NEMO

Slide tecniche

Mattia Campestri
Tommaso Liverani
Gioele Pisanelli

Implementazione Transposition Table

- Riga della table implementata come array di entry
- AddEntry() inserisce l'entry in testa all'array e "slitta" gli altri elementi eliminando l'ultimo (il più vecchio)
- GetTransposition restituisce l'entry associata ad una certa board, con la bestMove opportunamente modificata nel caso di board simmetriche

```
typedef struct {
    Entry row[TRANSPOSITION_TABLE_WIDTH];
    int8_t dim;
} TraspositionRow;


// Global variables
TraspositionRow WHITE_TABLE[TRANSPOSITION_TABLE_DIM];
TraspositionRow BLACK_TABLE[TRANSPOSITION_TABLE_DIM];


Move getBestMoveWhite(BoardState* currentBoard);
Move getBestMoveBlack(BoardState* currentBoard);

void addWhiteEntry(BoardState* board, Move nextMove, ScoreType euristicValue, uint8_t depthLeft, uint8_t type);
void addBlackEntry(BoardState* board, Move nextMove, ScoreType euristicValue, uint8_t depthLeft, uint8_t type);
void removeWhiteEntry(uint64_t hash);
void removeBlackEntry(uint64_t hash);

Entry getWhiteTrasposition(BoardState* currentBoard, Move* bestMove, int* euristicValue);
Entry getBlackTrasposition(BoardState* currentBoard, Move* bestMove, int* euristicValue);
```

Storia e controllo pareggi

- Il controllo dei pareggi consiste nella consultazione di un array composto dalle scacchiere passate già giocate e quelle future attualmente in valutazione
- Se la scacchiera al momento in analisi è uguale ad una di quelle presenti nell'array, allora si tratta di un pareggio
- Altrimenti viene aggiunta alle scacchiere future

```
void addFutureBoard(HistoryArray* history, BoardState board) {
    history->boards[(history->size) % HISTORY_DIM] = board;
    history->size++;
}

void addPlayedBoard(HistoryArray* history, BoardState board) {
    history->boards[history->playedBoardSize % HISTORY_DIM] = board;
    history->playedBoardSize++;
    history->size = history->playedBoardSize;
}

int isDraw(HistoryArray* history, BoardState board) {
    int i;
    for (i = 0; i < history->size; i++) {
        if (isEqual(history->boards[i % HISTORY_DIM], board))
            return 1;
    }
    return 0;
}
```

Mosse e catture

- Spostamenti pedine tramite macro sullo stato
- Controllo catture tramite matching con pattern
- Nello specifico, controllo catture con accampamenti o trono attraverso una finta scacchiera in cui sono state poste in tali caselle pedine alleate

```
#define APPLY_MOVE_WHITE(board, move) board[START_ROW(move)] ^= (1U << (17U - (START_COLUMN(move) << 1U)));
                                      board[TARGET_ROW(move)] ^= (1U << (17U - (TARGET_COLUMN(move) << 1U)))
#define APPLY_MOVE_BLACK(board, move) board[START_ROW(move)] ^= (1U << (16U - (START_COLUMN(move) << 1U)));
                                      board[TARGET_ROW(move)] ^= (1U << (16U - (TARGET_COLUMN(move) << 1U)))
#define APPLY_MOVE_KING(board, move) board[START_ROW(move)] ^= (3U << (16U - (START_COLUMN(move) * 2U)));
                                     board[TARGET_ROW(move)] ^= (3U << (16U - (TARGET_COLUMN(move) * 2U)))

uint32_t horizontalKingMask = 0b000000011101000000U; // nero re nero, già shiftata per il primo check
uint32_t horizontalMask = 0b0000000000011001U; // nero bianco nero
uint32_t verticalBlackMask = 0b0100000000000000U; // nero...
uint32_t verticalWhiteMask = 0b1000000000000000U; // bianco...
uint32_t verticalKingMask = 0b1100000000000000U; // re...

void fakeBoardBlack(BoardState * state) {
        uint32_t fakeRow08 = 0b000000010001000000U; // non setto casella centrale accampamento
        uint32_t fakeRow17 = 0b000000000100000000U;
        uint32_t fakeRow35 = 0b0100000000000001U;
        uint32_t fakeRow4 = 0b0001000000100000100U; // non setto casella centrale accampamento

        // tramite l'& elimino il re dal  trono in caso ci sia
        state->Board[0] = state->Board[0] | fakeRow08;
        state->Board[1] = state->Board[1] | fakeRow17;
        state->Board[3] = state->Board[3] | fakeRow35;
        state->Board[4] = (state->Board[4] & 0b1111111100111111111U) | fakeRow4;
        state->Board[5] = state->Board[5] | fakeRow35;
        state->Board[7] = state->Board[7] | fakeRow17;
        state->Board[8] = state->Board[8] | fakeRow08;
}
```