競技における諸注意

2015/03/19

一般的な注意

- 競技時間は長いです
  - 競技前の体調管理を怠らないように
  - 落ち着いて取り組みましょう
    - 早解きを競うコンテストではありません
  - 水分補給・トイレ休憩は適切に
    - 気分転換・軽い散歩としての利用

一般的な注意

- 合宿期間も長いです
  - 合宿中の体調管理を怠らないように
    - 特に睡眠時間の確保
    - 調子が悪いと感じたら早めにスタッフに伝える
  - 1 日くらい大きな失敗をしても挽回は可能

問題を読む

- Overview Sheet を見ましょう
  - 課題の形式・時間制限・メモリ制限はときに重大な情報
- 問題は全部読みましょう
  - 難易度順に並んでいるとは限りません
  - 「簡単な・得意な問題を読んでいなかった」は痛い

問題は全部読みましょう

問題を読む

- 読み落とし・読み間違いは防ぐ
  - 慣れている人でもよくやります
  - 読んでいるうちに重要な条件を忘れがち
    - 重要そうなことはメモ・印をつける
  - 入出力例を手で解いて確認

問題を読む

- 最近は通信型の課題が増えています
  - 問題文が長くなる傾向にあります
  - 与えられる関数，実装すべき関数，与えられる値，求めるべき値を整理
  - 採点プログラムのサンプル（sample grader）の仕様とソースコードにも目を通す

問題を読む

- 解釈に不安があれば早めに質問しましょう
  - 損にはなりません

アルゴリズム

- 解法が思いつかないとき
  - 明記された部分点・小課題に頼る
    - 「まず部分点解法→効率をよくする」というパターンは非常に多い
    - 入力のどのパラメーターの大きさが重要なのかわかることも多い
    - 実装すれば得点になる
    - ただし頼り過ぎないこと
      - 満点への道筋が複数あるかもしれない
      - 満点は部分点と全く違うアプローチということもときどきある
      - 場合によっては時間の無駄になってしまうかもしれない

アルゴリズム

- 解法が思いついた（と思った）とき
  - 実装にかかる時間を見積もる
  - より楽な・確実な・簡潔な方法はないか
  - 入出力例をそのアルゴリズムに従って解いて確認
    - 入出力例で捕まるような誤りを検出
      - 実装してから気づくと時間の消費が大きい
    - 計算すべき主要な値を手計算しておく
      - デバッグにも役立つ

実装・デバッグ

- なるべく書き慣れているコードを書きましょう
- どうしても不慣れな処理を書く場合はその部分がデバッグしやすいように
  - 関数に切り分けるなど
- 複雑になりそうならナイーブな実装をしてみてから効率をよくすることを考えるのもあり
  - 愚直な実装と比較できるようなコードにするとよい

実装・デバッグ

- コンパイラを頼りましょう（個人差あり）
  - g++ -std=c++11 -Wall -O2 -o hoge hoge.cpp
    - 警告を出してくれます
      - 例：signed int と unsigned int の比較
        - オーバーフローを見落としやすい
    - -Wextra というのもあります
  - scanf 等の戻り値を使っていないと怒られるのが嫌なときは -Wno-unused-result
  - 通信型の課題等，コンパイル方法が指定されている場合も，-Wall 等は使えます

実装・デバッグ

- gdb などのデバッガが使える人は頼りましょう
  - Segmentation Fault する場所を見つけやすい

- いわゆる printf デバッグも有用
  - 行番号 __LINE__ やファイル名 __FILE__ を活用
    - `fprintf(stderr, "%d¥n", __LINE__); fflush(stderr);`
    - `cerr << __LINE__ << endl;`

実装・デバッグ

- Technical Info Sheet の内容も再確認
  - 手元実行時のスタックオーバーフロー対策
    - ulimit -s unlimited
  - 巨大な入手力
    - cin / cout を scanf / printf に
  - オーバーフローに注意
    - int は 2.1×10^9 くらい, long long は 9.2×10^{18} くらいまで

提出

- 提出先は正しいか確認
  - 特に終了直前
- 提出するコードが最新のものか確認

テスト

- 不正解のとき（or 完全フィードバック以外のとき）
  - 小さいテストケースをたくさん作る
    - 愚直解と比較
  - 時間やメモリの使用量が大きいテストケースを考える
    - 漸近計算量が大丈夫なはずなのに通らないときは定数倍最適化の余地を検討
      - 遅い処理はどこかを考える or 計測する
      - 新たなバグが入らないように注意

テスト

- 複数テストケースに対して実行できるようにプログラムを書くと便利かもしれません（個人差あり）
  - while (~scanf("%d", &N)) { ... }
  - 変数の初期化を忘れないように

暇になった（もう解けない！）と感じたら

- 部分点の取り残しを再確認
  - 他が早く解けた人へのボーナスという意味もあります
  - 残り時間で何点取れるか，を意識して効率よくとりましょう
- 改めて問題文を丁寧に読む
- テストケースをいろいろ作って試す

暇になった（全問満点確定！）と感じたら

- 規則に違反しないように・周囲の迷惑にならないようにリラックスしていてください
  - コードを綺麗にしてみるもよし
  - より速い解法を考えてみるもよし
  - 競技環境に慣れてみるのもよし

おしまい

- IOI 目指してがんばってください