関数・論理型プログラミング実験
最終回

松田 一孝

TA: 武田広太郎 寺尾拓

今日の内容

- 最終課題とその説明

最終課題

- リバーシ（いわゆるオセロ）の思考ルーチンを実装せよ
  - 実装はOCaml, Haskell, Prolog, Curryのどれかを用いること

- また，別紙で説明されるプロトコルにならい対戦可能にせよ
  - こちらが用意したプログラム（とても弱い）と対戦し，平均的に勝てるようにせよ

対戦について

- 審判サーバを利用

サーバ（こちらが用意）

```
$ reversi-serv -p 3000
Waiting 2 connections ...
…
```

クライアント（こちらを実装）

```
$ reversiA -H "localhost" -p 3000 -n Player1
```

```
$ reversiB -H "localhost" -p 3000 -n Player2
```

参考の実装

- 以下がサポートページよりDL可
  - Haskell版（サーバ，クライアント）
  - OCaml版（クライアントのみ）
    - Haskell版のサーバで
      「Haskell版のクライアント vs
      OCaml版のクライアント」も可
    - なので，HaskellやOCamlを使う場合は
      プロトコルの実装の必要はない

注意

- 参考の実装は「とても弱い」
  - 置ける場所にランダムに置く
  - これら強いプログラムを作ることがさしあたっての目標である

最終課題：さらに…

- 工夫点についてまとめよ
- 友人のプログラム，人間（含自分），などと対戦し，その結果についてまとめよ
  - 結果「だけ」を書かないように
  - 勝った場合も負け場合もきちんと考察すること
    - どこが強いところでどこが弱いところか

レギュレーション

- プログラムの実行に必要なファイルのサイズの合計は4MBまでとする
  - コード全体 + 外部ファイル < 4MBかつ
  - 実行ファイル + 外部ファイル < 4MB
- 持ち時間は一分．使いつくしたら負け
  - ちなみに採点はこのMacを利用する予定

レポートの締切

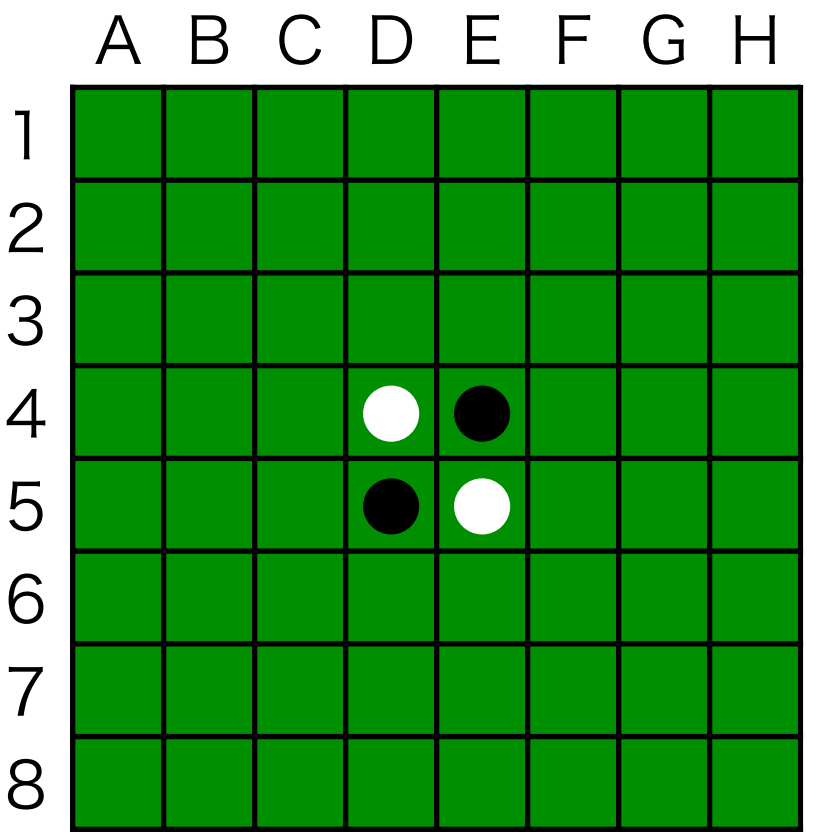
- 最終課題：8/17 24:00 JST（厳守）

解説

- リバーシ
- 課題をすすめるための
  いくつかのテクニックについて

リバーシ

- 日本ではオセロの名で知られる
- ルールは次を参照
  - http://www.othello.org/nakaji/lesson/lesson/rulej.html
  - 初期盤面は右
    黒先手である



リバーシの特長

- 二人零和有限確定完全情報ゲーム
  - 完全情報ゲーム
    - 相手が知る情報は自分も知っている
  - 有限確定
    - 必ず有限手で終わる
- 合法手が少ない
  - cf. 将棋，囲碁
- そもそも可能な局面も少ない

→コンピュータがくそ強い

演習の範囲でもちょっと頑張れば，
すぐに自分では勝てなくなる？

課題を進めるためのヒント

よくあるアプローチ

- 序盤
  - 定石
- 中盤
  - 評価関数を利用した探索
- 終盤
  - 読み切る

終盤：読み切り

- たとえば，50手目の状態で，以降の可能は盤面数は高々10!≒360万程度
  - 実際は合法手が少ないため，これよりはるかに少ない
- なので，終盤は勝ちか負けか現実的な時間で読み切ることができる

中盤：評価関数の利用

- 中盤では現実的な時間で読みきることが難しい
- 探索を打ち切り「よさそうな」盤面に辿りつく手を探す
  - 「よさ」の基準：評価関数
    - 評価関数の設計の指針
      - 相手の可能な手を減らす
      - 隅が取れる
      - …
  - α－β法，ネガマックス等の利用

序盤：定石

- リバーシにも様々な定石が知らている

- 計算機も限られた時間では
  読みがどうしても不正確になる
  → 定石の利用

他の技術

- 盤面の表現
  - 64-bit整数二個で表現化
    - 探索の効率up
      - 64-bitマシンではレジスタ上で計算可
    - メモリ効率を上げることで
      DBとしてもつデータを増やせる
      - 定石や最終盤の盤面など
    - GHCのInt64やocamlのInt64
- 並列化，ループ展開等一般の効率化

効率化→読める深さ向上→強い