第３４回世界コンピュータ将棋選手権

きふわらべ using cshogi
アピール文書
（大会後差替）

２０２４年０５月０９日　高橋智史

ZEROSHO

８戦０勝８敗　４５チーム中、最下位

YOSOKU　MODERU

今回のきふわらべの　予測モデル　の解説でもするか

開発者　高橋　智史

「ユージング・シーショーギって名前に付けておいて
何　全敗で負けてんだぜ　山岡さんに謝れだぜ！」

（←左の画像は著作権上、問題ありません）

コンピュータ将棋エンジン　きふわらべ

「まだ　cshogi のランダム・ムーブで出た方が強かった。
お父んが自力でバグ無く実装できない　予測モデル　を考えたのが敗因」

（←左の画像は著作権上、問題ありません）

ひよ子

「どうやって弱くすることができるか
ひまつぶしに　きふわらべのＰＲ文章を読みたいという需要もあるのよ」

（←左の画像は著作権上、問題ありません）

「予測モデルは　２指し手関係のＫＫ、ＫＰ、ＰＫ、ＰＰだぜ。
学習は　負けたら　評価値テーブルのアクセスしたセルの値を
ランダムに変更して、勝った時に保存だぜ。
それだけのことを　解説していくか～」

ライブラリ使用宣言

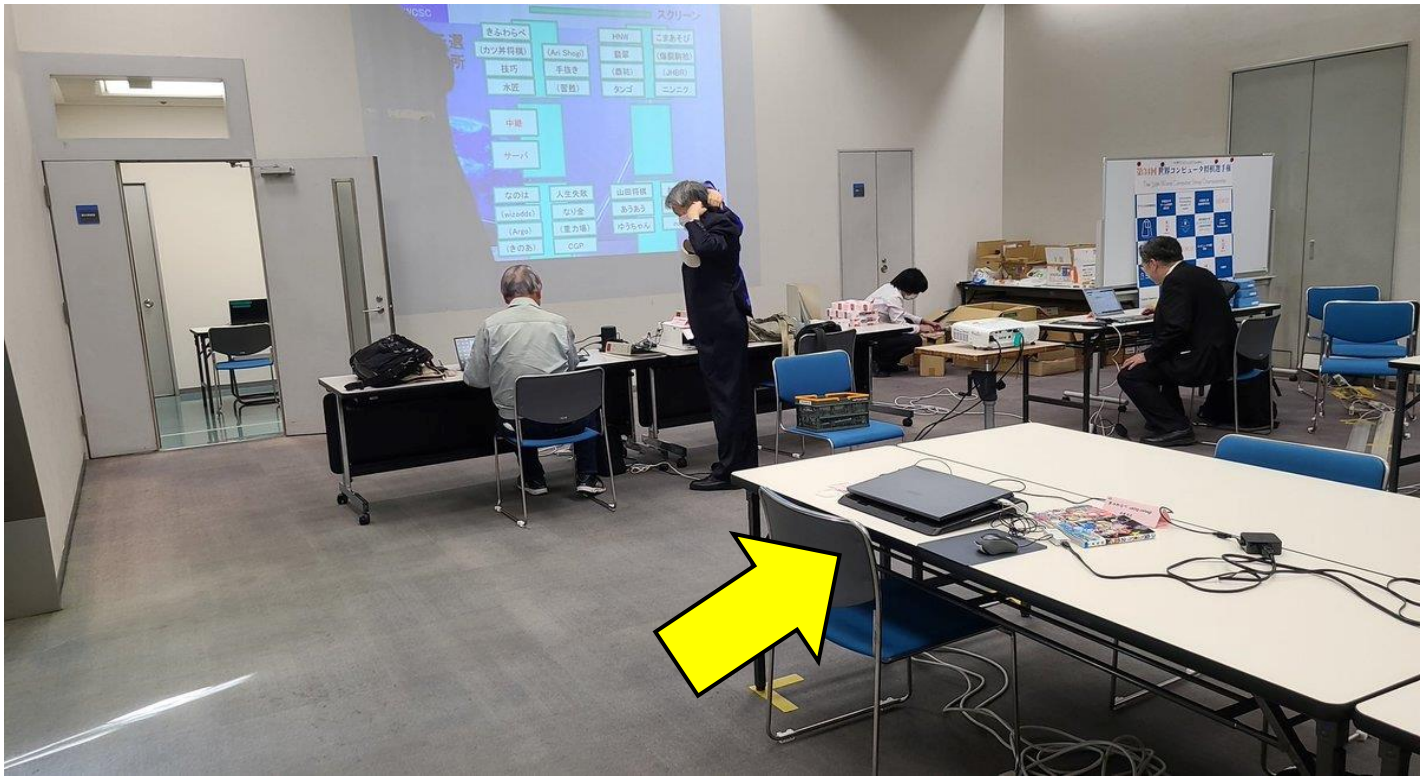
思考部に大きな影響を与える、他者の作成したプログラム・データ等を利用します。

TOUJITU

１次予選　当日　のきふわらべの様子

NO^TO

ノートＰＣ１台で出場



「 ☝ ☞ せっかく
AMD Ryzen Threadripper
7960X 24 Core 買ったのに
結局　リモート接続用の
ノートＰＣで出たのか」

「 Windows は Processor Groups
という制限で１プロセスにつき論理６４コアしか使えないので
これでは将来のスキル的に　スレッドリッパーの全力を出せないので、
ＰＣを１台買い足し、知人に Ubuntu をインストールしてもらって
Linux ディストリビューションの基本操作を覚えることまで　やったが……」

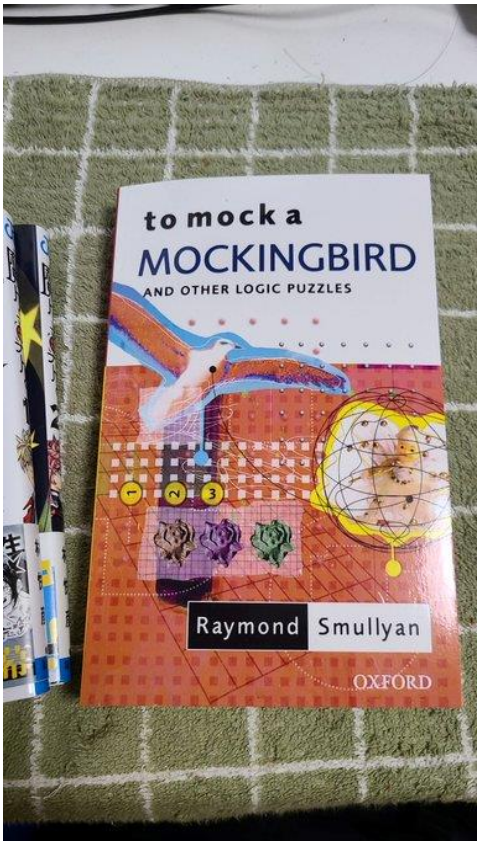
「 そもそも純粋関数型プログラミング言語　Haskell が Linux 前提な
解説ばっかりだったので　Linux　で準備してたのよ」

PC Watch > Zen 4に刷新されたHEDT向けCPU「Ryzen Threadripper 7980X/7970X」はいかなる性能か？
https://pc.watch.impress.co.jp/docs/column/hothot/1549610.html

☝ 左の白いのが買い足したＰＣで、中身は１番右のデカいやつのお下がり。
そのデカいのが Threadripper 7960X ＋ GEFORCE RTX 4090。

「わたしの手先の不器用さでは　静電気でＰＣを破壊してしまいかねないので
組み立ては　昔ゲームサークルで出会った人にやってもらっている。
所得３７０万円程度の４３歳のおっさんにしては　よくやってる方だと思う」

「☞　そして　ＳＫＩコンビネーター計算の
良い教科書のようなものが日本語訳されて
なくて洋書を買って　１０ページぐらい
グーグル翻訳して　読んだ所で挫折した。
もっぱら　その隣に置いてある　漫画本の
魔々勇々（ままゆうゆう）　を読んでた」



「週刊少年ジャンプ編集部にファンレターを
送るほど　ハマってたからな」

「３月３０日ぐらいに
関数型プログラミング言語の習得に
諦めを付けて、
電竜戦のワン・ファイル・マッチ練習会でも
使用経験のあった　Python の cshogi に
切り替えたのよ」

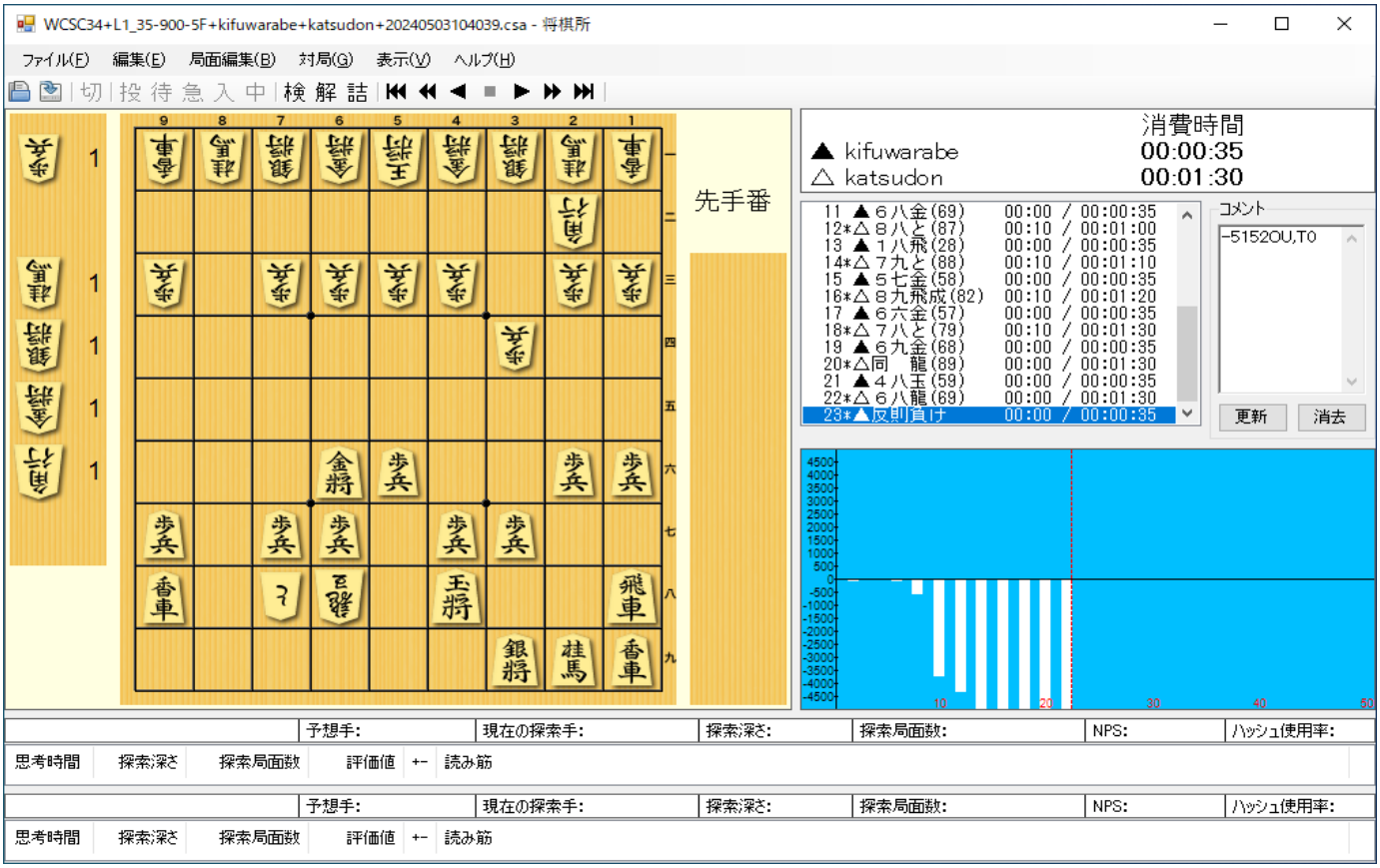
「だから　以下のＰＲ文章の内容は、
たった１か月ちょいの間のできごとだぜ」

NAIYOU

将棋の　内容　は無いよう

KAKUTOU

角頭　を受けずに飛車先を破られて終わり



第３４回世界コンピュータ将棋選手権　一次予選１回戦　きふわらべ using cshogi - カツ丼将棋

「 ☝ cshogi 使ってんのに　なんで反則できるの？」

「 玉が逃げるしかない、というときに　相手の玉を動かしてしまうんだぜ。
なぜ　王手されたときだけなのか？
王手されても　味方の駒がアタッカーを取り返してくれるときは反則しない。
何が起こってるのか　さっぱり分からないぜ」

「 自分で書いたプログラムなのに　なんも分かってないの　わらう」

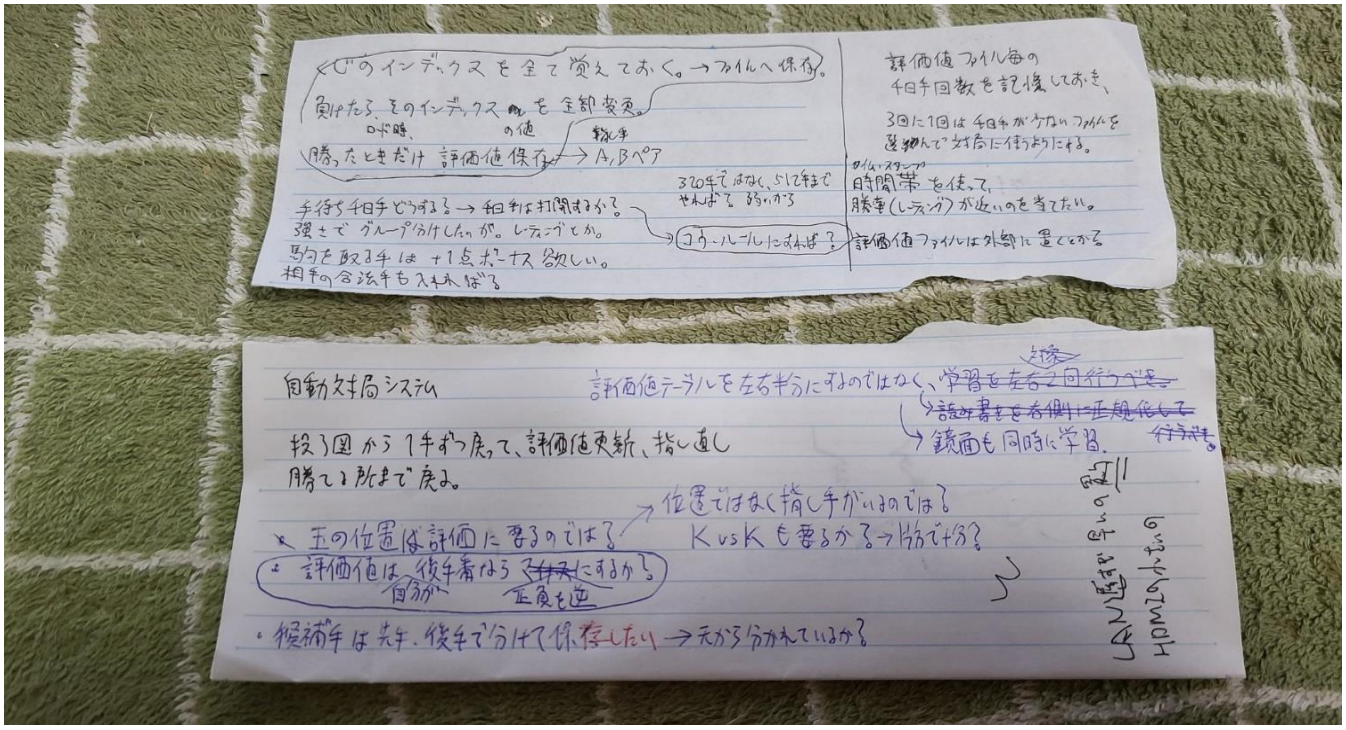
SEKKEISYO

設計書　も　無いよう

OMOITUITA

思いついたことと、やったマークしかない

「 電王トーナメントにも出場してた　mEssiah（メシア）　の開発者の桜丸から
設計図を書いてから実装しろと　よく言われているが、
設計図は書かないのかだぜ？」



「 ☝　ノートの余ったページを千切って　思いついたことをメモって、
実装したら　丸で囲って終わりだぜ」

「 これは　ＴＯＤＯリスト　であって、設計図ではないのでは？」

「 あんたのお父んは　１１歳からパソコンのコマンド打鍵を覚え始めて、
その後３３年間　プログラミングは　＊＊こんなもん＊＊
でしかないのよ。
昔、Ｂｏｎａｎｚａの学習部のソースコードも手書きして覚えてたわよ」

「 お父んがずっと最下位付近にいる実績を振り返って　なんでこんな
貧相な体験で飽きずに　ここまで続いてしまったのか分からないが
ノートの切れ端レベルの思いつきを解説していってくれだぜ」

SASITE

２指し手関係

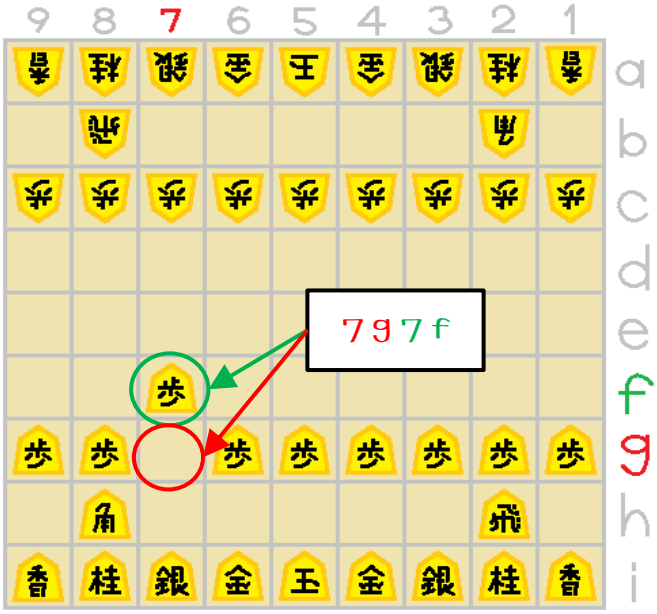
TYAKUSYU　　OUSYU

着手　と　応手　の関係

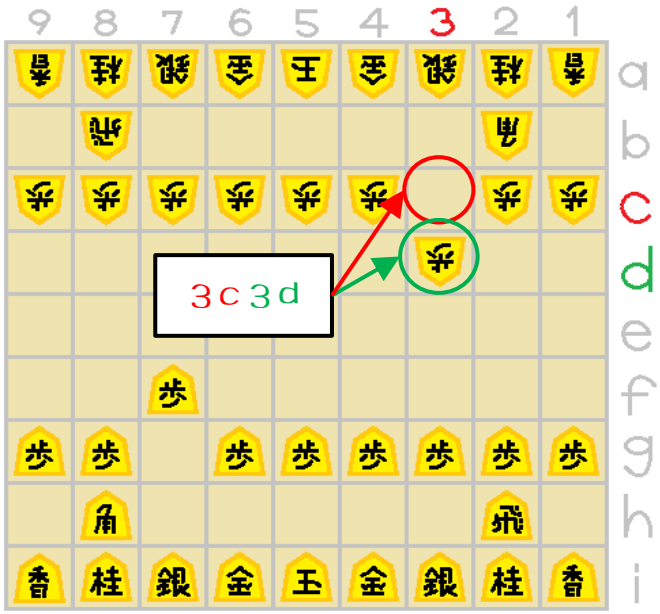
「２０００年１１月に
メイヤー－カレンが公開した
ＵＣＩという規格では

指し手の符号は
どのマスの駒を
どのマスへ移動する、
という感じで書く。

２００７年１月に書かれた草稿
☞　ＵＳＩというこの規格も
それに倣っているぜ」



「フロム　トゥー　よね」



「☜　ＵＳＩの草稿を公開した
トード・ロムスタッドという
ノルウェー人は、

ＵＣＩというチェスの規格を
将棋に持ってくるときに

筋と段が
アルファベット、アラビア数字
の順だったのを
アラビア数字、アルファベット
の順に逆にした。

将棋では
筋は数字の方が通りが良いと
考えたのではないか」

”７ｇ７ｆ＿３ｃ３ｄ＿”

「☝　で、まあ　１０文字あれば　着手と応手は　書けるわけだぜ」

NITI

関係は　無し（〇）　と　有り（１）　の２値

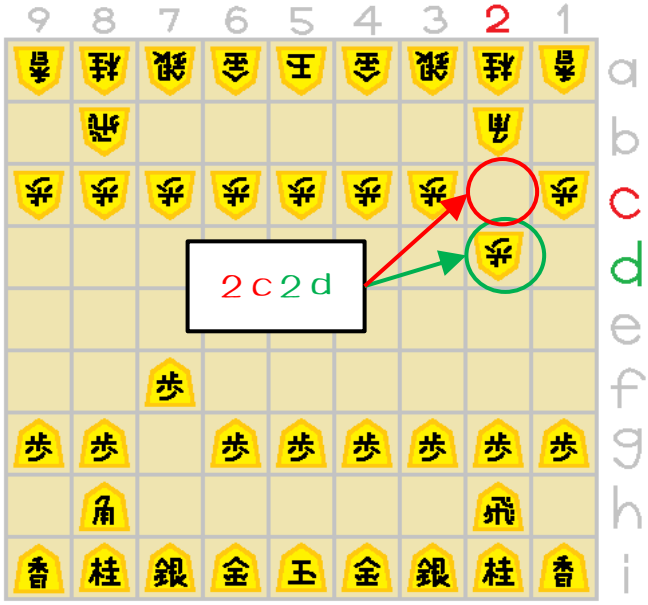
”７９７ｆ＿３ｃ３ｄ＿”　＝　1

「　お互い角道を開けるのは　有るだろうから　１　が入っていてほしいぜ」

「　２ｃ２ｄ　とか
良い手じゃないか？」

「最初、将棋盤を１～５筋まで
半分だけ使って
左右対称という扱いにすれば
メモリ容量が半分で済むと
思ったけど、

飛車先の歩を突くのを覚えたら
角先の歩も突くのよ」

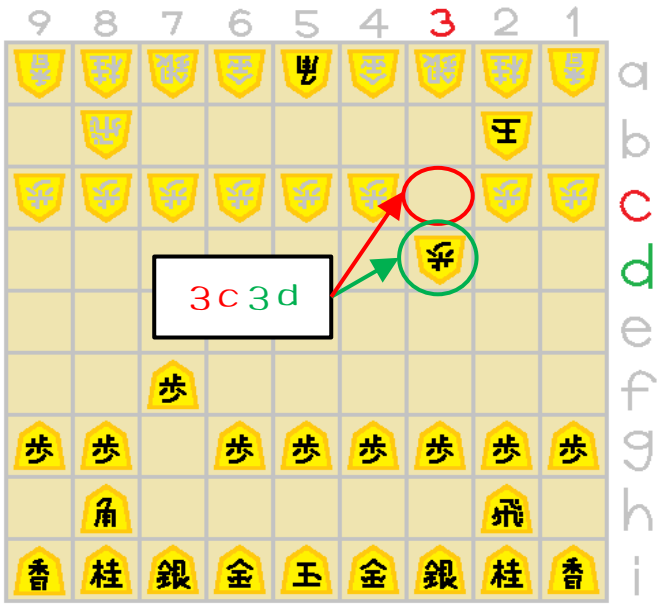


”７９７ｆ＿２ｃ２ｄ＿”　＝　〇

「　角頭歩戦法とか分からんから　無しで　〇　が入っていてほしいぜ」

「じゃあ　着手と応手の　有りそうなペアに　印を付けようとしたんだ」

「しかし　そんなん……」



「 ☜ ２ｂに玉がいるとき
３ｃのコビンは
開けないないだろ？

7g7f_3c3d_ 関係は 0 では？」

「 盤をあんまり見ず、
利きを見て反射神経的に
手拍子でポンポン指してると
思ってくれだぜ」

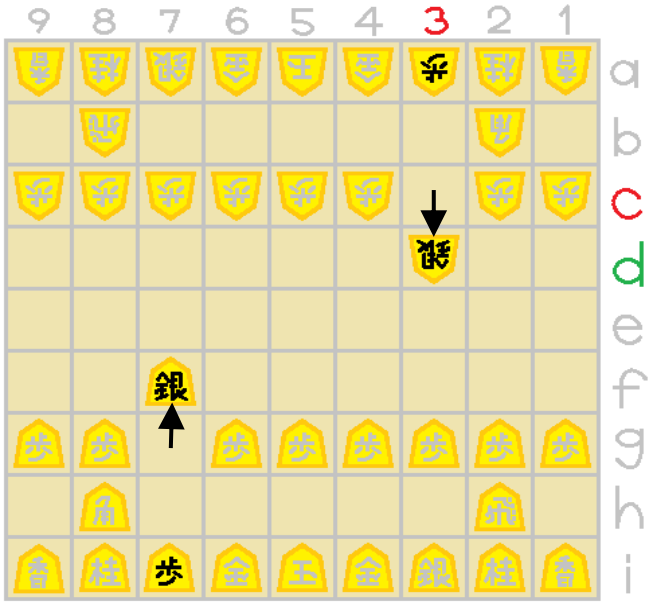
「 そんな欠陥のある予測モデル
作ろうとする人いないから
ちゃんと実装すれば
ダメだったという結果を得て
どうダメだったかという知見を
発表できるのに……」

MINAI
駒を見ない　（例外は後述）

「 ☞ ＵＳＩの規格では
指し手に駒は書いてないから

右図の局面も
7g7f_3c3d_ 関係だぜ」

「 こんな筋の通っていない
予測モデルを作ろうとする
お父んのモチベーションは
どこからくるのか？」



「 浅いアイデアから遠ざけられる挫折の水平線効果なんじゃない？」

KE^KE^ KE^PI^ PI^KE^ PI^PI^

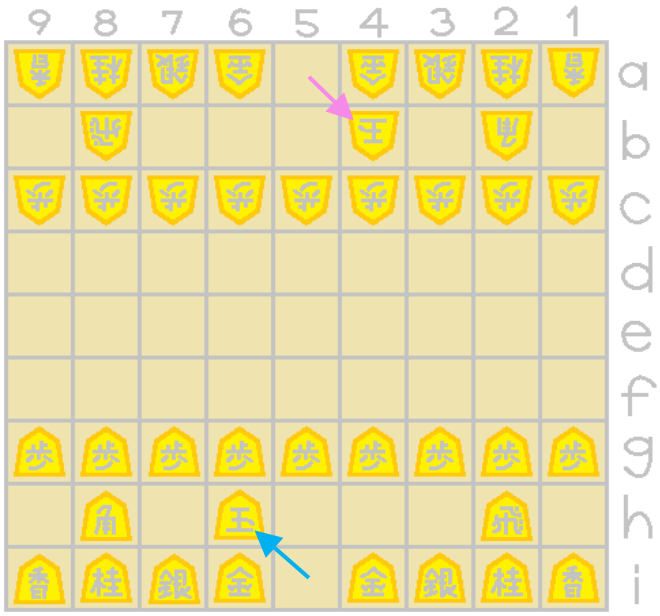
KK、 KP、 PK、 PP

RIKAI SITEINAI

作っている本人が 理解していない 仕様

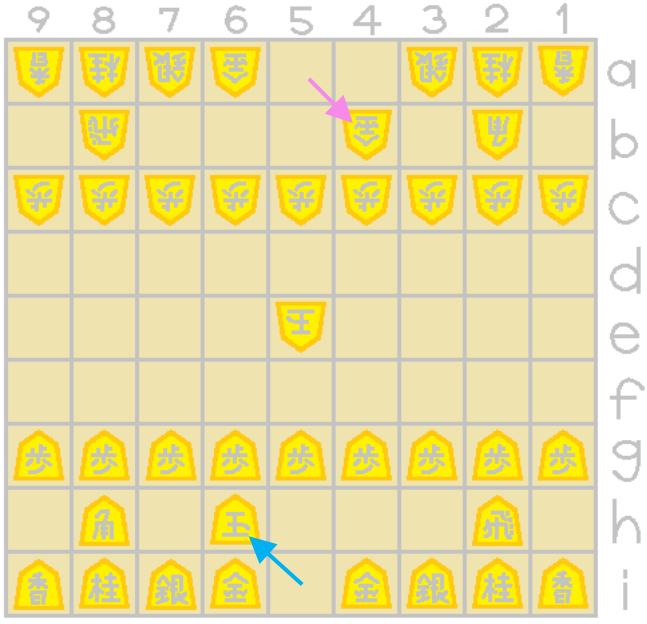
「☞ K（ケー）ってのは
玉のことだぜ。
King（キング）のケー」

「☞ P（ピー）ってのは
玉以外の駒のことだぜ。
Piece（ピース）のピー。
右図の他に、金駒もあるな」



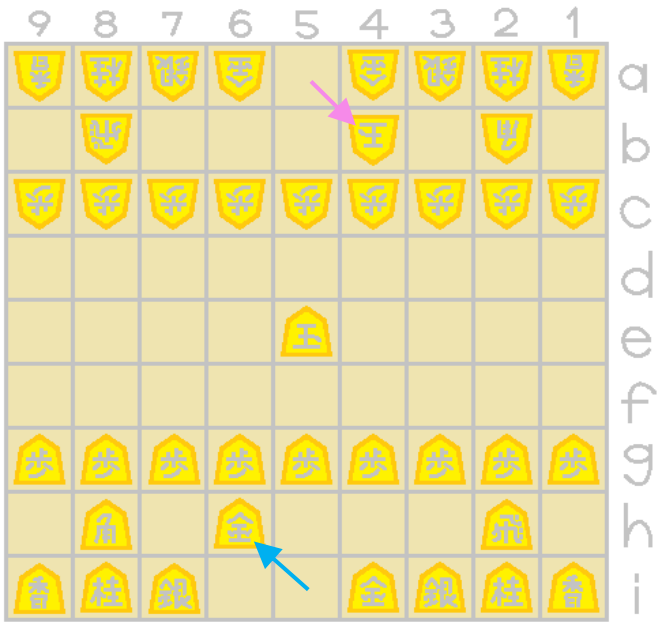
「☜ 左図の 5i6h_5a4b_ は
自玉の指し手と
敵玉の指し手の関係だから
KKと呼ぶぜ」

「 駒じゃなくて矢印を見るのね」

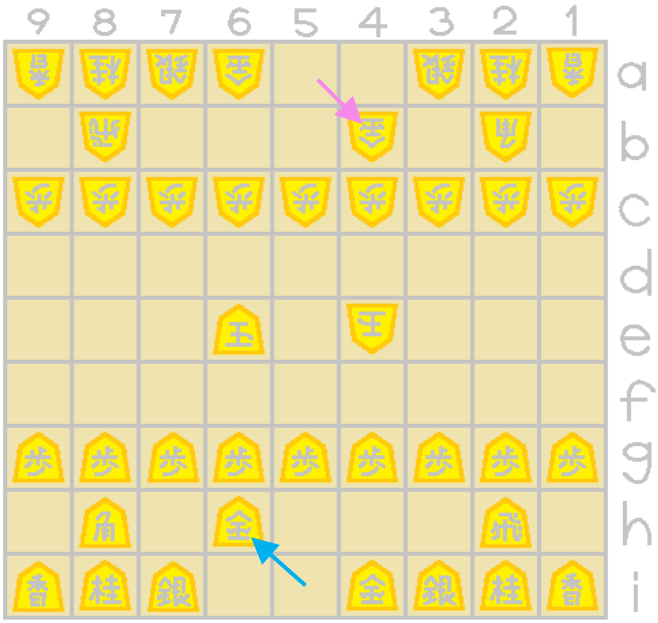


「☞ 右図の 5i6h_5a4b_ は
自玉の指し手と
敵金の指し手の関係だから
KPと呼ぶぜ」

「 符号は同じなのに
動かす駒が違うケースか」



「 ☜　左図の 5i6h_5a4b_ は
自金の指し手と
敵玉の指し手の関係だから
ＰＫと呼ぶぜ」



「 ☞　右図の 5i6h_5a4b_ は
自金の指し手と
敵金の指し手の関係だから
ＰＰと呼ぶぜ」

「 駒は見ないと言っても
玉とそれ以外は区別するのね～」

「 玉は　金にも銀にも利きが似てるからな、銀のふりして上へ突っ込んでいくぜ。
将棋で　玉は特別扱いしないと　ゲームにならないぜ。
玉は捨て駒に使ってはいけないという点で」

「 この図のような　ＫＫ、ＫＰ、ＰＫ、ＰＰ　なんて　実装できてないと思う」

「 わたしは何を実装したんだろな……？」

YON

１つの関係は ４ 種類ある

KK　”５ｉ６ｈ＿５ａ４ｂ＿”　＝　？

KP　”５ｉ６ｈ＿５ａ４ｂ＿”　＝　？

PK　”５ｉ６ｈ＿５ａ４ｂ＿”　＝　？

PP　”５ｉ６ｈ＿５ａ４ｂ＿”　＝　？

「 ☝ じゃあ １つの関係は ４種類あるんだ」

「 玉と それ以外の駒を分けたら それだけ必要だぜ」

KP　”５ｉ６ｈ＿５ａ４ｂ＿”　＝　？

KP　”５ａ４ｂ＿５ｉ６ｈ＿”　＝　－？

「 ☝ ＫＰがあれば 先後反転・正負反転で ＰＫを兼ねるんじゃない？」

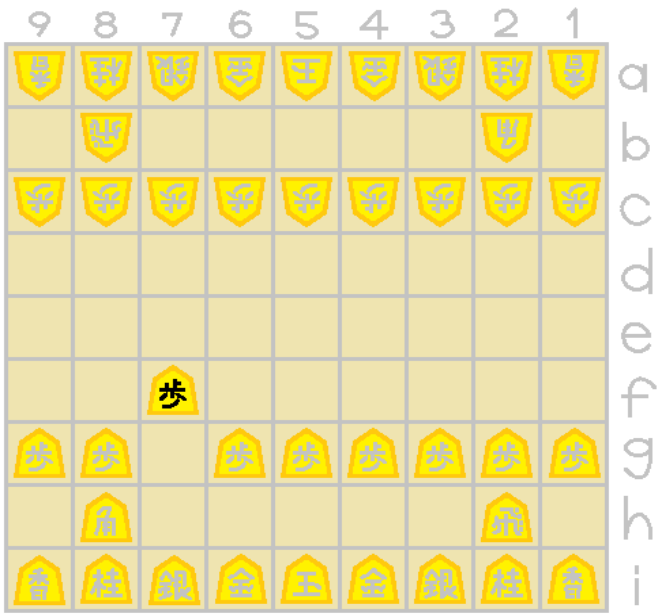
「 そういう工夫をしようとして プロジェクトが崩壊したんで
次期版を作るなら そういう工夫はしないようにするぜ」

HIRATE SYOKIKYOKUMEN

平手　初期局面　での例

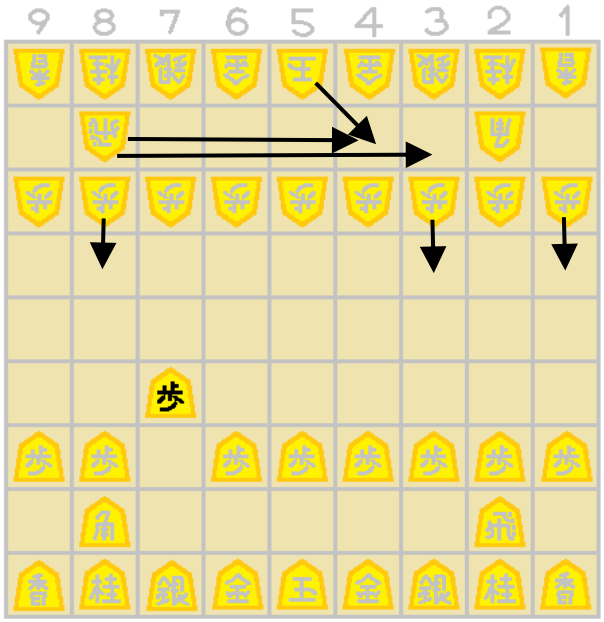
SOWA

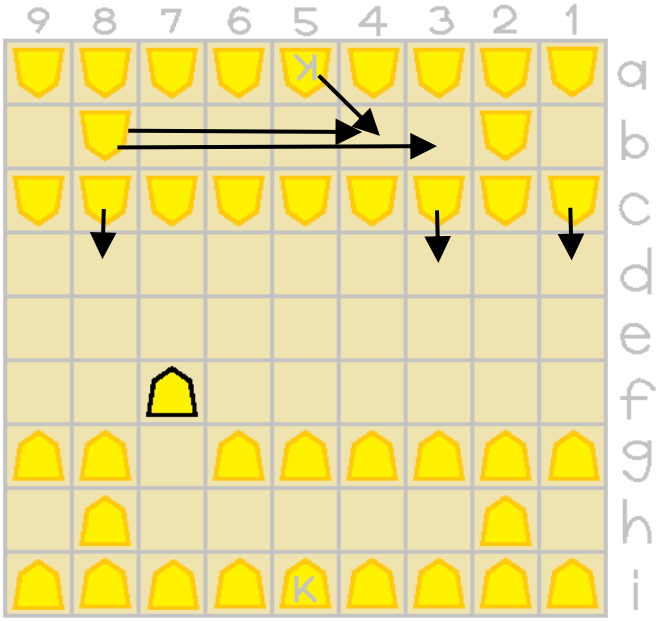
総和するだけ



「☜　じゃあ　７ｇ７ｆ　という
着手は　どういう風に
評価されるのか、
解説していこう」

「☞　相手の合法手が
１票を持ってる選挙だと
思ってくれだぜ。
右図でその挙手数　６　が
７ｇ７ｆの評価値だぜ。
わたしはそれを
ポリシー値と呼んでるぜ」





「☜　上図は正しくないな。

☜　お父んの予測モデルは
ＫとＰしか無いのだから
左図が正しい」

「まったくだぜ」

「 相手をステールメートする手には　選挙権者がいないから、
評価値が入らなくない？」

「 目を瞑るぜ」

「 将棋とは　相手の応手の数が多くなることを目指すゲームか？」

「 ただ１つの受け手しかない場面が連続するような、
直線的な場面とか　山ほどあるんじゃないの？」

「 べつに、相手の応手は　選挙権を持ってるだけで
応手をするかどうかとは　関係ないぜ」

合法手の中で 最大（SAIDAI） のポリシー値のある手の中から選ぶ

「 全ての指し手に　ポリシー値が付く仕組みは説明したから、
合法手の中で　最大のポリシー値を持つ手を選んで、
その中から　ランダムに１手　選べばいいわけだな。

わたしはこれを　I'm feeling luckey と呼んでるぜ」

「 グーグルのボタンの名前じゃない」

HOZON

KK、KP、PP評価値ファイルを 保存 しようぜ？

YOMIKOMI

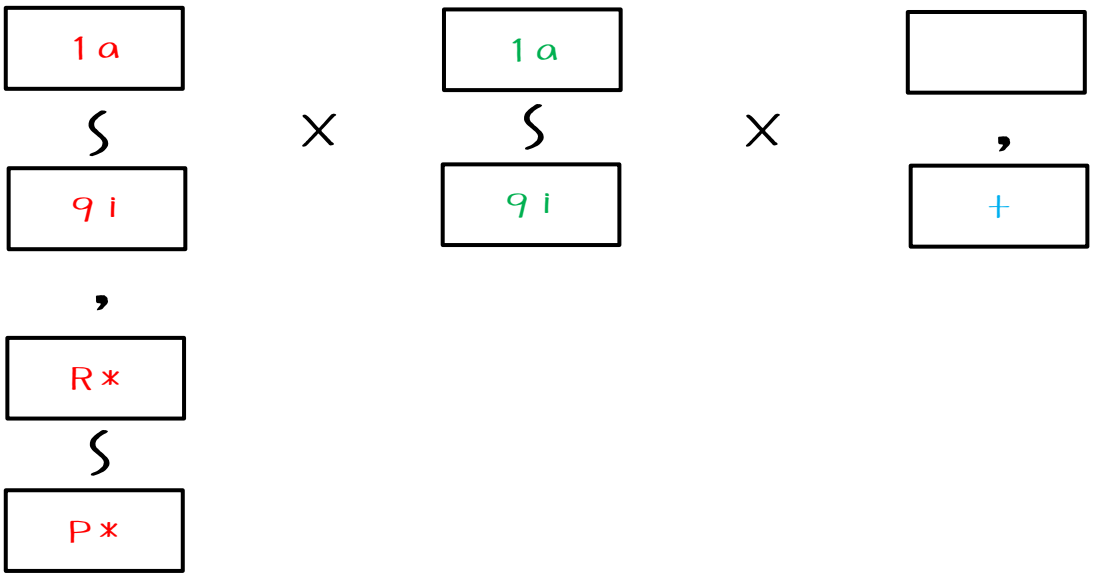
ファイルサイズがでかいと　読込　に時間かかるしな

「 今度　実装するときは　Python　なんかで評価値ファイル読込すんの止めとこ」

「 Rust を本体にして、合法手生成だけ　Python　にすればいいんじゃない？」

PIˆ

指し手Pのサイズ

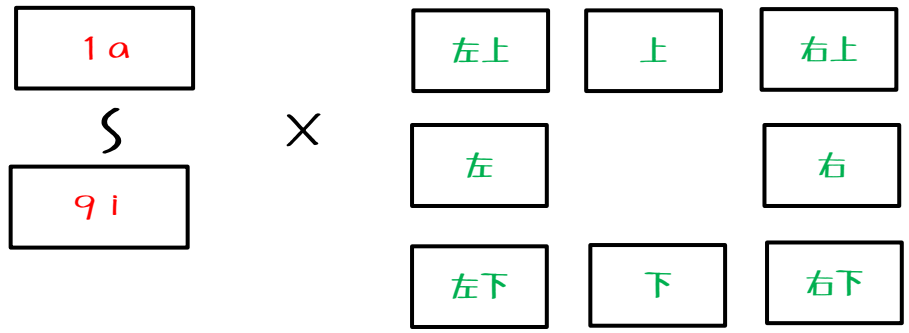


「 USIの指し手の符号って、
FROM　には８１マスと７種類の持ち駒、
TO　には８１マス、
PROMOTION　には有無の二値の、
88×81×2＝14，256　パターンより
少ない数しかないのな」

「 1aから3bへワープする駒は無いし、全結合は無駄が多そうねえ」

KE^

指し手Kのサイズ



「玉は　打と成りは無く、　移動先は８方向しかないから　さらに少なく
８１×８＝６４８　パターンより少ない」

「盤の隅なら　移動先はもっと少なくなるな」

SASITE

２つの指し手のテーブルのサイズ

「ＫＫテーブルのサイズは
６４８×　　６４８＝　　４１９，９０４ｂｉｔ

ＫＰテーブルのサイズは
６４８×１４，２５６＝　９，２３７，８８８　ｂｉｔ

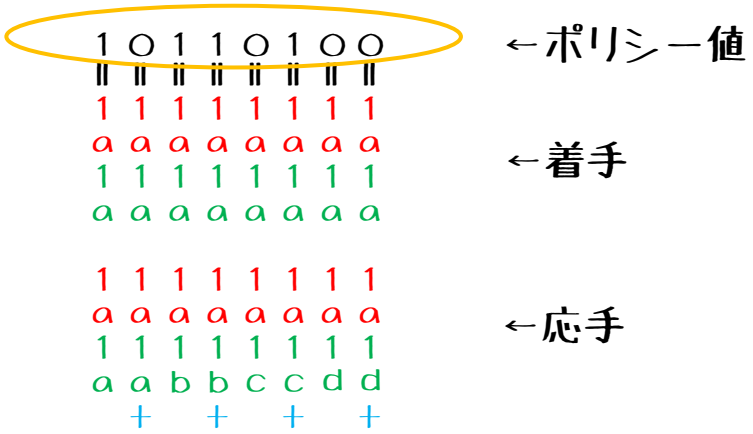
ＰＰテーブルのサイズは
１４，２５６×１４，２５６＝２０３，２３３，５３６　ｂｉｔ

だぜ」

「だいたい合計　２５．３メガバイトか」

「家の真ん中のｉ７のＰＣで読み込むのに５０秒ぐらいかかっちゃって
将棋所での自動の連続対局が　時間かかっちゃうのよね～」

BIGGU　　ENDHIAN
ビッグ　エンディアン



「 ☝ 例えばＰＰテーブルの最初の１バイトは
左から　上記のようにビットが入っていると思うぜ」

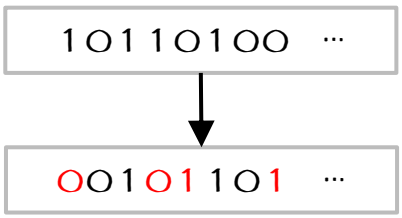
「 お父ん　実装を　ペッと書いたら
データの中身　確認してないもんな、
そんな風に入ってないかも知れないぜ？」

「 まあ　これで、次の一手を選ぶ仕組みの
小細工を除く　主要な部分は　説明したぜ」

GAKUSYU
学習

KOSHIN
評価値ファイルが良い感じに 更新 されるのが学習

「 学習（ラーニング）が何かというと」



「 ☝ なんか知らんけど　評価値ファイルの内容が
いい感じに更新されることだぜ」

「 良くならないんだったら　荒らしてるのとおんなじだからな」

「 じゃあ　きふわらべちゃんの評価値は　ぜんぜん良くなってないから
学習　って呼んだらダメなんじゃないの？」

「 学習ではないが　学習のふりをしている行為って　日本語で
なんて名前で呼べばいいんだぜ？」

「 お父んが学生時代に何と呼ばれていたか　思い出せだぜ」

「 落書き……？」

RAKUGAKI

落書き

ARASARERU

評価値ファイルが 荒らされる のが落書き

「 落書き（グラフィティ）が何かというと」

10110100 …

↓

00101101 …

「 ☝ なんか知らんけど　評価値ファイルの内容が
変わってることだぜ」

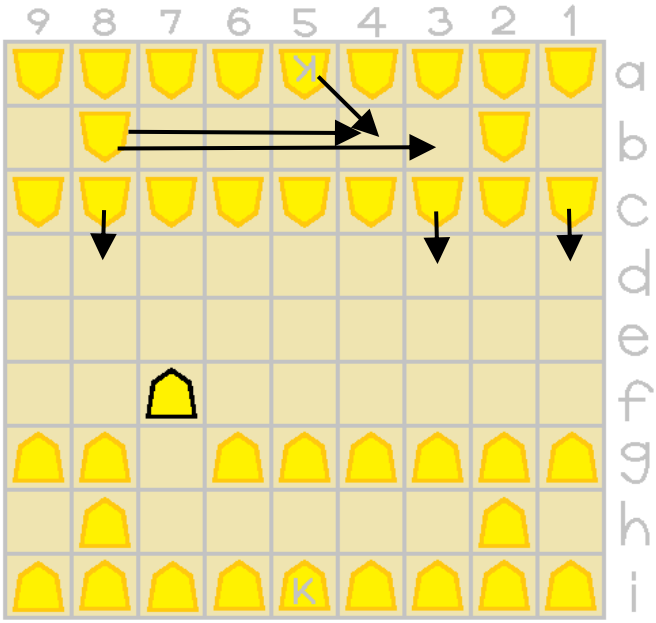
「 それなら実装されてるぜ」

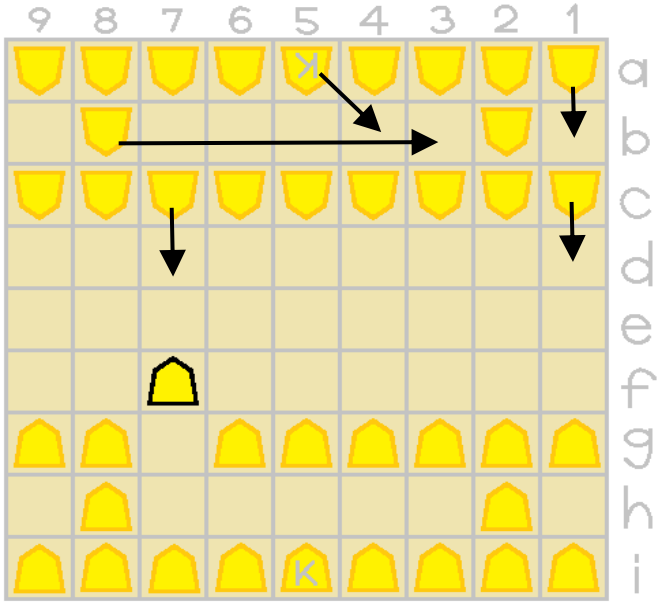
「 正しい説明ね」

KAWARU

指し手が変わるように落書きしようぜ？

「 ☞　得票してくれるから
その手を指したわけで」





「 ☜　選挙権者の顔を全部
覚えておいて、
票を入れてくれたやつ、
票を入れてくれなかったやつ、
乱数でゼロいちを上書きだぜ」

「 評価値テーブルから引用した指し手を全部覚えておいても
一局、千数百ぐらいだぜ。多くは重複なのだろう」

「 負け続けている間　選挙権者の顔を追加で覚え続けて、
勝ったら評価値テーブルを保存、顔は全部忘れるぜ」

「 近い強さの評価値テーブルが自己対戦したら、確率的にどっちでも負けるから
いつまでも評価値テーブルを　落書きしまくるんじゃない？」

「 プログラムが　全体的にバグだらけで
レーティングの導入など様々な工夫には　頭が回らないぜ」

RENZOKU TAIKYOKU

将棋所の　連続 対局　で落書き

「 落書きのためのフレームワークなんて　なんも無いんで、
きふわらべの　Python　ソースコードを丸ごとコピーして
プレイヤー１、プレイヤー２　という扱いにして、

将棋所を使って
持ち時間４０分、　手数の上限２０００　で連続対局だぜ」

「 手の偏った　ランダムムーブ　みたいなもんで　ほとんど千日手だが、
４００手ぐらいあれば　まぐれで詰むだろうと増やして、
多めに２０００、１手１秒消費してしまうので　２０００秒は３３分ちょい、
ざっくり４０分だぜ」

ROKU

評価値テーブル ６ 人前

「 プレイヤー１、プレイヤー２　に評価値テーブルを６つずつ、
連続対局中は　計１２個の評価値テーブルを持たせて
ランダムでどれ使うか選ばせているぜ」

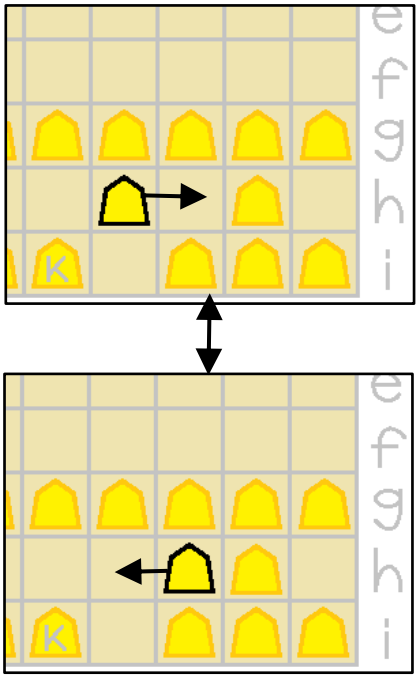
「 千日手になるもの同士の　評価値テーブルってあるし、
ばらつかせないとな」

「 学習が１２倍　遅くならない？」

「 遅いが、千日手では　落書きが進まないんで……。
６は　最初　サイコロの目ぐらいの意味だったが、
それより少ないと　千日手が多くて　勝ち負けが付くのが遅いんで
６つ持たせると　勝ち負けが付くとこまで進ませるのに　いい感じ」

KO
コウの更新

「 ☞　金がよくやるんだが
左右にパタパタ手待ち
お互いにやって　千日手」

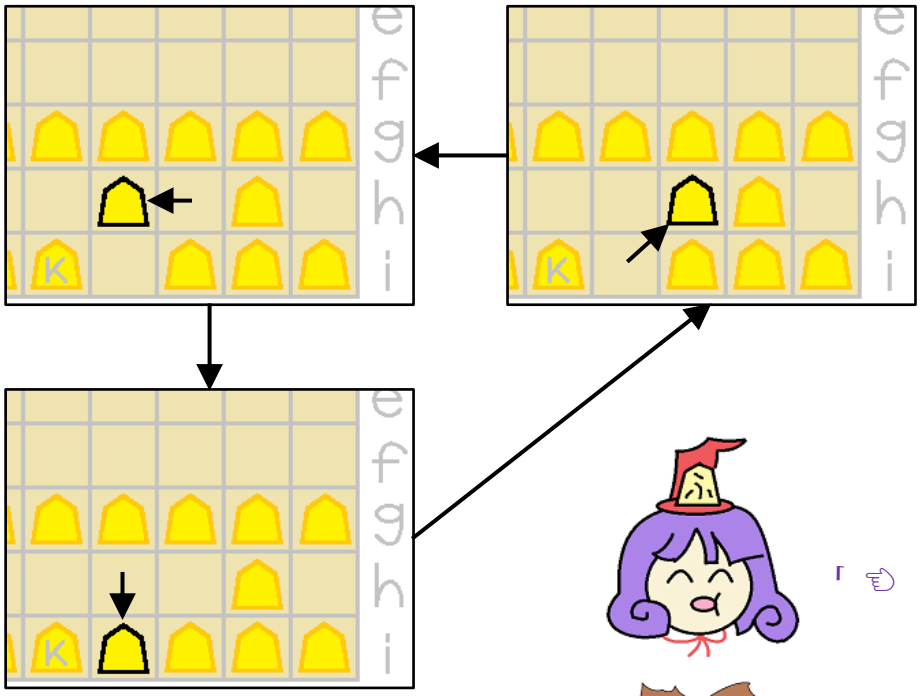


「 歩や香には真似できませんからね」

「 そこで、自分が２つ前に指した手は
指さない、とプログラムしてるぜ。

これは囲碁のコウと同じ実装だぜ。

ただし、将棋向けに　他に手がなければ
指すようにアレンジしてるぜ」



「 ☜　三角形でも回せるぜ」

「 ファインプレーにもなる手なのか
明らかにダメな手なのか
将棋アマ５級のわたしには
分からないんで
どうしたろうか」

「 大まかな　推測モデル　と　落書き　の話しは　終わりだぜ」

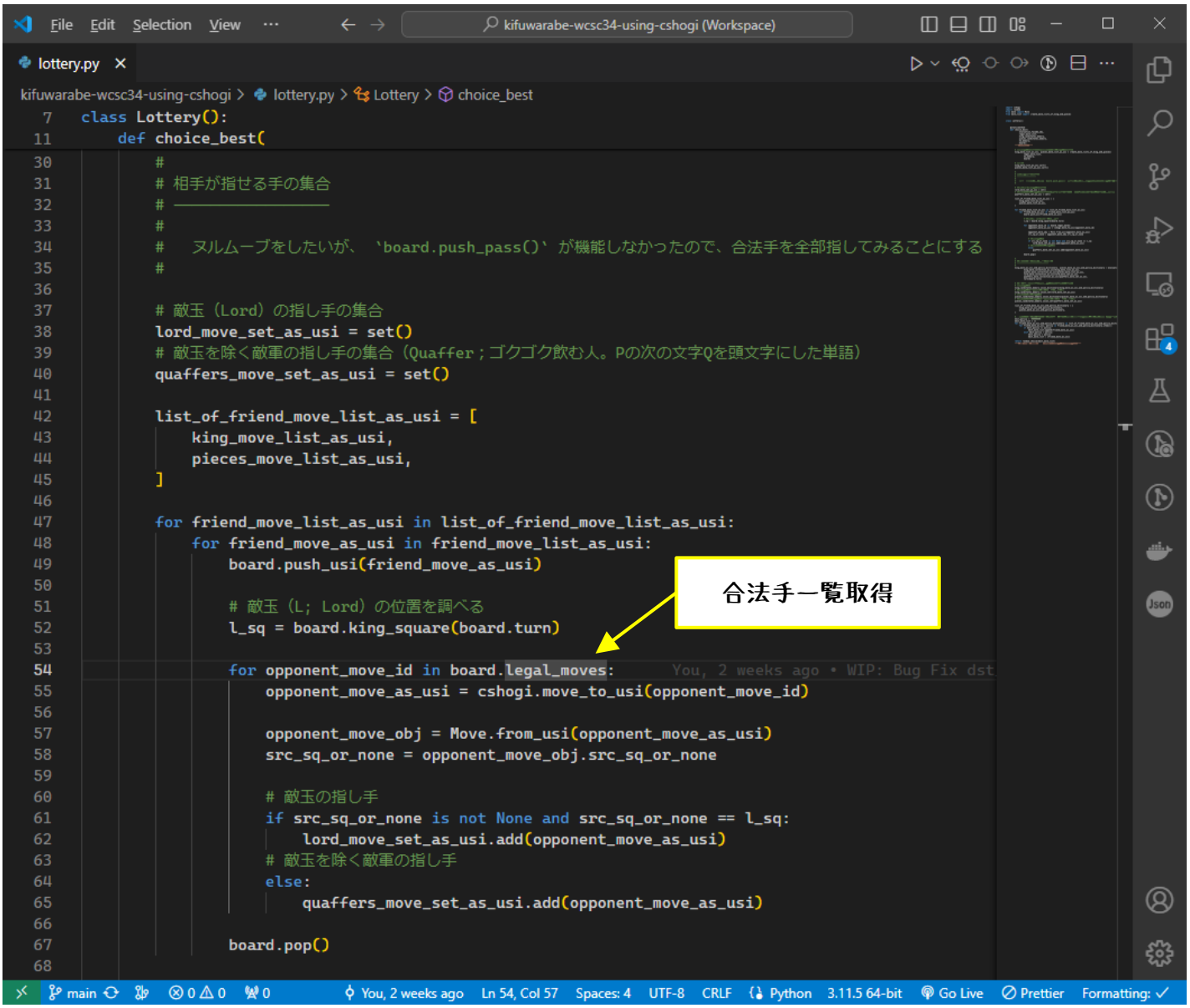
U^ZINGU　SI^SHOGI

ユージング・シーショーギ

SASITE　SEISEI

指し手　生成　を書かなくて済むから楽だ～

「せっかく　cshogi　使ってるんだから　スクリーンショットぐらい
貼ってみるかだぜ」




```
class Lottery():
    def choice_best(
        #
        # 相手が指せる手の集合
        # ――――――――――――
        #
        #   ヌルムーブをしたいが、 `board.push_pass()` が機能しなかったので、合法手を全部指してみることにする
        #

        # 敵玉（Lord）の指し手の集合
        lord_move_set_as_usi = set()
        # 敵玉を除く敵軍の指し手の集合（Quaffer；ゴクゴク飲む人。Pの次の文字Qを頭文字にした単語）
        quaffers_move_set_as_usi = set()

        list_of_friend_move_list_as_usi = [
            king_move_list_as_usi,
            pieces_move_list_as_usi,
        ]

        for friend_move_list_as_usi in list_of_friend_move_list_as_usi:
            for friend_move_as_usi in friend_move_list_as_usi:
                board.push_usi(friend_move_as_usi)

                # 敵玉（L; Lord）の位置を調べる
                l_sq = board.king_square(board.turn)

                for opponent_move_id in board.legal_moves:      You, 2 weeks ago • WIP: Bug Fix dst
                    opponent_move_as_usi = cshogi.move_to_usi(opponent_move_id)

                    opponent_move_obj = Move.from_usi(opponent_move_as_usi)
                    src_sq_or_none = opponent_move_obj.src_sq_or_none

                    # 敵玉の指し手
                    if src_sq_or_none is not None and src_sq_or_none == l_sq:
                        lord_move_set_as_usi.add(opponent_move_as_usi)
                    # 敵玉を除く敵軍の指し手
                    else:
                        quaffers_move_set_as_usi.add(opponent_move_as_usi)

                board.pop()
```

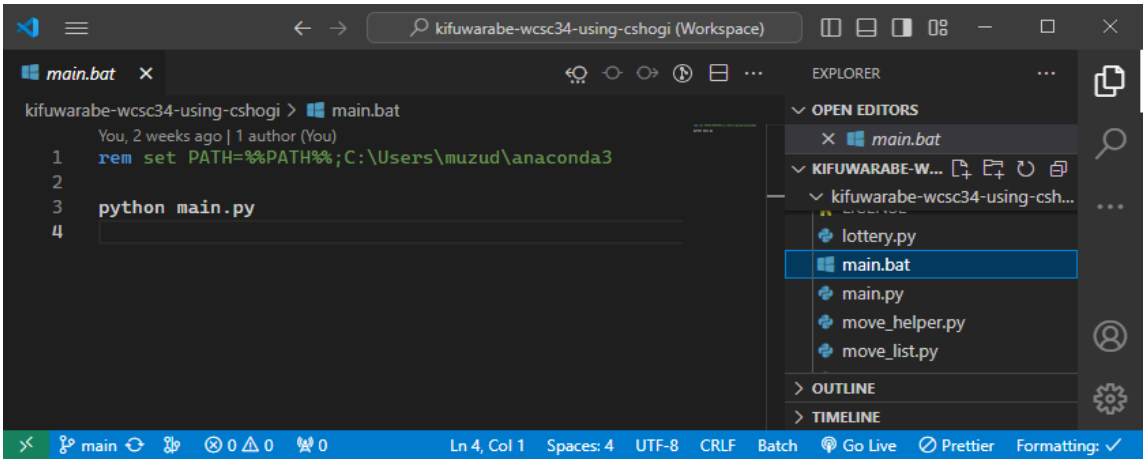
「自玉が　Ｋｉｎｇ、　自軍のその他の駒が　Ｐｉｅｃｅ　なのに対し、
敵玉が　Ｌｏｒｄ、　敵軍のその他の駒が　Ｑｕａｆｆｅｒ　とか
このＰＲ文書で触れていない　勝手な造語など　いろいろあって
お父んのソースコードを読む気には　ならないと思う」

KITTINGU

キッティングのようなこと

「プログラミングより　キッティング（※）の方が
難しいんじゃないかと思うんだが」

※さあ始めるぞ、というところまで必要な物一式全て用意すること



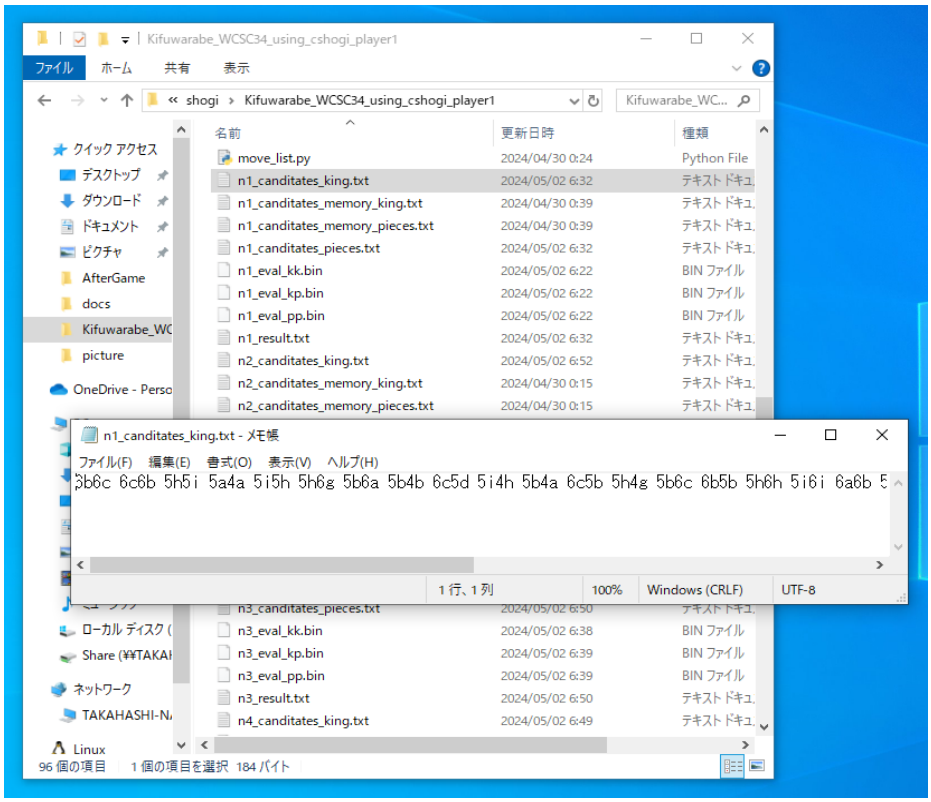
「　Ｐｙｔｈｏｎ　のセットアップも　ｃｓｈｏｇｉ　のインストールも
土曜日１日使い潰して　失敗して　結局たどり着いたのが

Ｐｙｔｈｏｎは公式からインストールせず、
ａｎａｃｏｎｄａ　をインストールして
バッチ・ファイルで　環境変数を設定して　ａｎａｃｏｎｄａにパスを通し、
将棋所からその　バッチ・ファイル　を呼ぶことだぜ」

「２５年ぐらい前のコマンド操作の感覚ねえ」

TIGA

cshogiと　USIの同じようで 違 うところ



「 ☝ “落書き” に使う指し手は　USIの符号で保存している。

cshogi は move（指し手）を整数で返してくれるから、
それをバイナリで保存した方がメモリが小さくて済むじゃないか、
と最初は思ったんだが

cshogi の move は現局面の board 変数に依存した整数になっているようで、
ポータブルなものではないようなので、
メモリでも外部記憶でも、USIの符号と区切りの半角空白という
およそ6バイトの可変長文字列を使ってるぜ」

「 USIの符号を　整数に変換すれば　前述の
Pなら　0～14，255
Kなら　0～　　657
の変域で済むから　固定長2バイト型整数で充分な気もするけど、

Pはもっと小さな変域で済むのでは？　という考えも気になって
手が回ってないのよね」

FUGUAI

知られていない 不具合

HANIGAI

評価値テーブルの 範囲外 へのアクセス

「ＫＫ、ＫＰ、ＰＰテーブルが配列で用意されているみたいだが、
配列の範囲外にアクセスして例外投げていて、
ポリシー値は　０　として数えているところが大量にある。
なんで　範囲外にアクセスするんだぜ？」

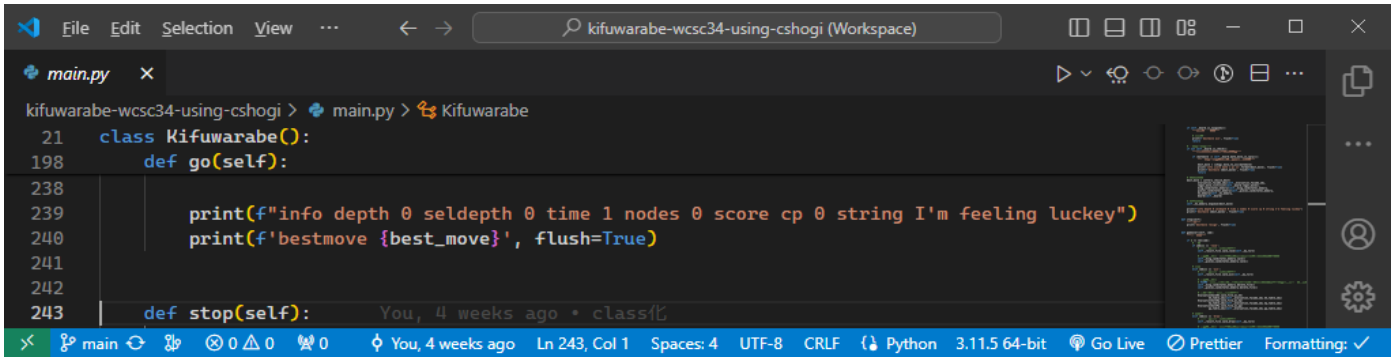
「さっぱりワカラン　わたしは２、３時間ぐらいで
アイデアを　ちゃちゃっと実装して
三連休は数十時間　壊れたプログラムを眺めて　休日も終わりだぜ」

IRANAI

盤の表示は要らない～

KUJI

くじ を引くだけなのだから



「　読み筋は I'm feeling luckey しか言わないわよ」

「アルファベータ探索もやってないしな。
お父んは　そういう将棋エンジン作りたがってたから
最初の１歩に着手できたな」

「 cshogi を使うようになって、選手権に出る前に　いつも作ってた
将棋盤の表示のようなものも　要らなくなって　楽だぜ～」

「 次の一手を引くだけの　おみくじに　盤なんて　無いわよ」

「 稲庭将棋の将棋盤が映っていないディスプレイみたいな
かっこいいやつ　作れないかな？」

「 画面に　くじ　だけ表示したらいいんじゃないの？」

「 よーし、わたしがこれから　やるべきことが　はっきりしてきたな。
少しでも良い　くじ　が引けるように、
シャッフル・アルゴリズム　を作ることだぜ」

「 全自動麻雀卓みたいだな」

「 WCSC34に初参加してた将棋エンジン　はるか　の小林さんが会場で
きふわらべの長いPR文章が読みたい、と言ってくださったので
書いたぜ！
よっしゃ終わり！　ドロップボックスに投げ込んで　寝よ！」

>>> Dad, stir up the lottery well