講義 1

プログラミングコンテストの取り組み方

チューター : 城下慎也 – IOI 2011 タイ大会 日本代表

2017/3/19

目次

- 本講義の流れ
- 最近の IOI の動向について
- 実装のテクニック
- 発想のテクニック
- 平方分割について
- 実際に春合宿に参加するにあたる諸注意
- 演習

本講義の流れについて

- 本講義はプログラミングコンテストを行う際の、基本的な内容や、個人的に気になることを確認していくものとなります。
- 人によっては知っている内容であることも多いと思います。

- 流れとしては、講義前半を説明、講義後半部分を演習とする方向性で考えております。
- アルゴリズム等についての解説は、基本の流れを全体に説明し、気になる内容とかは個別に演習時間等に対応するという形を考えております。

最近の IOI の動向について

- 昨年講義で説明した後、IOIシラバス[1]に少量の変更がありました。
- 赤文字で書かれた箇所が変更点となります。

- Out of focus が 1 つ増えた点を除くと、文字列処理とグラフアルゴリズムの追加がメインです。
  - 文字列：部分文字列検索やトライ
  - グラフ：橋や連結性など
- 文字列が追加されていく流れというのは、前回の IOI 2016 でもその傾向があり、実際に文字列を通信に使用する課題[2]が出題されました（e.g. Day 2 お絵かきパズル）。

文字列

- 少なくとも競技プログラミングを行うに当たっては、char 型の配列であるという認識で問題ありません。
- しかしながら、string 型を含め、独自仕様を知らないと思わぬところでバグを踏むことになります。

  有名？な例:
  - == を用いた比較
  - std::string の出力
  - for 文内に strlen を使用する
  - 制御文字の扱い

    :
- この辺りは、前もって仕様を確認しておき、今まで使ったことないものを本番で使う事態を回避しましょう。

実装のテクニック

実装のテクニック

- どう実装するか、というのはプログラミングコンテストで重要となります。
  - アルゴリズムを関数に小分けにし、関数ごとに仕様を決めておけば、見るべきポイントを絞れてデバッグしやすくなります。
  - ただし、小分けにしたがために生じるバグもあります（グローバル変数周りとか）

- 実装方針を前もって固めておくと実装しやすくなります。
  - ICPC 等のチーム戦などで直接 PC を触れない環境時にやることの一つですが、個人戦でもやる価値があることだと思います。
  - 急がば回れ、バグを踏みにくくなるように対策しておけば、結果的に早くなることも多いです。

発想のテクニック

発想のテクニック

- アルゴリズムの設計書や、数学的な問題の解き方などでも基本的なこととなりますが、主問題を限定することで大抵突破口が見えてきます。
  - 与えられる条件を厳しくする（e.g. サイズを小さくする、形式を限定する）
  - コーナーケースを考えてみる。

- 限定した場合、解けないことが発覚するならそれは筋の悪い方針だし、解けることが言えるなら主問題を解くヒントとなります。
- 競技プログラミングに限定して言えば、部分点そのものが方針のヒントとなっていることも多いです。

発想のテクニック（競プロ特化型）

- 特定の語句や、似た問題を考える。

  例：「最大値の最小値」や「K 番目の要素」→二分探索
  - 実は JOI 2014/2015 本選 4「舞踏会」も「中央値」だと考えれば上の考えに合致します。

- この方針自体は正しい面もあります。というのも、「構造が似ているなら相性の良いアルゴリズムも似ているだろう」と予測できるからです。

発想のテクニック（データ構造面）

- 代表的なものとして、N log N （セグメントツリー）と NvN （平方分割）の 2 種類があります。
- 単純な理論計算量で言えば N log N < NvN となり前者に軍配が上がりますが、セグメントツリーは多段であるがゆえに相性の悪い処理とかもあります（区間の反転などが最たる例）。

- どちらが良いかはやりたいことにかなり依存するので常に両方考慮するべきです。

発想のテクニック（主問題の分割）

- 代表的なものとして、log N（二分探索や分割統治でよく出る形）と √N（平方分割）の 2 種類があります。
- このようなフレームワークは JOI や IOI で頻繁に登場します。
  - 分割統治の例：IOI 2011 Day 1 Race[3]
    昨年講義した HL 分解も log N の分割の亜種であると言えます。
  - 平方分割の例：昨年度春合宿[4]より
    Day3 回転寿司
    Day4 雪降る道路（この問題も平方分割の亜種であると言えます）
- ここでは平方分割を掘り下げていきます。

平方分割

平方分割　目次

- 基本事項
- Baby Step Giant Step
- Mo のアルゴリズム

基本事項

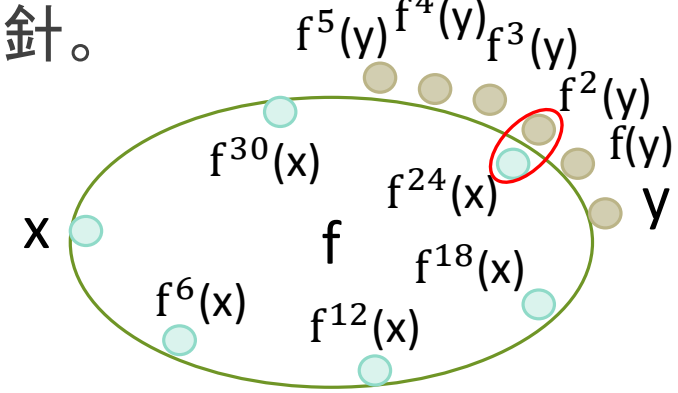
- 一般には大きさが $\sqrt{N}$ くらいのグループに分けて、グループ単位で処理できる箇所はグループ単位で処理し、残りを個別に処理する方針で解くフレームワークです。
- 例えば RMQ (区間の最小値を求める問題) では、グループ単位の最小値を$\sqrt{N}$ ごとに持っておいて、それで処理しきれない範囲（区間の両端付近）を個別に処理します。
- ざっくり表現をすると、「範囲を飛ばし飛ばしに処理する処理と微小なずれを補正する処理を組み合わせたフレームワーク」となります。

1			1			2		
3	1	4	1	5	9	2	6	5

例 : RMQ

Baby-step Giant-step[5]

- 元々は離散対数問題のアルゴリズムです。
- Wikipedia[5] での Note にもありますが、一般に有限サイズの巡回群に使用可能です。
- 平たく言えば、「N 回繰り返すと一周する周期的なある操作 f を x から？回行うと y になりました。？を O(√N) で求めてください」という問題で、「$f^{\sqrt{N}}$ が使えるときに利用できるアルゴリズム」です。
- $f^{\sqrt{N}}(x), f^{2\sqrt{N}}(x), \ldots, f^{N}(x)$ を出しておいて、$y, f(y), \ldots, f^{\sqrt{N}-1}(y)$ との一致を検出するという方針。



N=36 での一例

- 左の例だと、$f^{24}(x)=f^2(y)$ だから、差を求めて$f^{22}(x)=y$ だと言えるので、？=22 と求まります。

Mo のアルゴリズム[6]

- 最近競技プログラミング界隈で注目され、有名になりつつあるアルゴリズムです。
- 元とされるのは Codeforces のある記事[7]です。
- およそ M 個ある要素列の区間に対する処理を Q 個処理する際に、Q 個の処理順番をうまく選択することで全体で O(N√N) (N=max{M,Q} としたとき) とするアルゴリズムとなります。
- ただし、「区間の左端・右端を 1 動かすことを O(1) でできる場合」を前提とします。

※ 上記制約を満たさない場合でも使用可能なケースもあります。

Mo のアルゴリズム[6]

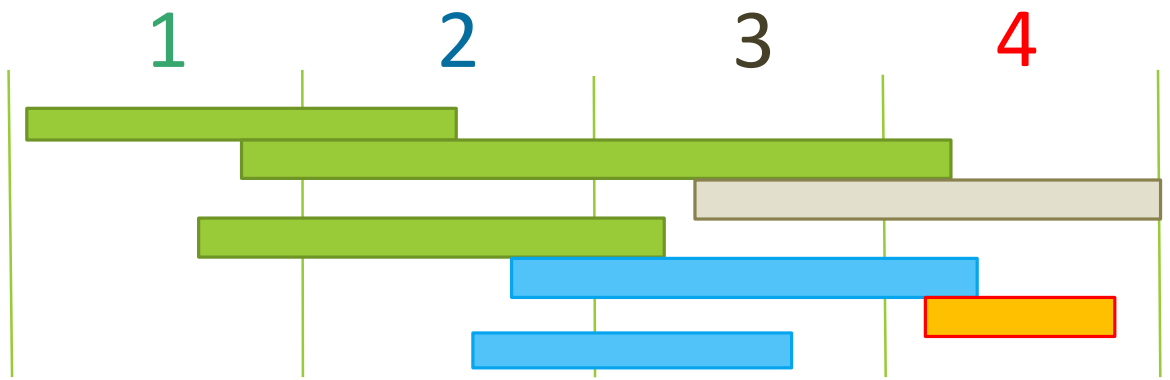
- 最近競技プログラミング界隈で注目され、有名になりつつあるアルゴリズムです。
- 元とされるのは Codeforces のある記事[7]です。
- およそ M 個ある要素列の区間に対する質問クエリを Q 個処理する際に、Q 個の処理順番をうまく選択することで全体で O(N√N) (N=max{M,Q} としたとき) とするアルゴリズムとなります。
- ただし、「区間の左端・右端を 1 動かすことを O(1) でできる場合かつ、質問クエリの処理順番を任意に決定できる場合」を前提とします。

※ 上記制約を満たさない場合でも使用可能なケースもあります。

- ナイーブな実装の場合、毎回左端を O(N) 回、右端を O(N) 回動かさなければならず、全体が $O(N^2)$ となってしまいます。

Mo のアルゴリズム[6]

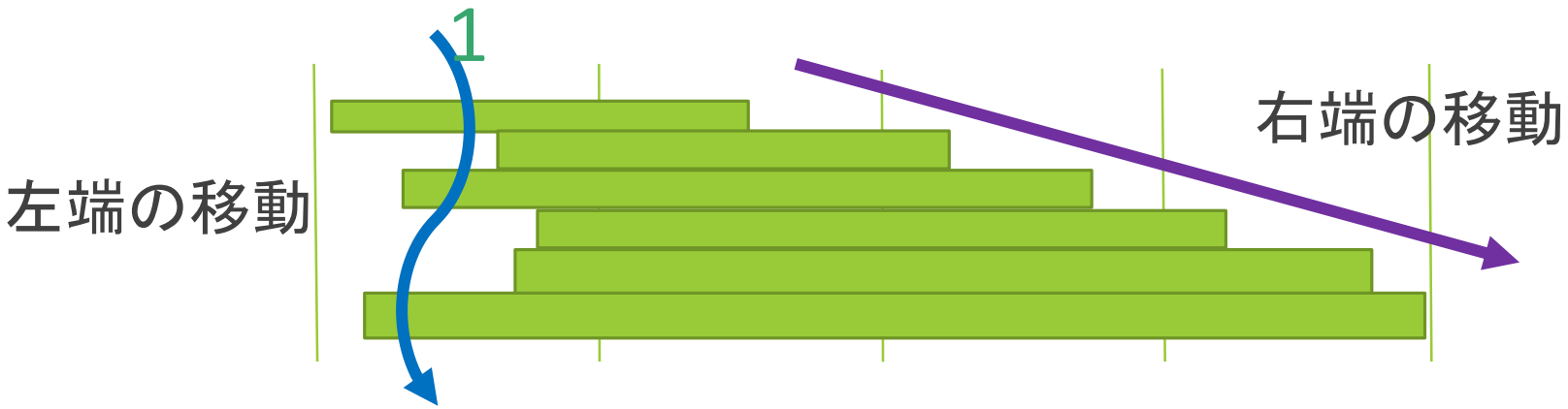
- 質問クエリの処理順番を任意に決定できるので、左端及び右端の総移動距離を抑えるような順番で処理することができます。
- 全体を √N ごとの領域に区切って、左端の位置の属する領域ごとにグループ分けします。



グループ分けの一例

Mo のアルゴリズム[6]

- 同一グループ内では、どの順番でクエリを処理しても、各クエリ間での左端の移動回数は √N 回以下です。
- 右端を処理する際、右端でソートして左から順に処理すれば、右端は左から右へ移動するだけでよくなる。
- グループごとに処理していくことにすれば、左端も右端も移動回数が O(N√N) となる。



グループ分けの一例

まとめ

- 様々なことに言及しましたが、結局は数をこなして触れることが一番だと思います。
- 今回触れたアルゴリズムは発展的な要素となりますが、出題範囲を広げつつある最近の IOI の傾向を鑑みるに、未来の IOI では出題対象となっている可能性も考えられると思います。
- そうでなくとも別解として使用可能な場合もある上に、アルゴリズムなどの知識は一般に知っておいて損はないので（そもそも他のプログラミングコンテストではシラバスがないことがほとんど）、貪欲に知識を吸収していってください。

参考

- [1]MisoF’s IOI Syllabus page(https://people.ksp.sk/~misof/ioi-syllabus/)
- [2] https://www.ioi-jp.org/ioi/2016/tasks/index.html
- [3] https://www.ioi-jp.org/ioi/2011/tasks/index.html
- [4] https://www.ioi-jp.org/camp/2016/2016-sp-tasks/index.html
- [5] https://en.wikipedia.org/wiki/Baby-step_giant-step
- [6] http://codeforces.com/blog/entry/7383