投稿用紙作成ツールの機能の説明

2016.8.14
2017.1.2 改訂
（バージョン 0.02 対応）

後々のために、このツールの仕組みをメモで残しておく。

1）Excel で盤面の生成

この部分は特に目新しいことはない。これまで自分で投稿用紙作成用として使用していたものであり、詰工房作品集「アトリエ」を作った際に配布したツールと、基本的に同じである。

とはいえ、未だに Excel で将棋盤面を作ることは一般的になく、要するに文字を 180 度回転する方法が知られていないようある。「MS Office では文字の 180 度の回転はできない」と信じてきっている方もいるので、盤面を作成する部分だけでも公開・配布する意味はあるかと思っている。

鍵となるのは「半角@で始まる日本語フォント」で、これをプラス・マイナス 90 度回転することで、結果として 180 度回転が実現できる。これは Excel を適当に操作しているだけでは見つけるのは難しいし、通常の Excel の解説本などに書かれていることは殆どない。

Excel で盤面を作ることの有利さは、サイズやフォントに自由度があること。Excel の機能の範囲であれば、罫線の種類や着色の変更もできる。

やり方を知っていれば、手作業でもできるが、駒の向きを一つ一つ設定していくのは相当な手間なので、盤面生成だけでもこのツールを使って頂ければいいと思う。

持駒は、柿木ファイルを入力にした場合はちょっとだけ工夫して表示をするが、fmo を入力した場合は、フェアリー駒使用に備えて、そのままの文字列を表示する。

フェアリー駒は、1 文字で表示されなければならない。これは盤面だけでなく棋譜も同じ（棋譜では 1 文字または「成」＋1 文字）。

２）手順の生成（略記しないとき）

ki2 ファイル（fmo ファイル）の内容を書き写しているに等しく、ファイルに書かれた棋譜の意味は解釈していない。

なので、kif ファイルでなく ki2 ファイルを入力に使うようにしている。

（逆に、kif の棋譜は人間が書く棋譜と微妙に違うが、プログラムが棋譜を正しく解釈するのに適している）

３）繰り返し現れる手順の略記

内部の動作ロジックは、手順の文字列に文法圧縮を適用してラベル付き順序木を生成し、出来上がった順序木から表示用文字列を生成している。

文法圧縮のアルゴリズムは多数あるが、最も単純な考え方である Re-Pair 法を採用している。

文法圧縮は本来、最適な結果を生成することが難しく、ランダムに近い文字列から最適（最小サイズ）の圧縮結果を得るアルゴリズムは NP 困難である。しかし、詰将棋の手順の場合は、前後の着手との因果関係が非常に強いため、文法圧縮に使うアルゴリズムの種類にほとんど関係なく、最適に近い圧縮結果が得られる（と予想している）。

内部の処理としては正確には、文法圧縮を適用する前に連長圧縮が可能な場合は連長圧縮を先に適用する。

ランダムな文字列を圧縮する場合は、文法圧縮と連長圧縮を適用する順序で結果が異なってしまうが、一律に連長圧縮を先に適用しても詰将棋の手順に関しては問題ない（問題を生じるケースが、あまり起こりそうにない）。

表示用文字列の生成では、出来上がった木の根から各ノードを再起呼び出しできればアルゴリズムは簡便で済むが、VBA では簡単にスタックオーバーフロしてしまう。仕方ないので、定番の二分木巡回アルゴリズムを使っている。

繰り返し手順の表記では、１回目と２回目以降とで手順の表記を変えるので、根方向からの参照が複数あるノードに１回目に訪問するときに２回目に訪問したときに生成する文字列を生成しておき、ノード訪問回数のフラグと兼ねている。

木の実装は、割とありふれたもの（各ノードは葉方向のノードへの参照を持つインスタンスで、順序を維持するために部分木の根への参照を配列に入れる）。

4）単純送りの略記

単純な馬鋸や単純な歩の連打を略記する機能である。

攻方 1 種類＋受方 1 種類の手が連続して繰り返されていて、位置が等速直線運動であるときに、略記の対象になる。

後で取ってつけた機能で、あまりうまく出来ているとは思えない（気合いが入っていない）ので、オプションとしている。

何がうまくないかというと極端には、玉を一直線にナイトで追う手順も略記してしまい、略記結果は訳の分からない出力になる。

それと、略記された結果の表記が不可逆圧縮になっているのが、今一つ気に入らない（うまい表記が思いつかなかった。なお、繰り返し手順の略記の結果は可逆圧縮であり、機械的に ki2 形式に復元できる）。

5）手数の制限

手数は、32 ビット版 Excel で 1 億 5 千万手、64 ビット版 Excel で 10 億手を上限として設定している。

（注：本稿執筆事典では手元に 64 ビット Office の環境がないため、10 億手は未検証）

これは、Excel の仕様上の制限で、32 ビット版では全体で使用可能なメモリは 2GB であり、これは実装されているメモリに関係なく上限となっている。この中から Excel 本体もメモリを使用する。32 ビット版で安定した動作する条件下で自由に使用できるメモリの上限は、自分が動作テストした限りでは 700MB 程度、1 つの変数（配列やオブジェクト）は 300MB 程度が限度のようである。

本ツールは、2 手 1 組で 1 つのインスタンスを生成して、インスタンス 1 つへの参照が 4 バイトを使用するため、1 億 5 千万手を上限としている。

64 ビット版 Excel では、1 つのインスタンスが使えるメモリの最大が 4GB

とのことなので、同じ理由で 10 億手を上限としている。

ちなみに、1 億 5 千万手のテスト用入力（既掲載）を略記で動作させると、小生の環境（Core i5、SSD）で 2 時間ぐらいの時間が掛かる。手順略記の部分の計算量オーダは、手数 n に対して $n\log n < O(n) < n^2$ のはずだが、実際はファイルを読み込んでメモリ内に格納する時間が支配的であるため、感覚的には処理時間が n に比例しているかのように感じる。

遅い原因の一つは、ファイルを 1 行ずつ読んでいるせいである。VBA ではファイル全部を一度に読み込んだ方が速いのは知っているが、1 億 5 千万手の ki2 ファイルは 800MB あり、32 ビット版 Excel ではメモリが不足する。

（そんなものに対応させる必要はない、というのが普通かも知れない。）

6）その他（メモリリークについて）

このツールを動作させると、メモリリークが発生することがあるが、これはツールの不具合ではないので、一応お断りしておく。

問題は Excel の「GDI オブジェクトが適切に開放されない」という既知の不具合に起因する。

ツールを何度も続けて動かすことは少ないと思われるので、特に対策はしていない。問題が生じたら Excel を再起動して頂きたい。

（終）