



COLUMNISTS

## ChessBase Cafe

Steve Lopez

### Fritz 9 Search Depths

The basis for my column this month came from an email I received from a guy named Ken. His question is interesting and it is one that is commonly asked:

*Fritz 9 question: I bought Chessmaster 10th edition and it is a great program. However, it seems a little odd (and to be a pain) waiting for the computer to make its move. Does Fritz have this same concept or does it just take a turn? I am a little impatient and just want to play. I think the computer is much smarter than I am, and should be able to just make a simple move... Anyway, I realize once you start playing chess it doesn't matter which program you use. Fritz seems to be the more professional of the two. Chessmaster, while a great program, seems to be more of a toy than a serious chess engine. Any comments? Will I have the same trouble with Fritz or should I upgrade? Chessmaster has a force move button, but who wants to make a move, hit ctr-F, then make a move, ctr-F – you get the picture.*

The e-mail is a bit troubling to me because it brings up a common misconception among new users of chess software programs: that a program should be able to *instantaneously* find the best move in a given position, and that a failure to do so is somehow a “flaw” in the program. *No chess program will do this.* In fact, there's a simple axiom that every user of a computer chess program should keep in mind:

**A chess program will always give you the best move it finds within the time you allow it to think, and the longer you let a chess program think, the better the move it will find.**

Now you certainly can set a chess program to make its moves instantaneously (or nearly so), but the quality of these moves typically won't be very high. This leads to another common complaint in e-mails and on message boards: “I set my chess program to think for two seconds per move, but it's not playing very well. I find it hard to believe that this is the program that beat grandmaster so-and-so on TV.”

Let's do a little bit of math as a starting point, and then we'll look at a practical example using *Fritz9*. We'll also learn how to read and interpret



what the program is showing in it's on-screen engine pane. To keep the math as simple as possible, let's consider a hypothetical (and certainly mythical) position in which it's White's turn to move. He has twenty-five possible moves in that position and Black has twenty-five possible replies to each of White's possible moves – to which White has twenty-five possible replies to each of Black's and so on.

Here's a bit of computer chess terminology for you. Chessplayers often refer to the number of moves in a game (i.e. "I beat him in thirty-three moves"), but you're actually talking about *move pairs* (White makes a move, Black makes a move in reply, and we term this pair as one "move" when talking about the length of a chess game). When you're talking about computer chess, each time a player moves a piece or pawn, it's known as a "half-move" or, more commonly, a **ply**. So, for example, a game in which Black checkmates White on move thirty-three would be a sixty-six ply ( $33 \times 2$ ) game in computer chess terminology.

In our hypothetical example, in which White has twenty-five possible moves in a position, a chess computer needs to evaluate all twenty-five positions to determine which leads to the best outcome for White. When it looks ahead to Black's replies (twenty-five for each of White's twenty-five candidate moves), it needs to evaluate 625 positions to determine what White's best initial move would be. The farther ahead the program looks, the more positions it must evaluate – and this total number of positions increases **exponentially** with each additional ply it looks at. Here's a sample progression (assuming that each player has exactly twenty-five candidate moves at each ply, no matter what his opponent plays):

- First ply: 25 positions
- Second ply: 625 positions
- Third ply: 15,625 positions
- Fourth ply: 390,625 positions
- Fifth ply: 9,765,625 positions
- Sixth ply: 244,140,625 positions
- Seventh ply: 6,103,515,625 positions
- Eighth ply: 152,587,890,625 positions

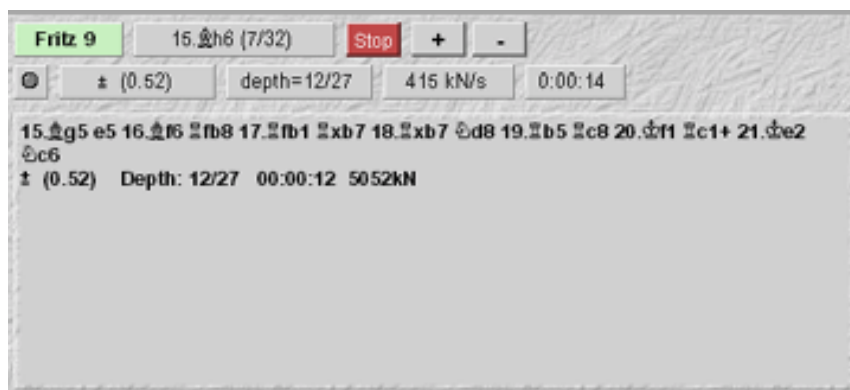
After just four "moves" (in human chess terms) a computer must evaluate over one hundred fifty two *billion* positions to find the best course of play for both sides.

Now every time I write about this exponential progression, I always get an argument from somebody: "Yes, but checks would reduce the number of possible replies for some of those moves" or "Yes, but computers weed out unpromising moves early in the search" or "Using hash tables will speed up the process", etc. That's all true, but those comments miss the point of the exercise. I'm merely illustrating the exponential progression that each ply in the "look-ahead" creates, and demonstrating the fact that even the fastest

computer processors can't easily handle the load.

And that, Ken, is why you have to wait for your computer software program to make a move. The choices are basically twofold: you can let your chess program move instantaneously and live with its dismal play (because it can't see far enough ahead to be able to distinguish a good move from a bad one) or exercise a little patience, secure in the knowledge that the longer you let the program think the better the game you'll get in return.

As a corollary to this discussion, let's take a look at Fritz9's engine analysis pane and I'll explain how to read the information that it provides. The exact position being analyzed doesn't matter – we're just discussing how to read the information:



The green block in the upper left hand corner of the display shows the name of the chess engine that's analyzing the position. In this illustration we're using *Fritz9*. ChessBase offers a number of different chess engines and these are *modular* – in other words, you can “plug in” a variety of different engines using the same interface. This saves you from learning a whole new set of commands and procedures each time you purchase a new chess program.

The next block to the right shows the move currently being considered (in this case it's 15.Bh6) as well as two additional numbers. The number to the right of the slash indicates the number of legal moves in the current board position. Here White has thirty-two legal moves. The number to the left of the slash indicates how many candidate moves the program has looked at so far. Farther down in the display (which we'll discuss in a moment) we see that the program is currently looking twelve plies deep. Within that examination of the twelfth ply, the program is currently considering #7 of the thirty-two legal moves; it's already looked at six prior candidate moves. This gives us a finer idea of how far the program has progressed in its search. We know that the program has completed an eleven ply “look-ahead” and is now just shy of a quarter of the way through its evaluation of the twelfth ply.

The “Stop” button allows you to halt the program's analysis, but note that if

you stop and restart it, the program won't pick up where it left off – it starts the search over from the beginning. The plus and minus buttons allow you to increase and decrease the number of variations displayed in the analysis pane's large box (so that you can see the second-best, third-best, etc. lines of play). However, displaying a large number of variations will slow the analysis process down a bit.

The first box in the second row of buttons is a “dashboard light” which will light up in red whenever the engine's evaluation changes by more than a pawn.

The next box to the right shows the best evaluation of the position that the engine has found. The first character is standard “Informant-style” chess notation for “White is slightly ahead.” (You can find all of these symbols displayed and described in *Fritz*' Help file.) The number in parentheses shows a numerical value for the engine's assessment of the position, and this is given down to 1/100th of a pawn. The overall value is given as a number of pawns (or the equivalent). The number before the dot is the number of whole pawns that one side is ahead, while the number after the dot is the fractional value down to 1/100th of a pawn. A positive value means that White is ahead, while a negative value means that Black is ahead. So in the illustration we see that White is slightly ahead by a value of 52/100ths of a pawn, or just over half a pawn.

The next box shows the overall search depth. The number to the left of the slash indicates how far ahead the program has looked in its “brute force” search. This number tells us the overall depth of the search. In this case we see a “twelve” displayed, meaning that *Fritz9* is currently looking twelve plies deep (six moves for each player). The number to the right of the slash indicates the depth of the engine's “selective search”: variations that start with checks, captures, or other “forcing” lines of play. The selective search value will be much higher than the main value of the overall search, since the forcing nature of these variations allows the program to see much more deeply. Here *Fritz9* is looking twenty-seven plies (or more than thirteen moves) ahead in certain forcing variations.

The next box to the right tells us how fast the program is analyzing positions. This value is given in kilonodes per second (**kN/s**), or thousands of positions per second. *Fritz9* is currently evaluating at 415 kN/s, which means that it's evaluating roughly 415,000 chess positions per second.

The last box shows us the current total time of the search. So *Fritz9* had been analyzing the position for fourteen seconds.

The largest box shows us the best line of play that *Fritz9* has found in its search. Meaning that the program is displaying the moves that should follow assuming best play for both sides. If you watch the analysis pane in action, you'll see the chess program “change its mind” by periodically

displaying different variations in this box. This indicates that the program is changing its opinion of the “best” line of play as it looks deeper into the position and finds better moves and different consequences for both players. We see in the illustration a seven-move variation that starts with White’s 15.Bg5. But why is it a seven-move variation when *Fritz* has seen just six moves ahead in its “brute force” search? This is where selective search mode comes into play: the exchange of rooks in this variation (a forced sequence), as well as the check later in the suggested line, have allowed the program to see a bit farther ahead.

After the variation, we see the chess notation symbol for “White is slightly ahead” as well as what that means in concrete mathematical terms: that White is just over a half-pawn ahead after *Fritz*’ suggested seven-move variation is played. We also see the search depth information repeated. The third grouping shows how long it took the program to arrive at that decision for the best line of play. So it took *Fritz* twelve seconds to determine that the variation displayed is the best sequence of moves for both sides. The last figure shows how many positions were analyzed in arriving at that decision; this figure is again given in **kN**. We see that *Fritz* evaluated 5,052,000 positions in twelve seconds (while it’s not “instantaneous,” it’s certainly pretty dangd impressive!).

As the chess program looks deeper into a position, you’ll see these figures and symbols change. The program will suggest different variations, and you’ll certainly see the evaluations, the elapsed time, and the speed/depth figures change as well. And the longer you let your chess program think, the better the variations it’ll find and display.

All of this goes to show that good chess takes time, even for a chess program running on a machine with a lightning-fast processor. I stated the axiom before and it bears repeating:

**A chess program will always give you the best move it finds within the time you allow it to think, and the longer you let a chess program think the better the move it will find.**

The fact that *Fritz* (or any other chess program) doesn’t move instantaneously isn’t indicative of a “flaw” in the program; instead it shows us that the chess engine is working hard to find the best move in the given position.

Until next month, have fun!

---

© 2006, Steven A. Lopez. All rights reserved.

---

All the ChessBase software described by Steve in this column, as well as many more ChessBase programs, are available in the [ChessCafe Online Catalog](#).

---

***Steve wants your questions!! Send it along and perhaps it will be answered in an upcoming column. Please include your name and country of residence. [Yes, I have a question for Steve!](#)***

---



[TOP OF PAGE](#)



[HOME](#)



[COLUMNS](#)



[LINKS](#)



[ARCHIVES](#)



[ABOUT THE  
CHESS CAFE](#)

[\[ChessCafe Home Page\]](#) [\[Book Review\]](#) [\[Columnists\]](#)

[\[Endgame Study\]](#) [\[The Skittles Room\]](#) [\[Archives\]](#)

[\[Links\]](#) [\[Online Bookstore\]](#) [\[About ChessCafe.com\]](#) [\[Contact Us\]](#)

Copyright 2006 CyberCafes, LLC. All Rights Reserved.

"**The Chess Cafe®**" is a registered trademark of Russell Enterprises, Inc.