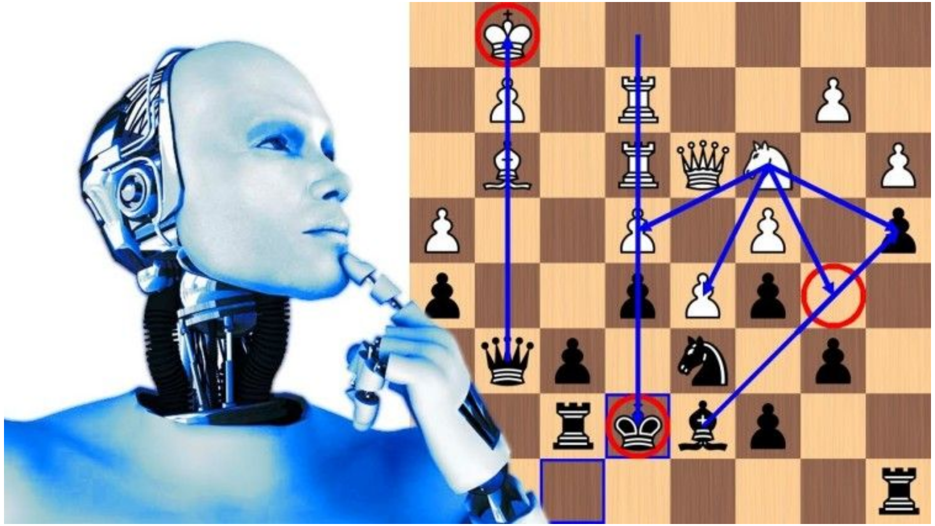
Early AI: A Brief History of a Chess Engine



By Pierluigi Meloni

About me..

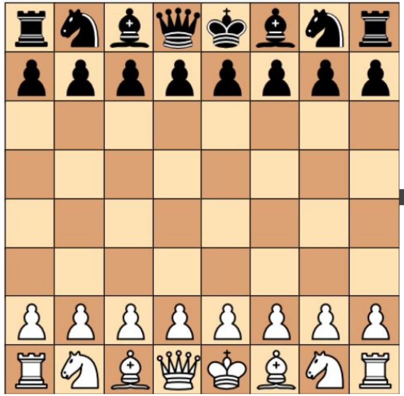
- Born in early 70’s..perfect for falling in love with one of the first microcomputers in 1982..(Sinclair Spectrum)
- Husband to Gemma, father to Edoardo (18) and Francesco (13). Room-mate with Frida (cat)
- Engineer (automation), working in the Oil&Gas industry
- Crazy about chess (creator of STUChess), programming (anything..), IOT, Android Apps (father of MQTT Alert) ,3d printing,esp8266,small robots,AI...mostly CV, Drums playing,Judo,Cooking,Travelling & Walking (like an hamster..)

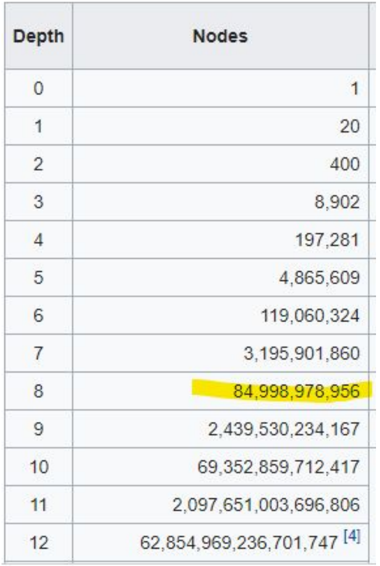
https://www.linkedin.com/in/pierluigi-meloni-910860/

Chess is computationally impossible...

- There are more possible ways to make the first 4 moves (8 ply) from the starting position than estimated stars in our galaxy (perft(8)= 84 billions vs 40-50 billions...)
- humans could not calculate so far perft(16). Perft(15) is 2,015,099,950,053,364,471,960

https://www.chessprogramming.org/Perft_Results





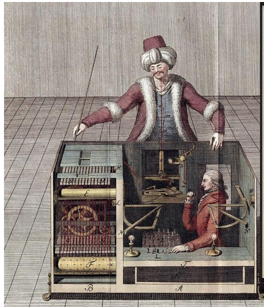
Depth	Nodes
0	1
1	20
2	400
3	8,902
4	197,281
5	4,865,609
6	119,060,324
7	3,195,901,860
8	84,998,978,956
9	2,439,530,234,167
10	69,352,859,712,417
11	2,097,651,003,696,806
12	62,854,969,236,701,747 [4]

>

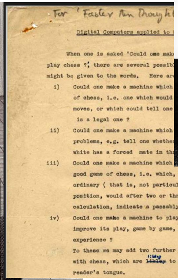
…which is why great minds tried to create "chess playing machines"

..just few examples..

The Turk(1770 -1805)



Turing '48



For 'Faster Than Though

Digital Computers applied to

When one is asked 'Could one mak
play chess ?', there are several possib
might be given to the words. Here ar

i) Could one make a machine which
of chess, i.e. one which would
moves, or which could tell one
is a legal one ?

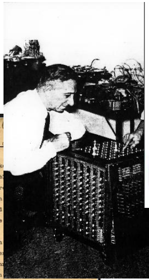
ii) Could one make a machine which
problems, e.g. tell one whethe
white has a forced mate in th

iii) Could one make a machine which
good game of chess, i.e. which, confronted with an
ordinary (that is, not particularly unusual) chess
position, would after two or three minutes of
calculation, indicate a passably good legal move?

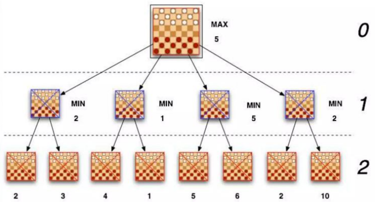
iv) Could one make a machine to play chess, and to
improve its play, game by game, profiting from its
experience ?

To these we may add two further questions, unconnected
with chess, which are ~~likley~~ likely to be on the tip of the
reader's tongue.

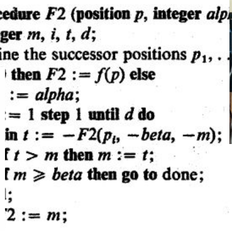
Shannon 50s



Von Neuman minmax 50s



Knuth alpha beta!!! '75

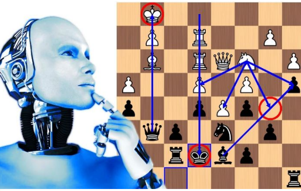


```
integer procedure F2 (position p, integer alp
  begin integer m, i, t, d;
    determine the successor positions p_1, . .
        then F2 := f(p) else
        := alpha;
        := 1 step 1 until d do
        in t := −F2(p_i, −beta, −m);
        f t > m then m := t;
        f m ⩾ beta then go to done;
        ;
        2 := m;
```

Ken Thompson 80's (first master level)

IBM Deepblue '96

Google AlphaZero 2017



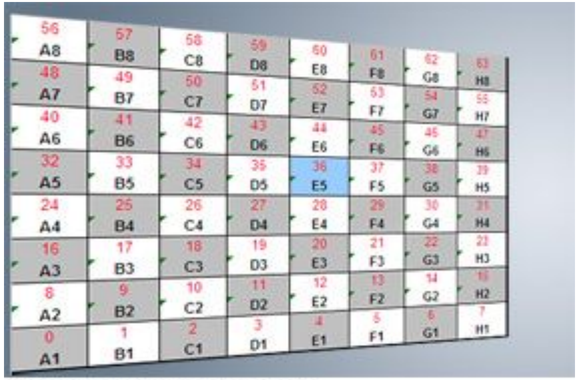
AI

Many of these people won the Turing award…i.e the Nobel prize for Computer Science

..but what does it take to write a chess engine?

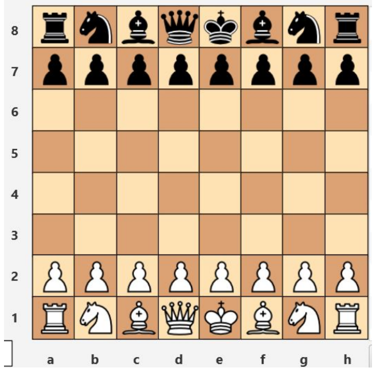
Step 1: Chess Board Representation and data structure

- Chessboard and pieces representation: arrays? Bitboard? Mixed approach?
- While–do vs bit shifts
- **Maximize bitwise operators** where possible
- Data structures to support moves generation arrays
- Keep it simple and clean as all the rest rely on this
- Choices made in this phase are very difficult to change
- Add a FEN interpreter. Very important for testing (https://en.wikipedia.org/wiki/Forsyth%E2%80%93Edwards_Notation) ex: `rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1`
- Don’t waste too much time on GUI but you need one..remember keep it simple..Unicode for chess pieces are good enough: `0x2654,0x2655,0x2656,0x2657,0x2658,0x2659,0x265A,0x265B,0x265C,0x265D,0x265E,0x265F`



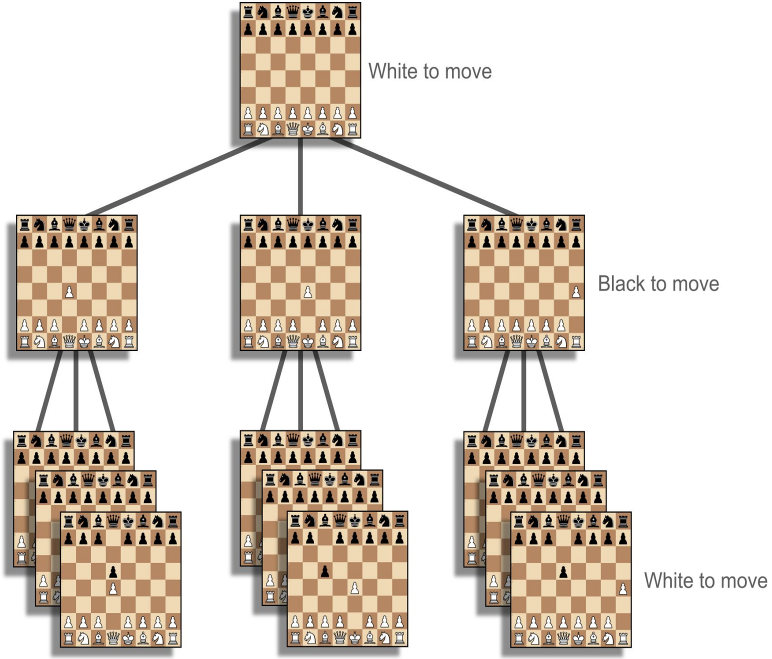
56 A8	57 B8	58 C8	59 D8	60 E8	61 F8	62 G8	63 H8
48 A7	49 B7	50 C7	51 D7	52 E7	53 F7	54 G7	55 H7
40 A6	41 B6	42 C6	43 D6	44 E6	45 F6	46 G6	47 H6
32 A5	33 B5	34 C5	35 D5	36 E5	37 F5	38 G5	39 H5
24 A4	25 B4	26 C4	27 D4	28 E4	29 F4	30 G4	31 H4
16 A3	17 B3	18 C3	19 D3	20 E3	21 F3	22 G3	23 H3
8 A2	9 B2	10 C2	11 D2	12 E2	13 F2	14 G2	15 H2
0 A1	1 B1	2 C1	3 D1	4 E1	5 F1	6 G1	7 H1

Square Centric Board Representation



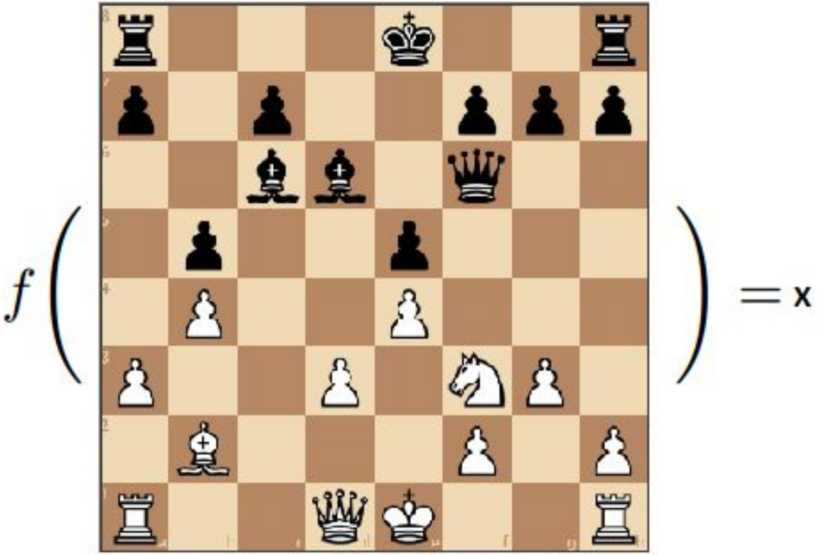
Step 2: Moves Generation plus make/unmake

- Generate **ALL LEGAL** moves given a starting position and the side to move
- Pseudo legal vs legal
- Way more complex than it seems: chess moves complex (special moves, sliding pieces, flags). ' easy to give up here..
- Speed is the key as it will be called millions of times...optimize, optimize, optimize
- Debugging easy
- Assign “pre-score” (very useful for sorting later
- Implement **make/unmake** to execute the move and take back (absolutely NOT trivial)
- Make sure you pass Perf test



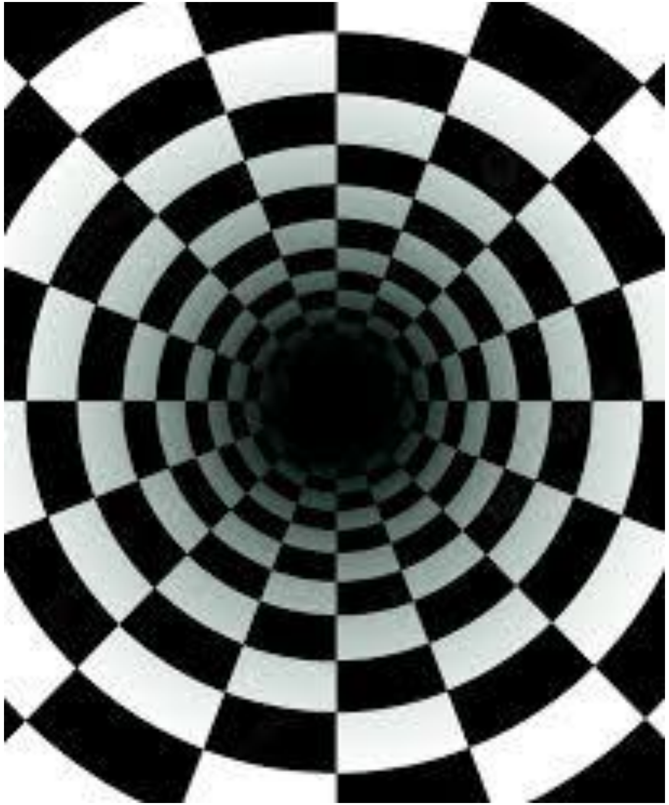
Step 3: Create an evaluation function

- Create a function that assigns a static score to a position (from - inf to inf)
- Very easy to code and to debug. Can be made incredibly sophisticated…
- You can make a decent engine with just values algebraic sum (queen 900, rook 500, knights and bishops 300, pawn 100).
- X in centipawn (i.e. one pawn = 100)
- X > 0 good for white, X < 0 good for black → white tries to **Maximize**, black tries to **Minimize**
- Compromise complex eval function vs speed.
- Start thinking about transpositions (performance killer…): hash table with Zobrist key

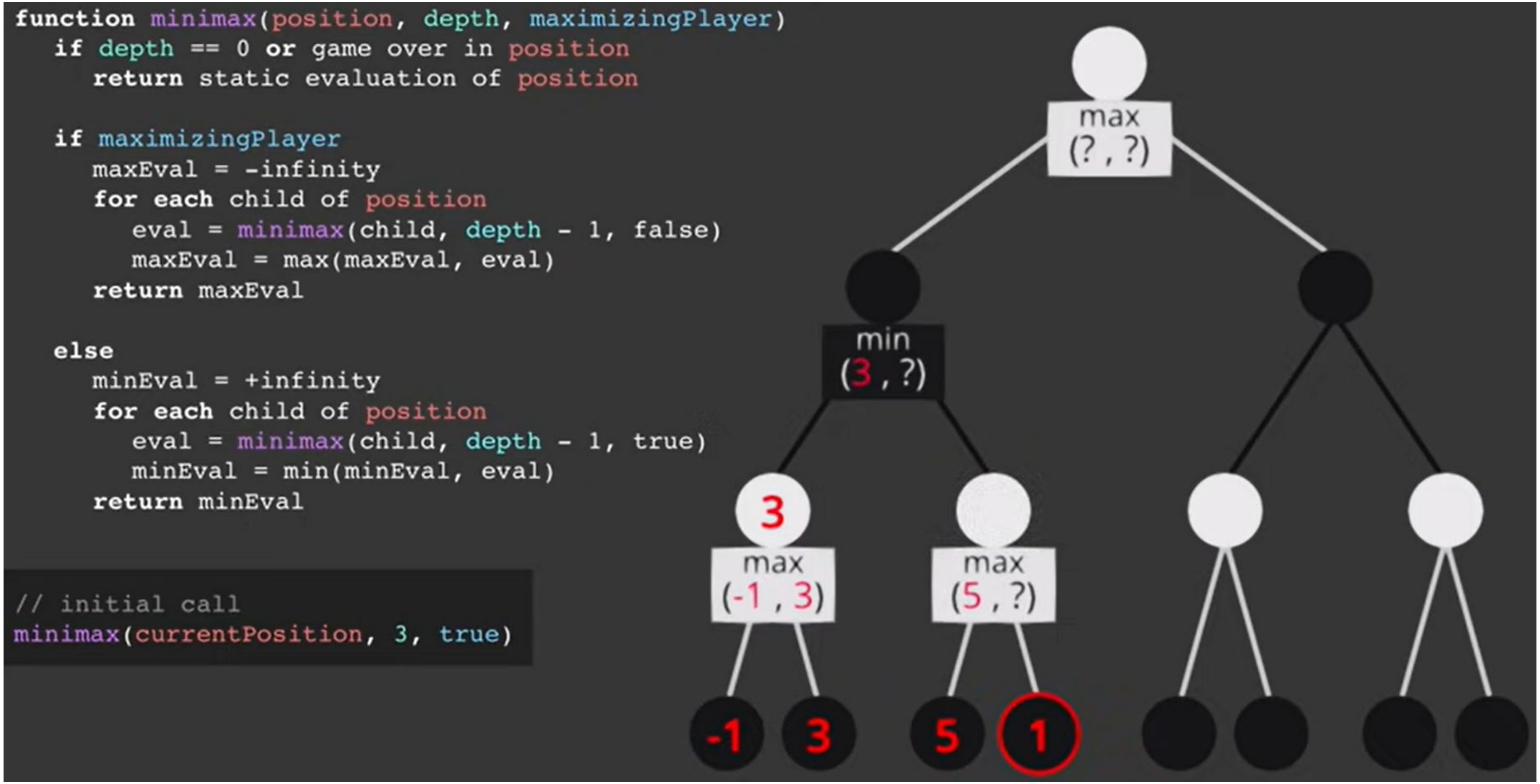


Step 4: Search!! (the abyss...) 1/2

- **Understand** and implement (start with understanding minmax) alpha-beta
- Relatively easy to code, not trivial to integrate in your data structure. Absolutely a **nightmare to debug** (**recursive**). This is where typically you start showing **signs of madness** and people tend to keep you away...
- Most give up here..
- You must sort the legal/pseudo legal moves by pre-score (the smartest the pre-score, the better). Sorting is slow...some sort just some moves..
- **Massive** effect on performances



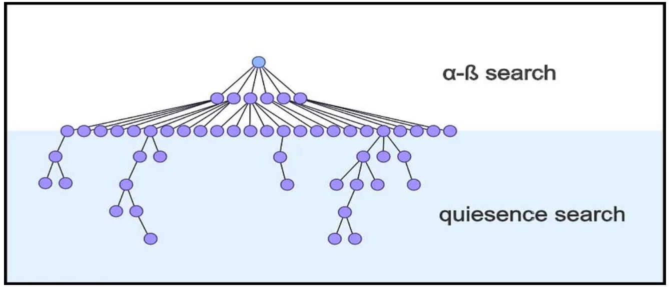
Step 4: Search!! (minimax the basics: pseudo code) 2/2



```
function minimax(position, depth, maximizingPlayer)
   if depth == 0 or game over in position
      return static evaluation of position

   if maximizingPlayer
      maxEval = -infinity
      for each child of position
         eval = minimax(child, depth - 1, false)
         maxEval = max(maxEval, eval)
      return maxEval

   else
      minEval = +infinity
      for each child of position
         eval = minimax(child, depth - 1, true)
         minEval = min(minEval, eval)
      return minEval


// initial call
minimax(currentPosition, 3, true)
```

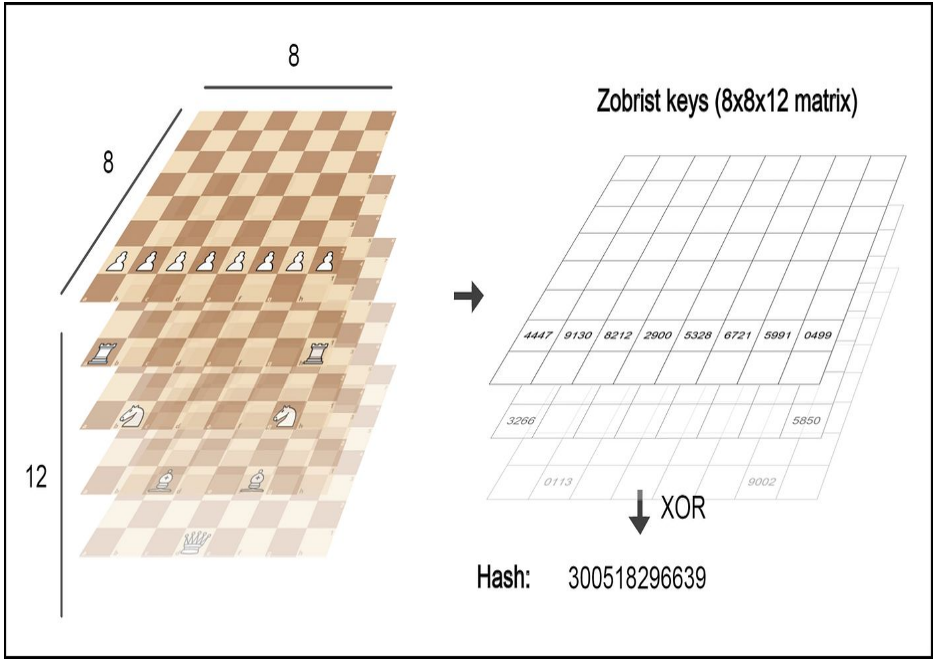
Step 4: Search!! Avoid the horizon effect: quiescence!

- Fixed depth may (and in fact it does) lead to **horizon effect**: just after the depth you get mated of loose your queen…
- Humans also fail due to horizon effect: “ok..I do this, he does that, I do this, he does that, I do this….”.. And then you just overlook the corridor mate…
- Quiescence is the answer: just Re-run search at the end of search BUT feed it with just captures and checks (yes..you have to re-run moves generator….)



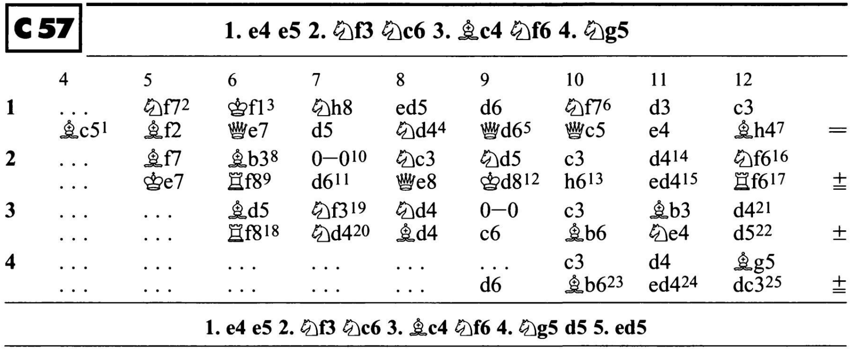
Step 5: Avoid rework: Fight transposition!

- What is a transposition in chess? It’s the **same position reached with different moves order**. For example: 1.e4, e5 2.Nf3 and 1.Nf3, e5 2.e4
- We “only” have long integers in computers..
- Find a function that maps a given chess position to an integer (similar position to generate very different numbers)
- **Zobrist keys**! A XOR festival!
- Why **XOR**? **Incremental**,**Fast, reversible**..
- We create a hashtable indexed with zobrist key of the position and we store the evaluation, among other thing. If we have seen the position before (same Zobriest key) we get the eval from there..



Step 6: other....

- GUI..interface with existing GUIs
- Implement a command line version of your program
- Develop an opening book (not easy..can be very effective in chess matches against other engines)
- Test test test.....



C 57 **1. e4 e5 2. ♘f3 ♘c6 3. ♗c4 ♘f6 4. ♘g5**

	4	5	6	7	8	9	10	11	12	
1	. . . ♗c5[1]	♘f7[2] ♗f2	♔f1[3] ♕e7	♘h8 d5	ed5 ♘d4[4]	d6 ♕d6[5]	♘f7[6] ♕c5	d3 e4	c3 ♗h4[7]	=
2	. . . . . .	♗f7 ♔e7	♗b3[8] ♖f8[9]	0—0[10] d6[11]	♘c3 ♕e8	♘d5 ♔d8[12]	c3 h6[13]	d4[14] ed4[15]	♘f6[16] ♖f6[17]	⩲
3	. . . . . .	. . . . . .	♗d5 ♖f8[18]	♘f3[19] ♘d4[20]	♘d4 ♗d4	0—0 c6	c3 ♗b6	♗b3 ♘e4	d4[21] d5[22]	±
4	. . . . . .	. . . . . .	. . . . . .	. . . . . .	. . . . . .	. . . d6	c3 ♗b6[23]	d4 ed4[24]	♗g5 dc3[25]	⩲

1. e4 e5 2. ♘f3 ♘c6 3. ♗c4 ♘f6 4. ♘g5 d5 5. ed5

References:

- https://www.chessprogramming.org/Main_Page
- Many youtube channels..
  - https://www.youtube.com/@chessprogramming591
  - https://www.youtube.com/@sentdex
  - https://www.youtube.com/@BlueFeverSoft
  - ……many others…
- Study TSCP (Tom’s Simple Chess Program)
- Be very very patient….

Thank you for your time and attention 🙂