$ を $i = 3, \ldots, 19$ でやって $a[19]$ を求める
- 再帰関数を使う
  - $f(0) = 100, f(1) = 100, f(2) = 200, f(n) = f(n-1) + f(n-2) + f(n-3)$ として $f(19)$ を実行
  - ただし適切にメモ化などを行わないと 69748 回の関数呼び出しが行われてよくない
- 手計算しておいて出力卍

Addition

偶奇が等しい 2 数を足し合わせていき、最後に 1 つだけ残るか

- 偶奇が等しいかつそのときに限り、2 数の和は偶数である
- 黒板に数が 1 つだけ残ることは以下に同値である
  - $\sum A_i$ が偶数
    - オーバーフローに注意
  - A_i のうち、奇数が偶数個だけ存在する

Product Max

$a \leq x \leq b, c \leq y \leq d$ のときの xy の 最大値を求めよ

Table: a, b の範囲に対する $\max(xy)$

	$0 \leq c \leq d$	$c \leq 0 \leq d$	$c \leq d \leq 0$
$0 \leq a \leq b$	bd	bd	ad
$a \leq 0 \leq b$	bd	$\max(ac, bd)$	ac
$a \leq b \leq 0$	bc	ac	ac

- いずれにせよ $\max(ac, ad, bc, bd)$ をとればよい

高橋くんと文字列操作

回転を繰り返して所定の文字列が得られるか

- 回転、循環シフト (rotation, circular shift) と言われる操作

回転、循環シフト

- ビット列、配列またはキューの先頭または末尾の要素を、末尾または先頭へ移動する操作

  - C++では `algorithm` ヘッダ内の `rotate` 関数で実現できる
  - Python では `str.rotate` メソッドで実現できる

- $|s|$ 回の操作で元の文字列に戻る
- 回転の計算量が $O(|s|)$ 、文字列比較の計算量が $O(|s|)$ ならば全体の時間計算量は $O(|s|^2)$
- $s+s$ の $[|s|-i, 2|s|-i)$ の範囲を取り出しても i 回操作したのと同じ文字列が得られる

Magic 2

A, B, C のうちいずれかを 2 倍する操作を K 回まで繰り返し、$A < B < C$ にすることができるか

- $A < 2^b B < 2^c C (b + c \leq K)$ なる b, c が存在すればよい
  - K 回のループの間に、$A < B$ となるまで `B *= 2` をし、$A < B$ になったら `C *= 2` する
  - $A < B < C$ ならば成功、そうでなければ失敗

Takahashikun, The Strider

1m 進みその場で X 度回転することを繰り返したとき、何回目で元の位置に戻るかを求めよ

- 高橋くんは円周上を移動するから、KX が 360 の倍数となるような最小の K を求める
- $X, 360$ の最小公倍数を l 、最大公約数を g とすると、$KX = l, lg = 360X$ より $K = 360/g$
- 最小公倍数を求める関数はたいてい用意されているし、ユークリッドの互除法で簡単に実装できる
- もちろん、`X*i` が 360 の倍数になるかを順に判定してもよい

300 点問題

Connection and Disconnection

S を K 回繰り返した文字列について、最小で何文字置き換えればどの隣り合う 2 文字も相異なるようにできるか

- K が最大で 10^9 になるので、愚直にシミュレートできない
- T は S を K 回繰り返したものなので S についての解を K 倍する
  - ただし S の始端と終端が同じである場合は注意が必要
    - S の両端を除いた部分が K 回、始端と終端が各 1 回、S の終端と S の始端のつなぎ目が $K-1$ 回出る
  - S が全て同じ文字で構成される場合も分ける必要がある

Figure: S の始端と終端が同じである場合の T の例

```
aaa b cccc ddddd eee aaa|aaa b cccc ddddd eee aaa
aza b czcz dzdzd eze aza|zaz b czcz dzdzd eze aza
```

DNA Sequence

A,T,C,G からなる文字列 S の連続する空でない部分文字列 T について、A と T、C と G の数が等しいものの数を求めよ

- $N \leq 5000$ より ${}_N\mathrm{C}_2 (< 2 \times 10^7)$ 通りを全探索しても間に合う
  - ただし判定を $O(1)$ でする必要がある
  - S の $[0, n)$ の範囲に含まれる A,T,C,G の個数をあらかじめ $O(N)$ でカウントしておけば $[a, b)$ の範囲でも $O(1)$ で求められる
- 余談：Python は配列の処理がめちゃくちゃ遅いので PyPy を使わないと通りませんでした
  - NumPy なら通るかも

One Quadrillion and One Dalmatians

名前が順に a,b,...,z,aa,ab,...,az,...,za,...,zz,aaa,... となるとき、N 番目の名前を求めよ

- 一見 10 進数を 26 進数に変換する問題に見える
- a と aa は異なる
  - a，aa，aaa はそれぞれ 1,2,3 桁の 1 番目なので先に名前の長さを求めれば 26 進数への変換に持ち込める
- 名前が L 文字である条件は

$$\sum_{i=0}^{L-1} 26^i \leq N < \sum_{i=0}^{L} 26^i$$

であるから、$N - \sum_{i=0}^{L-1} 26^i$ を 26 進数へ変換すればよい

400 点問題

Connect Cities

すべての都市を相互に移動できるようにするには最小で何本の道路を新しく作ればよいか

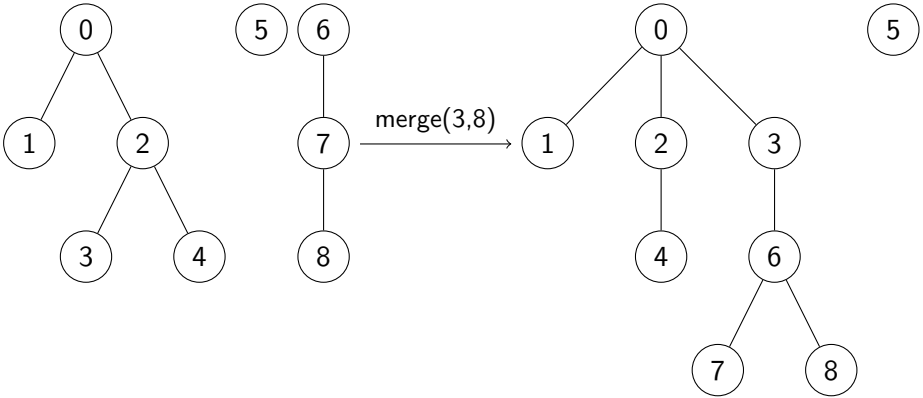
- Union-Find を使う典型問題

Union-Find(Disjoint-set Data Structure)、素集合データ構造

- Union と Find の操作が定義されている集合族データ型
- Union(Merge): 2 つの集合をまとめて 1 つにする
- Find: ある要素がどの集合に含まれるかを探索し、代表元を返す
- しばしば森 (閉路を持たないグラフ、木の集合) として表現される

- $\{0\}, \ldots, \{N-1\}$ の N 個の集合を持った Union-Find を生成する
- 各道路について Union して、最終的なグループ数 g を求める。解は $N - g$

Union-Find 森



Div Game

整数 N は相異なる素数の冪で何回割り切ることができるか

- 素因数分解すれば簡単に解ける
  - $N = p_1^{e_1} p_2^{e_2} \cdots p_n^{e_n} (i \neq j \implies p_i \neq p_j)$ で表されるとき、$e_i \geq \sum_{k=1}^{a_i} k$ を満たす最大の a_i についてその和を求めればよい
- $N \leq 10^{12}$ であるから、$[2, N]$ の範囲を試し割りすることはできない

定理

自然数 $N \geq 2$ について、N が素数でない $\iff$ N は $\sqrt{N}$ 以下の素因数を持つ

- 素因数は疎であるから、連想配列に格納するとよい
  - p_i は再利用しないので e_i のみ配列に格納しておいても十分

Redistribution

総和が S で全ての項が 3 以上である数列はいくつ存在するか。10^9+7 で割ったあまりを求めよ

- 条件を満たす数列の項数 N の範囲は $1 \leq N \leq S/3$
  - 項数 N の数列 (a_n) について、条件は次のように言い換えられる

$$a_i \geq 0, a_1 + \cdots + a_N = S - 3N$$

  - このような条件を見たす数列は $S-3N$ 個の $\circ$ と $N-1$ 本の $|$ の並びに対応し、その場合の数は ${}_{S-2N-1}\mathrm{P}_{S-3N}/(N-1)!$
  - 順列は大きな数になるから途中であまりを取る必要がある
  - 割り算がとてもめんどくさい (モジュラ逆数を求めるには冪を求める必要がある)
    - フェルマーの小定理を用いて $x/y = xy^{-1} \equiv xy^{p-2} \mod p$

フェルマーの小定理

a と p が互いに素であるとき、$a^{p-1} \equiv 1 \mod p$

Redistribution

- DP を使うことで順列の計算を回避できる
  - 総和が S となるようなすべての項が 3 以上である空でもよい数列の場合の数を $A[S]$ とおく
    - すなわち解は $A[S]$
    - 空な数列 () の和は 0 であるから $A[0] = 1$
  - 総和 $i-3$ で条件 (各項が 3 以上) を満たす数列について考える
    - この数列の末尾に 3 を加えると、総和 i で条件を満たす新たな数列が生成できる
  - 総和 $i-4, \ldots, 0$ で条件を満たす数列の末尾に $4, \ldots, i+3$ を加えても同様に総和 i で条件を満たす
  - 以上より $A[i] = A[i-3] + A[i-4] + \cdots + A[0], A[2] = A[1] = 0, A[0] = 1$
    - 遷移が $O(S)$、各項の計算量が $O(S)$ であるから計算量は $O(S^2)$
  - $A[i-1] = A[i-4] + \cdots + A[0]$ より、$A[i] = A[i-3] + A[i-1]$ に変形できる
    - この場合各項の計算量が $O(1)$ に抑えられ $O(S)$ で解ける
- $10^9 + 7$ で割ったあまりを求めることに注意

Section 3

本番のコンテストについて

参加方法

1. トップページの左側の「予定されたコンテスト」から選ぶ
   - ARC(橙)、AGC(赤) 系は難しいので ABC(青) 系のコンテストに参加しよう
   - レートが下がってもいいなら全部参加するのもあり
   - 現状は Nosub 撤退すれば Unrated になる
     - そのためにある程度解けてからまとめて提出する手法がある (tourist 出し)
2. 参加登録する
   - 企業コンの場合は個人情報を入力することも
3. 開催時間にもう一度開く

おすすめの言語

- C が書ける人：C++
  - 便利機能が多い
    - 文字列型、STL のコンテナクラスや関数、クラス、ラムダ式など
  - C 言語のコードはそのまま動く
  - 競プロでは一番人口が多い
  - 最近 ACL(AtCoder Library) が導入されて Union-Find などを実装しなくて良くなった
- ほとんど書けない人：Python
  - ライブラリが豊富で便利機能が多い
    - range 型、ジェネレータ、numpy モジュール
  - C と比べると遅いが、現状解けない問題は報告されていない
- 速度狂：Rust
  - C/C++より更に早い
  - 暗黙の型変換を一切しないのでめっちゃコンパイルエラーが出る
  - 全ての型の変数はデフォルトで immutable(書き換え不可)

注意点

- コンテスト開催中に Twitter などで余計なことは喋らないようにしよう
  - 全員に公開されている情報は OK
    - 「A 問題 AC した」は大丈夫、「A 問題 WA/TLE/RE した」はだめ
    - 「B 問題は余弦定理を使う」、「C 問題は bit 全探索で間に合う」などもだめ
    - ソースコードを上げたり生配信したりは当然だめ
  - 詳細はコンテストページ下部の「ルール」から確認できる
- 検索や自作ライブラリの使用は自由
  - ただしコンテスト中の情報共有は不可
  - コードスニペットやテンプレートを用意しておくとよい

役に立つツール

- AtCoder Problems
  - コンテスト終了後、問題の difficulty を推定して色で表示してくれる
    - API が提供されているので、AtCoder のページで色付けするスクリプトなども作られている
  - 過去問が参照しやすい
    - 解いたことのある問題が色付けされる
  - バーチャルコンテストが開催されている
- atcoder-cli
  - Node.js が必要
    - online-judge-tools が必要で、online-judge-tools の利用には Python3 が必要
  - ディレクトリの作成、実行テスト、提出がコマンドラインでできるツール
  - 問題を選択するとサンプルケースを拾ってきてコマンド一つでテストできる