**2018-6-29 4I実験AIテーマ設計書 v.2.１**

1. **BasicAI**

AIplayerを表すクラス．IAiの派生クラスとして定義する．

* 1. **プロパティ （プロパティの宣言時に値の初期化を行うこと．）**

|  |  |  |
| --- | --- | --- |
| 型 | 名前 | 説明 |
| string | name | AIplayer (戦略) を区別するラベル(各自が作成するAIの名前をつける) |
| public int | depth | 先読みの深さ(d)を表す. デフォルトの初期値は1 |
| public bool | isInitialized | AIplayer初期化の有無を表す．初期値はfalse, 初期化が済めばtrue． |
| public List<State> | closedList | 1手目以降の着手済状態のリスト．ゲームの棋譜を表す．  初期値は new List<State>(); で設定．  Stateオブジェクト(の参照)を要素とする**List(動的一次元配列)**． |
| public int[,] | weightTable | 各マス目の重みを定義する重みテーブル(size x size の2次元配列)  ボードの縦横のサイズ(Boardクラスのsize プロパティ)で配列を作成 |

* 1. **メソッドその1 (基本的機能)**

**public BasicAI()**デフォールトコンストラクタ．各プロパティの初期値を**7.1**の表に従って設定する．

　　　　　　　　　　weightTable をinitWeight(1) で初期化する．

**string getName()**AIの名前を（string型で）返す．

**private void initWeight(int weight)**  weightTable 配列の全ての値をweightで初期化する．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| int | weight | weightTable 配列の値の初期値 |

**public int getWeight(int row, int column)**　 weightTable の(row, column)マス目の値を返す．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| int | row | weightTable上の位置（行，上下）を表す．一番上が0，一番下が7． |
| int | column | weightTable上の位置（列，左右）を表す．一番左が0，一番右が7． |
| int | 戻り値 | weightTable の(row, column)マス目の値を返す． |

**public Place nextHand(Board board, Place previousMove, bool isBlack)**

ボード情報を元に指定の色(**isBlack**)の手番が次に置く場所を返す．AIplayer (戦略)呼び出しの共通の窓口．AIテーマでは，このメソッドから呼び出すAIplayer (戦略)のメソッドを書き換えていく方法で実験を行う．参照先のboardに対し，指定の色(**isBlack**)の着手可能セルそれぞれの評価値を更新し，指定の色(**isBlack**)の手番の着手した盤面において次の手番の着手可能セルそれぞれの評価値を更新する．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| Board | board | ボードオブジェクト(の参照)を渡す． |
| Place | previousMove | 黒(player)の(直前の)着手セルの位置をAIに知らせる． |
| bool | isBlack | 次の手番を表す．黒手番ならtrue，白ならfalse． |
| Place | 戻り値 | ボード上の着手した位置\*1を表す．パスならnull を返す．**副作用で**参照先のboardの，指定の色(**isBlack**)の着手可能セルの評価値を更新する． |

テスト用の本体定義の例・・・return new Place(2,4); // new演算子で生成した白着手\*1を返す

**public Place simpleNextHand(Board board, Place previousMove, bool isBlack)**

ボード情報を元に，指定の色(**isBlack**)の手番が次に置く場所を左上から右に探して，おけるところに置く．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| Board | board | ボードオブジェクト(の参照)を渡す． |
| Place | previousMove | 黒(player)の(直前の)着手セルの位置をAIに知らせる． |
| bool | isBlack | 次の手番を表す．黒手番ならtrue，白ならfalse． |
| Place | 戻り値 | ボード上の着手した位置を表す．左上から右に探して\*2，最初に見つかった着手可能セル\*3の場所．パス(着手可能な場所なし)ならnull を返す． |

このメソッドを実装して**nextHand/3**から(**simplenextHand/3**を)呼び出すと，弱いAIと対戦できる．

ヒント：\*2 for文の2重ループを作る． \*3 GUI設計書p.3, **canPlace/3**

* 1. **boardの参照へのセルのプロパティ(色別評価値の)更新テスト用のメソッド**

(GUI側で定義，生成された)オブジェクトの参照渡し\*4で，(AI側での副作用によって)オブジェクトのプロパティの値を変更する． \*4http://ufcpp.net/study/csharp/sp\_ref.html

・eValue・・・マス目の良さを表す値 -> GUI設計書 p.1, Cell のプロパティ 白黒別のプロパティを持つ

・eValue へのアクセス関数 -> \*5 GUI設計書 p.1, Cell のプロパティ，p.3, **getEValue/3, setEValue/4**

**private void initializeEValue(Board board)**

isInitializedがfalseなら，参照先のboardのセル全ての各評価値(eValueBlack, eValueWhite) \*5を0に初期化し，isInitializedの値をtrueにする．isInitializedがtrueなら何もしない．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| Board | board | ボードオブジェクト(の参照)を渡す． |
|  | 戻り値 | － |

**Test方法　nextHand/3**のメソッド定義冒頭で**initializeEValue/1**を呼び出してボード表示を確認．

**public Place simpleNextHandPlus( Board board, Place previousMove, bool isBlack)**

incEValueOfAvailableMoves/2を呼び出して，**副作用で**参照先のboardのセルについて，指定の色の着手可能セル全ての評価値(eValueBlackまたはeValueWhite)を +1する．次に**simpleNextHand/3**を呼び出して

指定の手番(isBlack)が次に置く場所を左上から右に探して，おけるところに置いてその手を返す．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| Board | board | ボードオブジェクト(の参照)を渡す． |
| Place | previousMove | 黒(player)の(直前の)着手セルの位置をAI側で記録する． |
| bool | isBlack | 次の手番を表す．黒手番ならtrue，白ならfalse． |
| Place | 戻り値 | ボード上の着手した位置．**simplenextHand/3**と同じ戻り値を返す． |

**private void incEValueOfAvailableMoves(Board board, bool isBlack)**

boardの指定の色の着手可能セル全ての評価値を +1する．ヒント：**incEValue/4 ，canPlace/3**を用いよ

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| Board | board | ボードオブジェクト(の参照)を渡す． |
| bool | isBlack | 指定の色を表す．黒の着手可能セルならtrue，白ならfalse． |
|  | 戻り値 | －　副作用あり． |

**private void incEValue(Board board, int row, int column, bool isBlack)**

参照先のboardのセルについて，指定の色の指定したセルの評価値を +1する．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| Board | board | ボードオブジェクト(の参照)を渡す． |
| int | row | ボード上の位置（行，上下）を表す．一番上が0，一番下が7． |
| int | column | ボード上の位置（列，左右）を表す．一番左が0，一番右が7． |
| bool | isBlack | 指定の色を表す．黒の着手可能セルならtrue，白ならfalse． |
|  | 戻り値 | －　副作用あり． |

**考察課題1 (練習) nextHand/3**のメソッド定義冒頭で**initializeEValue/1**を呼び出して，ボードのセルの評価値表示がどう変化していくか観察し，表示される評価値の意味を説明せよ．

**7.4 メソッド(盤面状態の評価関数1)・・・**着手の質の数値化

実行時に盤面の評価値算出の評価関数を動的に制御\*1 する技法として**delegate機能**を利用する．

1. まず評価関数メソッド(への参照)を格納する変数の型として**Efuncデリゲート型**を宣言し，
2. 用いる盤面状態の評価関数をint戻り値，2引数(State型とbool型)の共通形式のメソッドとして定義．

評価関数メソッドの呼び出し時(**BasicAI**のメソッド側の定義)には，

1. Efunc型の変数を宣言し，呼び出したい評価関数メソッド名を代入(**Efunc efunc = canMoves**;)
2. **efunc(**state, isBlack**)**; で実行でき，**efunc**の値を書き換えれば，異なるメソッドが実行される．

\*1例えば中盤まではEf1で(着手を)評価し，隅が取れる状況になれば評価関数をEf2に変更して評価．

**7.4.1 基本となる評価関数メソッド**

**public delegate int Efunc(State state, bool isBlack);**

メソッドへの参照を格納できる型(として**Efuncデリゲート型**)を宣言するための記述．**Efunc型の変数**は，戻り値がintで2引数(State型とbool型)を持つメソッド(盤面状態の評価関数)への参照を格納できる．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| delegate | Efunc | (盤面)状態の評価関数メソッドへの参照を格納できる(変数の)型名．  戻り値がintで2引数(State型とbool型)を持つメソッド． |
| State | State | (Efuncの戻り値となる)評価値が計算される(盤面)状態の参照．  previousMoveプロパティを用いて算出． |
| bool | isBlack | 評価値が算出される(評価される)手番を表す．黒ならtrue，白ならfalse． |
|  | 戻り値 |  |

(参考) デリゲートを自作するメリット <http://jehupc.exblog.jp/16034401/>

**public int pieces(State state, bool isBlack)**

(盤面)状態の指定の色の石数を数え，返す．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| State | State | 評価値が計算される(盤面)状態． |
| bool | isBlack | 評価値が算出される(評価される)手番を表す．黒ならtrue，白ならfalse． |
| int | 戻り値 | (盤面)状態の指定の色の石数を返す． |

(ヒント) GUI設計書p.2, (参考) 少なく取る　<http://bassy84.net/othello.sukunakutoru.html>

**public int canMoves(State state, bool isBlack)**

(盤面)状態の指定の色の着手数(打てる手の数)を返す．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| State | State | 評価値が計算される(盤面)状態． |
| bool | isBlack | 評価値が算出される(評価される)手番を表す．黒ならtrue，白ならfalse． |
| int | 戻り値 | (盤面)状態の指定の色の着手数を返す． |

**Efuncの**実行テスト用メソッド(BasicAIクラスに追加)

**Place simpleNextHandPlus( Board board, Place previousMove, bool isBlack, Efunc efunc)**

**state/3**でboardの複製を生成し，efunc/2を呼び出して指定の色の着手可能セル全ての評価値にefunc/2で算出した評価値を代入し，**copyEValueOnCanPlace/2**でboardの参照へ(評価値を)コピーしてGUI側で可視化する．次にボード情報を元に，指定の手番(isBlack)が次に置く場所を左上から右に探して，おけるところに置く．

**7.4.2 少し強い評価関数メソッド**

**public int weithtedPieces(State state, bool isBlack)**

(盤面)状態の指定の色の石のあるマス目のweightTableの値(重み)の和を返す．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| State | State | 評価値が計算される(盤面)状態． |
| bool | isBlack | 評価値が算出される(評価される)手番を表す．黒ならtrue，白ならfalse． |
| int | 戻り値 | (盤面)状態の指定の色の石のあるマス目のweightTableの値(重み)の和を返す． |

**Test方法**weightTableが初期値1の状態で，マス目に表示される値がpieces/2　と同じかを調べる．

7.6節の**考察課題3(必修)で，**weightTableの値(重み)を調整し，強い評価関数を工夫して作成する．

**public int canMoves2Pieces(State state, bool isBlack)**

現在の手数を盤面の石数和-4 で算出し，指定手数(定数)と比較する．指定手数までは**canMoves/2** を返し，指定手数を超えた場合は**Pieces/2**を返す．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| State | State | 評価値が計算される(盤面)状態． |
| bool | isBlack | 評価値が算出される(評価される)手番を表す．黒ならtrue，白ならfalse． |
| int | 戻り値 | 指定された手数までは**canMoves/2** を返し，指定手数を超えた場合は**Pieces/2**を返す． |

**7.4.2節**まで実装できたら，**8章 State**クラス(pp.8-9, **8.1～8.4**節)の実装，テストを行うこと．

**7.4.3 強い評価関数メソッドへの改良（選択課題）**

**public int improvedWeithtedPieces(State state, bool isBlack)**

**weithtedPieces/2** に7.4.2節までの評価関数を加えて改良した評価関数を作成し，より強い手を選ぶための評価関数に改良せよ．

発展：独自の評価関数を作る場合には，**efunc(**state, isBlack**)**; で実行できる形式で記述すること．

**7.5 メソッドその2 (全幅(横型)サーチ機能の実現)**

**7.5.1節**のソースコードを3日目に配布するので，**simpleBdSearchNextHand/3** の本体冒頭のコメント，

ソースコードを読んで，**simplebdSearch/6** の各引数(サーチのパラメータ)の設定方法を理解すること．

**7.5.1 白黒それぞれの1手先読み**

public **Place simpleBdSearchNextHand(Board board, Place previousMove, bool isBlack)**

boardの複製(root\*6)を元に，評価関数と評価基準を設定して1手先読み**simplebdSearch/6**で手番の**着手**を決定．先読み過程としてboardの着手可能セルの評価値を算出・更新．着手後の**次状態**を次の手番で1手先読みし，着手可能セルの評価値を算出・更新．rootと次状態を順にclosedListの末尾に追加して**着手**を返す．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| Board | board | ボードオブジェクト(の参照)を渡す． |
| Place | previousMove | 相手の(直前の)着手セルの位置をAIに知らせる．(棋譜用データ) |
| bool | isBlack | 手番を表す．黒手番ならtrue，白ならfalse． |
| Place | 戻り値 | 1手先読みし，手番の評価基準で選んだボード上の着手した位置を表す．  パス(着手可能な場所なし)ならnull を返す．副作用でboard側の手番と次の手番の色それぞれの着手可能セルの評価値を算出・更新する． |

仕様の詳細(手順) 　ヒント：\*6 **isBlack =/= previousisBlack**

1. まず，boardの参照から探索木のrootノードを生成．(**State/3** のコピーコンストラクタ呼出し\*6)

depthプロパティを1に設定．

1. 手番の評価基準，手番・相手の評価関数で**simplebdSearch/6**を実行し，**着手**と**次状態**を決定．
2. **copyEValueOnCanPlace/2**(p.8)でroot状態の手番の各着手可能セルの手番色の評価値をboard側の対応セルの手番色の評価値にコピー(副作用)．rootをclosedListの末尾に追加．
3. **copyEValueOnCanPlace/2**で**次状態**の相手の手番(相手番)の各着手可能セルの相手番色の評価値を，board側の対応セルの相手番色の評価値にコピー(副作用)．次状態をclosedListの末尾に追加．
4. **次状態**のプロパティから手番の**着手**を返す．

**public State simplebdSearch(State root, bool isBlack, bool isMax, int nthPV,**

**Efunc currentEf, Efunc rivalEf)**

1. **expandNodeAndEvalThem/3**を呼んでroot(親)の手番の着手可能セルそれぞれについて，次状態を生成，**currentEf**で(その次状態を)評価し，(次状態を)親のnextStateListの末尾に追加する．
2. **sortNextStateList/2**(p.10)で親(**root**)のnextStateListから着手する次状態を決定．
3. **expandNodeAndEvalThem/3**を呼んで，次状態に対し，相手の手番(!isBlack)の着手可能セルそれぞれについて次々状態を生成し，**rivalEf**で(次々状態を)評価する．
4. 次状態を返す．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| State | root | サーチ木の起点(ルートノード), 現在のボード状態(盤面)の複製(の参照)． |
| bool | isBlack | 手番を表す．黒手番ならtrue，白ならfalse． |
| bool | isMax | 手番の基準．次状態リストを評価値(eValue)でソートする．trueなら降順，falseなら昇順．nthPVとペアで着手(後の次状態)を決定する． |
| int | nthPV | 手番の基準．評価値でソートされた次状態を選ぶ順位を表す．1なら先頭． |
| Efunc | currentEf | 手番の評価値を算出する評価関数のメソッド名．(p.9, 8.3節) |
| Efunc | rivalEf | 相手の評価値を算出する評価関数のメソッド名． |
| State | 戻り値 | 手番の基準に従って決定した，次状態を返す． |

これらのメソッドを実装して**nextHand/3**から**simpleBdSearchNextHand/3**を呼び出すと，指定した評価関数で算出した評価値を白黒それぞれの着手可能セルに表示し， 1手先読みして指定した評価関数と評価基準で着手を選ぶ(少しだけ賢い)AIと対戦できる．

テスト用メソッド：まず最後の2引数を省略(本体に直接記述)した**simplebdSearch/4** を作成．動作確認後，

デリゲートによる引数によるメソッド渡しを実装して，動的なメソッド指定をテスト．

**private void expandNodeAndEvalThem( State state, bool isBlack, Efunc efunc)**

親状態から手番の着手可能セルそれぞれについて，指定された評価関数で**lookAhead/4** (p.4)を呼び出し，次状態を生成． その次状態を親の**nextStateList**の末尾に追加する．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| State | root | サーチ木の起点(ルートノード), 現在のボード状態(盤面)の複製(の参照)． |
| bool | isBlack | 手番を表す．黒手番ならtrue，白ならfalse． |
| Efunc | efunc | 評価値を算出する評価関数のメソッド名． |
|  | 戻り値 |  |

**private State lookAhead( State state, Place move, bool isBlack, Efunc efunc)**

**State/3**を呼んで(stateの複製に手番(isBlack)の着手(move)を実行して)次状態を生成．次状態の評価値(eValue)を，指定した評価関数(**efunc**)で算出・更新し，stateの着手したセル(move)の手番色の評価値に代入する．次状態を返す．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| State | state | 1手先読みの起点となる状態．着手可能セルの手番色の評価値を更新する． |
| Place | move | state に対する手番の着手（着手したセル）． |
| bool | isBlack | 手番を表す．黒手番ならtrue，白ならfalse． |
| Efunc | efunc | 状態の評価値を算出する評価関数のメソッド名． |
| State | 戻り値 | stateの複製に手番の着手を実行して生成した次状態を返す．  次状態のeValueプロパティには算出した評価値が代入される． |

7.4.1節の実装&テストが終われば，7.5節の実験課題(初心者向けの接待playとの対戦)ができるようになる．

**7.5.2 n手先読み　（上級選択課題）**

public **Place bdSearchNextHand( Board board, Place previousMove, bool isBlack, int depth)**

引数にdepthを追加して深さdepthの先読み評価をする横型探索を実装せよ．(再帰呼び出しで記述)

ミニマックス法, multiPVの概要を調べてから設計すること．**simplebdSearch/8**と同様に，相手の着手選択の評価関数・評価基準は(rivalEf, rivalIsMax, rivalNthPV)とする．(rivalNthPV = 1なら最善手(または最悪手)選択を意味する．) 状態の評価値は，深さdepthのサーチ木の葉ノードを白，黒それぞれの評価関数で色別に算出し，先読み状態から1手戻った状態に評価値を再帰呼出しの戻り値で色別に伝播させる．

実行時はdepthの値を1から順に大きくして，実用的な計算量の範囲で実行できたdepthの最大値を調べよ．

(最大値を調べようとするとハングアップする可能性があるので事前にファイルをセーブしておくこと．)

**7.6 実験および考察課題 (5日目)**

**考察課題2(必修)**

1. 7.5.1節の白黒それぞれの1手先読みのアルゴリズムの動作をサーチ木として図で説明せよ．

・まず，白，黒それぞれの着手可能セル数を2として，深さ2の横型サーチ木を描く．

・白，黒それぞれの着手可能セルに代入(可視化)される評価値がどのノードから伝播するかを図示する．

**考察課題3(必修)**

**7.4.2 少し強い評価関数メソッド**の**weithtedPieces/2**で用いる重みテーブルweightTableの値を調整して，できる限り強い手（例：負けない手）を選ぶための評価関数を作成し，用いたweightTableを図で説明せよ．

作成した評価関数を用いて実験課題3を行い，記録した対戦データをもとに，作成した評価関数を評価せよ．

**実験課題3 (必修)**

1. self-playによるplayer自身の上達度とAIの賢さ評価
   1. 対戦用AIを準備し，AIの強さ設定(先読みの深さ，用いた評価関数・基準)を説明せよ．
   2. 自身で作成したAIと5-10回程繰り返し対戦し，ゲーム状況(白石数, 黒石数, 石差)を序盤(20手目)，中盤(40手目)、終盤(60手目)ごとに記録し，対戦結果の履歴をグラフ化し，自身の上達度を説明せよ．

(1-3) 負けなくなるまでに要したゲーム数とAIの強さ設定との関係を考察せよ．

(1-4) 自身で作成・対戦したAIの賢さを定量的，主観的それぞれの観点で評価せよ．

**考察課題4 (選択)**

**7.4.3節 強い評価関数メソッドへの改良**を行い，作成した作成した評価関数を説明せよ．

作成した評価関数を用いて実験課題4を行い，記録した対戦データをもとに，作成した評価関数を評価せよ．

**実験課題4 (選択)**

(2) 対戦によるAIの賢さの相互評価　(但し設定した内容を相手に知らせずに実験・評価すること)

(2-1) 班員相互で他者が作成したAIと2-3回繰り返し対戦し，対戦相手とゲーム結果を記録せよ．

(2-2) 対戦したAIの評価関数・基準を推定し，AIの賢さを評価せよ．

**実験課題5 (中級選択)**

初心者(人間player)を対象として，オセロの対戦スキルを上達支援するための，着手可能セルへ可視化する評価値，およびAIの着手制御のための適切な評価関数・基準を設計し，初心者を被験者とした繰り返し対戦実験によって設計したAIの有効性を評価せよ．

条件：各被験者に対し，Player/AI用評価関数・基準を同一の条件下で設定して対戦用AIを準備する．

**実験課題6 (上級選択)**

人間playerを相手に接待play(接戦)を演じるAIの着手制御を設計し，複数の被験者との対戦で評価せよ．

(1) 対戦結果での接戦の度合い，対戦結果の記録，対戦相手が受けた印象のアンケート調査から，設計した接待playの質を定量的に評価せよ．

(2) より精度の高い接待playを演じるためのAIの着手制御の改良点，設計のポイントを考察せよ．

1. **State**

ボードの探索状態(Board Search State)を表すクラス．Boardクラスの派生クラスとして定義する．

(class State : Board) オセロゲームの盤面の遷移状態を表す**探索木**の各ノードの情報を保持する．

2015-5-16 変更点　nextStateListの初期化 nextStateList = new List<State>();

**（説明）**

**8章でのAIプログラム向けのC#プログラミング技法**

1. オブジェクトの変数への代入・・クラスは**参照型**\*1：参照情報(実体がどこにあるかの情報)を代入

\*1 http://ufcpp.net/study/csharp/oo\_reference.html

1. **動的なデータ構造**

**List(動的一次元配列)**  ・・・実行時に長さを変更できる配列 -> 8.1節

オブジェクトの複製　 ・・・盤面の先読み(可能な着手の並列評価)のために用いる -> 8.2節

　　tree ・・・探索（先読みの枝分かれ）を表すためのrootを起点とする木構造，

StateオブジェクトとnextStateListプロパティとで**再帰的**に記述．

1. **動的なメソッド呼び出し** ・・・実行中に呼び出すメソッドを引数で与える． -> 7.4節
   1. **プロパティ**

|  |  |  |
| --- | --- | --- |
| 型 | 名前 | 説明 |
| public Place | previousMove | このボード状態(盤面)の(直前の)着手(セル)の位置． |
| public bool | previousIsBlack | 指定の色で(直前の)着手を区別．黒ならtrue，白ならfalse． |
| public int | eValue | 指定の色(isBlack)でのこのボード状態(盤面)の評価値． |
| public State | previousState | この状態(盤面)の親ノード(1手前の状態)．Stateオブジェクト(の参照) |
| public List<State> | nextStateList | この状態(盤面)から着手(遷移)可能な次状態(全て)のリストで，Stateオブジェクト(の参照)を要素とする**List(動的一次元配列)**\*． |

\* **List<** (型名) **>** と記述することで(型名)型の**リスト(動的な配列)**を作成

http://www.ipentec.com/document/document.aspx?page=csharp-use-list-generics

* 1. **メソッド(基本)**

**public State()**デフォールトコンストラクタ．eValueは0に，nextStateListは new List<State>(); で

初期化する．

**public State(Board source, Place previousMove, bool previousIsBlack)**

探索木のrootノードを生成するためのコピーコンストラクタ．引数sourceオブジェクトの複製(ディープコピー)を生成する．まず基底クラス(**Board**)のコピーコンストラクタを引数つきで明示的に呼び出して\*1，ボード状態の複製を作る．次に派生クラス(**State)**のプロパティの値を初期化する．previousMove, previousIsBlackに引数の値\*2を代入し，eValueは0に，previousStateはnullに, nextStateListは new List<State>(); で初期化．

ヒント \*1 :base(source) {定義本体 ，\*2 this. を使って引数とプロパティとを区別する．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| Board | source | 基底クラスの複製を作るためのボードオブジェクト(の参照)．  探索木のrootノードのボード状態． |
| Place | previousMove | このボード状態(盤面)の(直前の)着手(セル)の位置． |
| bool | previousIsBlack | 指定の色で(直前の)着手を区別．黒ならtrue，白ならfalse． |
| State | 戻り値 | － |

**public State(State source, Place move, bool isBlack)**

sourceオブジェクトから探索木の子ノード(次状態，先読みするための盤面)を生成するコピーコンストラクタ．まず基底クラス(**Board**)のコピーコンストラクタを引数つきで明示的に呼び出してボード状態の複製を作り，this（生成したオブジェクト）に基底クラスのset/3メソッドを用いて(moveで)1手進めて状態を更新する．次に派生クラス(**State)**のプロパティの値を初期化する．previousState, previousMove, previousIsBlackに引数の値を代入し，eValueは0に，nextStateListは new List<State>(); で初期化する．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| State | source | 複製するオブジェクト．次に展開する(調べる)ノードを表す．子ノード(次状態)を作るための親ノード． |
| Place | move | sourceノードからこのボード状態(盤面)へ遷移する着手(セル)の位置． |
| bool | isBlack | 指定の色でこのボード状態(盤面)へ遷移する着手(を区別)． |
| State | 戻り値 | － |

**public void copyEValueOnCanPlace( Board board, bool isBlack)**

指定の色の着手可能セル全ての(指定の色の)eValue (黒ならeValueBlack，白ならeValueWhite)を，board

の同一セルの指定の色のeValueへコピーする．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| Board | board | 基底クラスのボードオブジェクト(の参照)．  探索木のrootノードのボード状態． |
| bool | isBlack | 指定の色で(直前の)着手を区別．黒ならtrue，白ならfalse． |
|  | 戻り値 | － |

* 1. **メソッド (7.4節　全横型サーチで用いる，ソート用のメソッド)**

**public State sortNextStateList(bool isMax, int nth)**

nextStateList を評価値(eValue)の降順または昇順でソートし，先頭からnth番目の要素を返す．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| bool | isMax | 次状態リストを評価値(eValue)でソートする手番の基準．trueなら降順，falseなら昇順．nthとペアで着手(後の次状態)を決定する． |
| int | nth | 評価値でソートされた次状態を選ぶ順位を表す(手番の)基準．0なら先頭．  注意 nth = nthPV -1; |
| State | 戻り値 | ソートしたnextStateListの先頭からnth番目の要素を返す |

ヒント　・List<T> を任意にソートする方法　　　http://programmers.high-way.info/cs/list-sort.html

・リストの要素に，リストのインスタンス[インデックス番号]でアクセスする方法

<http://www.ipentec.com/document/document.aspx?page=csharp-use-list-generics>

ここまで実装したら，次は7.5節(p.5)へ．

**8.5メソッド(盤面状態の評価関数2)（中級・上級選択課題）**

実験第4週時に，7.5.1節の白黒1手先読み探索が完成した場合に，以下の評価関数を実装してよい．

(BasicAIクラスのメソッドに追加)

**public int fixedPieces(State state, bool isBlack)　 （中級選択課題）**

盤面状態での指定の色の確定石数の概数(隅，辺の確定石数)を返す

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| State | state | 評価値が計算される(盤面)状態． |
| bool | isBlack | 確定石数を評価する手番(指定の色)．黒ならtrue，白ならfalse． |
| int | 戻り値 | 指定の色の確定石数の概数(隅，辺の確定石数)を返す． |

　(参考)辺の確定石の数計算方法　　　出典：<http://uguisu.skr.jp/othello/5-2.html>

まず石が隅に１つでもあるか調べる．（無ければ確定石は無い）

隅に石がある＆自分の石であれば、そこから辺の石の連続は確定石．

両隅に石＆相手の石の場合、途切れていなければその辺は確定石．

**public int openness(State state, bool isBlack)　 （上級選択課題）**

指定の位置(previousMoveプロパティ)へ指定の色を置いた着手の**開放度**を返す．

|  |  |  |
| --- | --- | --- |
| 型 | 引数 | 説明 |
| State | state | 評価値が計算される(盤面)状態． |
| bool | isBlack | 開放度を評価する手番(指定の色)．黒ならtrue，白ならfalse． |
| int | 戻り値 | 指定の位置へ指定の色を置いた場合の開放度を返す． |

開放度とは？　着手によって裏返ったピースそれぞれの8近傍での空白セル数の和

<http://bassy84.net/othello.kaihoudo.html>

ヒント：GUI設計書 p.4, reverse/3 の算出手順の一部(裏返すピース位置をそれぞれ/全て返す)をメソッドとして切り出す．例えばbool canFlip/5 -> private **List<Place> canFlip/5**

指定のセルから指定の方向へ挟めるか調べ，裏返るピース位置のリストを返す．挟めなければnullを返す．

**private int countBlank(Place place)**

指定位置の周囲8近傍でピースのないセルの数を返す． 　　　　　　　　　　　　　　　　　　以上