Cel mai important e de rezolvat bugul de final

More depth always better than better evaluation

imbunatatiri de adaugat: rosu=neimportante, verde=importante

|  |  |
| --- | --- |
| Ca dificultate | * Zugswang (se face cu null move pruning, ajuta 20-75% cica) * Quiverence search (with delta pruning) – cica poate ocupa pana la 50% din timpul total, aici daca nu merge null move pruning atunci eval * Sa functioneze cat timp gandeste celalalt * King lee?? |
| Ca viteza | * bitboard move generation -cred ca ramane pentru peste 2 dati * daca tot nu fac cu bitboards, trebuie chestia cu doua margini libere la tabla ( exemplu alb 1-6, liber 10, negru 11-16, margine -1) * Transposition table – ititializat cu opening table * Multithreading (atentie fara multe lockuri) * Zero window search |
| Interfata | * Generalizare alb-negru * undo, redo, draw, resign * Status ready/thinking * Afisare piese capturate * Transformare drop-down * Editare cu Set de piese doar de o culoare, schimbi uloarea cu click stg/dr sau cicland click stanga * Probabil trebuie o clasa player |
| Altele | * Maturi clasice * Timp(la meciurile contra timp cand gasesti mutare buna la depth n-1, o cauti doar pe aia la depth n ca sa economisesti timp)(se face foarte bine cu iterative deepening daca se foloseste transposition table * Advanced remiza din lipsa de material * Minimax => remiza>victorie spectaculoasa cu sanse minime pt cel mai dificil, opac, sec si contraintuitiv contraatac * In general preprocesarea e mai importanta decat utilizarea memoriei * Monte Carlo tree search (ideea e ca nu trebuie minimizat nr posibilitati al adversarului cata vreme ele sunt inferioare) * Pentru endgame valoarea jucatorului mai slab inmultita? |
| Buguri | * Am rulat cu scor nr intreg, nu fractionar * Vezi bugul la queen ending ca pierde regina ca prostul * Nu am reusit refutation table si history table, de ce? (history 64x64? De ce?) |

Idei de pornire pentru interfata

* Daca amandoi jucatorii sunt oameni, table se roteste la fiecare tura
* Daca maxim un jucator e om, tabla nu se roteste automat niciodata

notite v5

|  |  |  |
| --- | --- | --- |
|  | Avantaje | dezavantaje |
| Campuri clasa | -cod separat si structurat | -probleme la cazurile speciale(parametrii in plus, etc)  - la mutare se muta mai multe chestii desi nu e nevoie de toate |
| Campuri char/int | -nu sunt probleme la cazurile speciale  -mutarea muta doar ce e nevoie | -codul nu e separat si structurat |

Pana la urma trebuie campuri clasa pentru determinarea supraincarcarii pieselor si schimburi mai eficiente, dar trebuie verificat daca e compatibil cu algoritmul alfa-beta

Programele profesionale folosesc bitboards, deci clar nu e bine cu clasa

verificarea regelui daca e in sah

|  |  |  |
| --- | --- | --- |
|  | Avantaje | dezavantaje |
| Atacat bool | -rapid | -cod partial repetitiv |
| Atacat mutari | -daca se omite calculul mutarilor, rapid  Altfel incet  -nerepetitiv | -trebuie calculate mutarile  (celuilalt player, deci practic ambilor) |
| Atacat stiva | -exista deja pt evaluare control | -lent, calculeaza si apararea |

-trebuie vectori in loc de liste pt mutari pt viteza

-determinarea mutarilor corecte nu se poate face decat considerand si mutarea regelui de fiecare data (mutarile regelui nu pot fi determinare direct corect)

-determinarea mutarilor corecte se face imediat dupa determinarea generala

-retinerea coordonatelor regilor vs cautarea unui rege de fiecare data

In varianta initiala de alfabeta, daca gaseste un mat si apoi un altul in mai putine mutari, il face pe primul gasit. E important ca nu se aplica decat in cazul matului, deoarece in cazul unei formule complicate e imposibil sa fie pur si simplu 2 valori egale, si chiar daca ar fi, m-as complica prea mult sa rectific asta. In concluzie, valoarea matului trebuie sa depinda de depth.

Complexitatea alfabeta, e verificata

Nr nou=nr vechi(2+nr)

Nr vechi=nrnou/(2+nr)

Comparatie timp - medie din 3 – in secunde

Rezulta din multe multe teste c++ si c# ca prealocarea variabilelor nu ajuta

Prealocarea nu e multithreading friendly

Class e la fel de rapid ca struct c++ si c#

**C++ si optimizari – basic 4**

C# debug 11.35, 11.68, 11.37 =>11.47

C# release 2.91, 2.76, 2.97 => 2.88

C++ debug 1.71, 1.72, 2.08 => 1.84

C++ release 0.63, 0.74, 0.63 => 0.67

C# release unsafe 2.81, 2.85, 2.92 => 2.86 la fel

**C++ si optimizari – position 4**

C# debug 9.38, 9.43, 9.41 =>9.41

C# release 2.66, 2.69, 2.69 => 2.68

C++ debug 1.69, 1.69, 1.69 =>1.69

C++ release 0.62, 0.62, 0.62 => 0.62

**optimizari si evaluare – 5**

C++ debug basic 23.79, 23.68, 24.47 => 23.94 ////67.69

C++ debug pos 13.53, 13.55, 13.53 => 13.54///39.67

C++ release basic 23.71, 23.7, 24.49 => ~23.9

C++ release pos 13.52, 13.57, 13.57 => 13.55

**Ordonare mutari – irelevant, e mult mai eficient cu TT**

Vs dll release/release, depth 4, no order 0.78

Vs dll release/release, depth 5, no order 14.98

Vs dll release/release, depth 4, order all, eval d0 1.30

Vs dll release/release, depth 5, order all, eval d0 18.14

Vs dll release/release, depth 4, order first, eval d0 0.73

Vs dll release/release, depth 5, order first, eval d0 13.02

Vs dll release/release, depth 4, order first, eval d1 0.72

Vs dll release/release, depth 5, order first, eval d1 12.72, 12.86

**Refutation table – c# rel 4, debug**

Fara, Total 16.18

1. e2-e4 4.25 b8-c6

2. g1-f3 2.07 a8-b8

3. f1-c4 4.14 g8-f6

4. d2-d3 3.94 e7-e5

5. e1-g1 1.78 b7-b5

Cu, total 16.32

1. e2-e4 4.32 b8-c6

2. g1-f3 2.19 a8-b8

3. f1-c4 4.28 g8-f6

4. d2-d3 4 e7-e5

5. e1-g1 1.53 b7-b5

**History table – c# rel 4, debug**

Fara, total 15.83

1. e2-e4 4.34 b8-c6

2. g1-f3 2.11 a8-b8

3. f1-c4 4.17 g8-f6

4. d2-d3 4.4 e7-e5

5. e1-g1 1.81 b7-b5

Cu, total 28.21

1. e2-e4 3.7 b8-c6

2. g1-f3 1.7 g8-f6

3. b1-c3 4.44 b7-b6

4. f1-c4 9.34 e7-e5

5. e1-g1 9.03 f8-b4

**History table - clasic speed test, c# rel 4, debug**

Normal 11.66

Prima cu 8.94

A doua cu 13 ???????

A treia cu 15.98 ??????

A patra cu 15.1

Wtf too good

Prima cu v2 9.16

A doua cu v2 7.06

A treia cu v2 3.23

A patra cu v2 1.86

A cincea cu v2 1.95

**History table – c# rel 4, debug - determinist**

Fara, total 62.57

1. e2-e4 1.97 d7-d5

2. d1-f3 12.2 d5-e4

3. f3-e4 20.43 g8-h6

4. g1-f3 16.13 e8-d7

5. f1-c4 11.84 f7-f6

Cu, total ~77

1. e2-e4 4.59 d7-d5

2. d1-f3 9.77 d5-e4

3. f3-e4 19.71 f7-f6

4. g1-f3 17.25 e7-e6

5. f1-c4 28.67 e6-e5

**Cpp implementare**

C# pur, position4, debug 11.48, 11.24, 11.05 => 11.25

C# pur, position4, release 2.70, 2.71, 2.73 => 2.71

Vs dll, position4, debug/debug 4.58, 4.27, 4.45 => 4.43

Vs dll, position4, release/release 0.71, 0.81, 0.73 => 0.75

cb dll

2 processes