

tdu競プロ勉強会02

elzup

目的

- ・ Java で解いてみる経験
- ・ アルゴリズムの理解: 深さ優先探索

深さ優先探索

DFS (Depth-First-Search)

- http://www.slideshare.net/chokudai/dfs-49066641?ref=http://atc001.contest.atcoder.jp/tasks/dfs_a

例問題

- ・ 過去問 ICPC 2004年 愛媛大会
- ・ B問題 Red and Black
- ・ <http://judge.u-aizu.ac.jp/onlinejudge/description.jsp?id=1130&lang=jp>

入力値

6 9

. . . . # .

. #

.

.

.

.

.

@ . . .

. # . . # .

h w

mmmmmm

mmmmmm

mmmmmm

mmmmmm

mmmmmm

mmmmmm

mmmmmm

mmmmmm

mmmmmm

出力値

- ・ 45
- ・ (最初の一から到達可能なタイルの数)

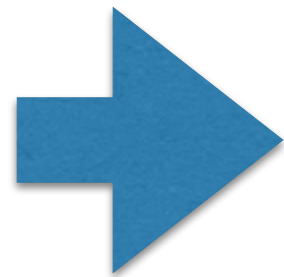
実装方法の検討

- ・ 入力をどう扱うか?(型, クラス, 配列)
- ・ -> 各マスの情報を保持する二次配列が必要
- ・ 探索(網羅)するアルゴリズム -> DFS

情報の保持の仕方

- ・ マスの情報は「未探索か, 探索済みか」が必要
- ・ ‘#’赤いタイルは1, ‘.’黒いタイルを0とする
- ・ ※ ‘@’スタート地点も初期値は0とする, 「スタートはどこか」は別途変数に保存する

情報の保持の仕方

[illegible]

```
map = 000010    sx = 1
      000001    sy = 7
      000000
      000000
      000000
      000000
      000000
      100001
      010010
```

実装してみよう

実装: input 受け取り

- ・ `0 0` を受け取るまでデータセットを受け取るが、
とりあえず1データセットを考えて書く
- ・ `int h, w;`
- ・ `int[][] map;`

実装: input 受け取り

- ・ map に保存する時のロジック
- ・ ‘.’ -> 0 (探索対象)
- ・ ‘#’ -> 1 (赤, 壁, 探索非対称, 探索済み)
- ・ ‘@’ -> sx, sy に座標を保存 -> 0 (探索対象)

実装: 再帰DFS

- ・ あるマスをチェック {
- ・ 塗りつぶされていたら戻る
- ・ あるマスを塗りつぶす
- ・ その4近辺をチェックする (再帰)
- ・ }

実装: カウントアップ, 出力

- ・ 塗りつぶした(探索済にした) 回数を
カウントするだけ

デバッグ

- ・ 二次配列の中身を出力する
- ・ ブレークポイント
- ・ 自作のケース（強いケース）

サンプル シュミレーション & 可視化係

- ・ 同じ問題を2人で解く時にコーディングじゃない人がやる事
- ・ めっちゃデバッグのサポートになる, 複雑な問題で超大事
- ・ 今回で言えば

参考リンク

- ・ ICPC 過去問 in AOJ
- ・ <http://judge.u-aizu.ac.jp/onlinejudge/finder.jsp?volumeNo=11>
- ・ 秋田大 過去問分析
- ・ <http://www23.atwiki.jp/akitaicpc/pages/20.html>