# 模擬練習会 講評/解説

ACM-ICPC OB/OGの会

2005/06/19

# 統計情報

| 問題 | 正解数 | 誤答数 | 難易度 |
|---|---|---|---|
| A: Keitai Message | 40 | 99 | 易 |
| B: Make Purse Light | 20 | 59 | やや易 |
| C: Dragon Fantasy | 4 | 10 | やや難 |
| D: Area Separation | 6 | 28 | やや難 |
| E: Poor Mail Forwarding | 0 | 0 | 難 |
| F: Gather the Maps! | 10 | 31 | 普通 |
| **エラーメール** | 48 | | |

# 全体講評 (1)

- 最上位チーム　5問
  - 解答数，解答時間などジャッジの期待に応えてくれる出来
- 2問以上正解　23チーム
  1問以上正解　43チーム
  - 解ける楽しみを感じてもらえたと思います

# 全体講評 (2)

- 予想とほぼ同じか，それより多く解いてもらえたので良かった
- ミス，失敗が少し目立った
  - 問題はよく読みましょう (特にA問題)
  - Mailerの設定を確認
  - サブミットのミスも多かった
    - 件名の書式が違っている
    - ソースコード，結果の貼り間違い，忘れ

# A: Keitai Message (**概要**)

- **与えられる数字列から，規則に沿って出力される文字列を答える問題**
  - **例**：222030 → cd

- **例年，問題Aに一番簡単な問題が出される**
- **例年に比べると，少し難易度は高い**
- **3分の2の 40チームが正解**

# A: Keitai Message (ミス1)

- 問題は簡単なのだが，ミスが多かった
  - 1行目はデータセットの数
    - 1行目(30)に “d” と答えてはいけない (11チーム，44通)
    - 問題文に，明確に書いてあるのだが
  - 提出の際のミス
    - Subject: の書き方を間違えてしまう
    - ソースを貼り間違い，忘れ

**本番では気をつけましょう**

# A: Keitai Message (ミス2)

- Mailerの設定による誤答も多かった
  - 折り返し文字数が小さくて改行されてしまう
  - Quoted-Printableで送ってしまっている
  - 直にSMTPで送って “.” の処理を忘れている

**練習セッションを利用して確認しましょう**

# B: Make Purse Light (概要)

- ある状況から指定される金額を支払い，残った硬貨の数を最小とする問題
- 無駄に多く払って，釣銭をとってもよい
  - ○ 10円5枚を払い，50円1枚の釣銭
  - × 同じ硬貨が戻ってくるのは不可

- 難易度：やや易
- 正解 20チーム (誤答 59通)

# B: Make Purse Light (解法)

- コインの枚数が少ないので全探索する
  - 全探索する問題は頻出パターン
    - cf. Get a Rectangular Field (2001年国内予選)
  - for文4つくらいは，重ねても良いでしょう
  - コインの戻りは貪欲解法でOK
    - コインの金額が，10，50，100，500なので
- 実は探索せずに解が求まる
  - 全部ぶちまけて，戻ってくる硬貨以外を払う
  - 気付けたなら，いいのでは？(ジャッジの感想)

# B: Make Purse Light (補足)

- 出力の方法が少しだけ複雑
  - 払わない硬貨は出力しない
    - if文を一つ追加すれば良い
  - 連続するテストケースの間を空行で区切る
    - 最初，最後に入れてはならない
    - これを忘れているチームもいくつかあった
- コインの枚数，種類が違ったら？
  - DP (動的計画法) の問題になる

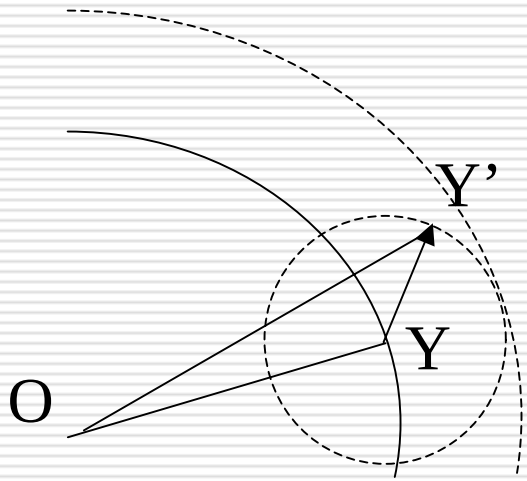
# C: Dragon Fantasy (概要，解法)

- 魔王から逃げながら，クリスタルを集めることができるかどうかを答える問題
- 難易度：やや難
- 正解 4チーム (誤答 10通)

- **枝刈り**をしながら，**深さ優先探索**をする
  - 問題サイズが20なので，順列生成は不可
  - 枝刈りをきちんとしないと，時間がかかりすぎるようなデータが含まれている

# C: Dragon Fantasy (補足)

- クリスタルは直線で取りにいって良い.
  また，判定は，クリスタルを取る時で良い.
  - 一度円周上にくると，離れられない
    → 迂回することはできない
    → 最短距離の直線で結べばOK



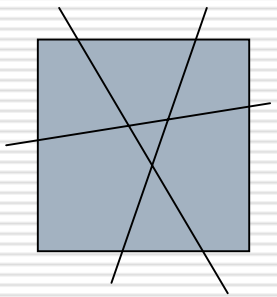

勇者がY→Y'に動くと，
三角不等式より，
OY + YY' ≧ OY'
円の外には出れない

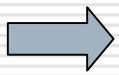
# C: Dragon Fantasy (補足)

- 計算量が階乗となる探索問題の大きさと解法の予測
  - 10!以下：全探索して良い
  - 20!程度：枝刈りをすれば良い
  - 50!以上：通常，探索では解けない
    - DP (動的計画法) で解く方法が多い (e.g. F)
- 重要な探索アルゴリズム
  - 深さ優先探索：解法を1つ求めるのに適する
  - 幅優先探索：最小値などを求めるのに適する

# D: Area Separation (概要)

- 与えられるいくつかの直線によって，
  正方形がいくつに分けられるかを求める



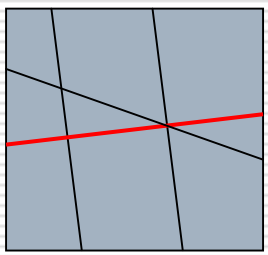


7

- 図形問題は，例年1問程度出される
- 難易度: やや難
- 6チーム正解 (誤答 28通)

# D: Area Separation (解法)

- 直線を引いた時の分け方に注目する
  - 増える領域数 = 交点の数 + 1



赤い線を引くと,
2 (既存の線との交点) + 1 = 3
だけ領域が増える

- これを正しく実装すれば良い
  - 3本以上の直線が1点で交わる場合
  - 交点が, 正方形の周上, 外にある場合
  - 同一の点かの判定 (問題に指示がある)

# D: Area Separation (**補足**)

- **浮動小数点数の扱い**
  - **丸め誤差に注意が必要**
  - **比較を行う場合に，単純に比較してはダメ**
    - if ( x1 == x2 )　**これはよく失敗する**
  - **誤差の分を考慮に入れて実装する**
    - if ( fabs(x1 – x2) < 1e-12 ) **のようにする**
    - **マクロを作っておくと良い**
      - #define EPS 1e-12
      - #define EQ(x, y)　(fabs((x) – (y)) < EPS)

# E: Poor Mail Forwarding (概要)

- ある規則のもとで，郵便物の配送をシミュレートするプログラムを書く
- 難易度: 難
- 正解 0チーム

- 時間内に解くのは大変
  - 上位チームでも，相当の時間が必要
- もっと簡単なシミュレーションは頻出問題

# E: Poor Mail Forwarding (解法)

- 各頂点間の最短距離を求める
- 各頂点間で，次に配送する宛先を求める
- この情報をもとに，条件に従って
  シミュレーションを行う

- ただし，時間1ずつ調べるのでは
  間に合わないので工夫が必要
  - 時刻の最大値が $2^{31}$なので

# F: Gather the Maps! (概要)

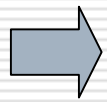
- スケジュールの空いている日に，断片をやりとりしてすべての断片を集めたい．そのような最小日数を求める

- 難易度: 普通
- 正解 10チーム (誤答 31通)
- 多くのチームが，A，Bの後に手をつけた

# F: Gather the Maps! (解法)

- 動的計画法を利用して解く
  - 各人に集められる断片をテーブルで持つ
  - このテーブルを，1日目から順に更新していく
- 例：

| 入力 | 最初 | 1日目 | 2日目 | 3日目 |
|---|---|---|---|---|
| 4 | | | | |
| 2 1 3 | [0] | [0, 1] | [0, 1] | [0, 1, 2, 3] |
| 1 1 | [1] | [0, 1] | [0, 1] | [0, 1] |
| 2 2 3 | [2] | [2] | [2, 3] | [0, 1, 2, 3] |
| 1 2 | [3] | [3] | [2, 3] | [2, 3] |



答：3

- グラフによる解法もある (近日公開予定)

# F: Gather the Maps! (誤答)

- **単純な最小全域木** (Minimum Spanning Tree) **の考え方では間違い.**
  - **このタイプと思われる誤答はかなり多かった**
  - **例**:

**入力**

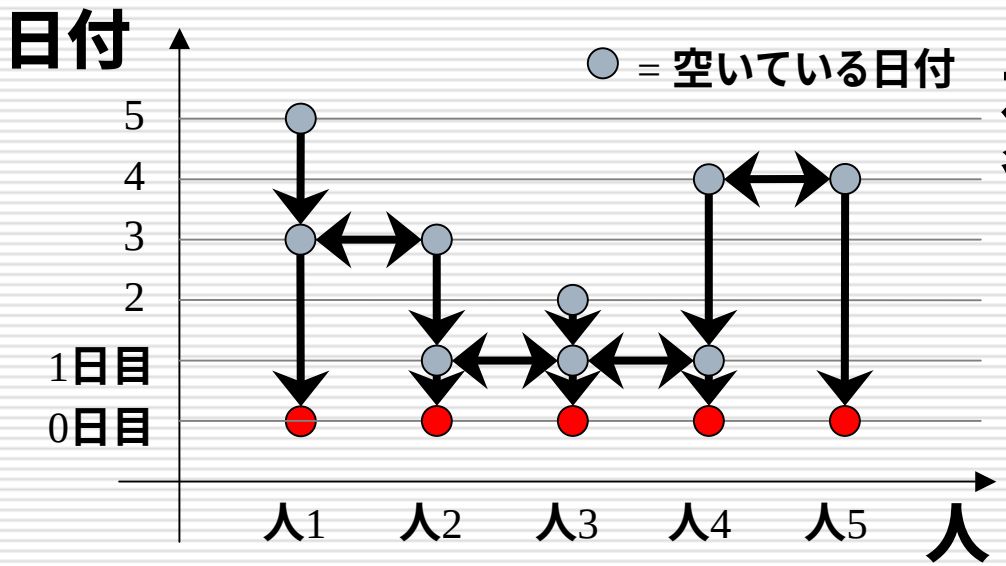
```
4
1 2
2 1 2
2 1 3
1 3
```

**この入力に対して,**
**3日ですべての頂点間が連結されるが,**
**正しくは, 地図を集めることができない**

- **人数が最大50なので, 単純な探索は不可**

# F: Gather the Maps! (解法2)

- "単純な" 最小全域木では誤答だが，
  この考え方の派生でも，解けなくはない




左のような有向グラフ上で，
各点から最下部の赤点(●)
まで全て到達可能か調べる

(Prim 法に近い雰囲気で書ける)

- 一見して無駄な部分をキャッシュするだけで，
  結局は，動的計画解法と同等になるのだが