2022-01-19版

Cython コード・変更部分

Cython のソースコード　solver.pyx　のうち、赤がCython 化のために追加・変更された部分です。
印刷の都合で、字下げが狂っている可能性もありますので、元のソースファイル(solver.pyx)が正しい字下げになっているので、印刷はあくまで参考ということでご覧ください。

その他に、Cython で無くなり、別の方法で対応している箇所があります。例えば、例外処理。numPy の配列は、Cython(C)の配列にしている。

変数は、型指定して宣言する。

Cython のプログラムからしか呼ばれない関数の宣言は、def → cdef　と変更する。

変更箇所は限定的で、かつ規則的なので、どのように変更すれば良いか把握できると思います。

以下の赤い部分の変更だけで、実行速度が 100 倍(リミッターを外すと 130 倍)になった。
数値計算、とくに整数演算・論理演算が中心のプログラムは、100倍程度の速度向上が期待できるでしょう。

solver.pyx

```
 1  ##  NP(Number Place) Solver Module  [Cython version]
 2  ##
 3  ##  This Module's main function is solve(bd).
 4  ##
 5  ##(c) 2019-2021  FUJIWARA Hirofumi, Knowledge Engineering Center, Time Intermedia, Inc.
 6  ## This code is licensed under MIT license (see LICENSE.txt for details)
 7  ##
 8
 9  #distutils: language=c++
10
11  #cython: language_level=3
12  #cython: profile=False
13  #cython: boundscheck=False
14  #cython: wraparound=False
15
16  import  parameter
17  import  numpy as np
18  import  sys
19  import  NP
20
21  DEF EXCEPTION_CODE = -99
```

```
22
23  cdef int  SIZE = parameter.SIZE
24  cdef int  SUBSIZE = parameter.SUBSIZE
25
26  cdef int[:,:]   board
27  cdef char[:,:,:] candidate
28
29  def solve( bd ):
30
31      initialize()
32
33      blanks = setProblem(bd)
34      if blanks < 0:
35            return -1
36
37      if checkLoop() >= 0:
38            return blankCount()
39
40      return -1
41
42
43  cdef initialize():
44      global candidate, board
45
46      board = np.zeros((SIZE,SIZE),dtype=np.int32)
47      candidate = np.ones((SIZE,SIZE,SIZE+1),dtype=np.int8)
48
49
50  cdef int setProblem( bd ) except? EXCEPTION_CODE:
51      cdef  int   r, c
52
53      for r in range(SIZE):
54            for c in range(SIZE):
55                  if bd[r][c]:
56                        if not setValue( r, c, bd[r][c] ):
57                              return EXCEPTION_CODE
58      return blankCount()
59
60
61  cdef bint checkLoop() except? EXCEPTION_CODE:
62      cdef bint   changed, ret
63      cdef int    r, c
64
65      changed = True
66      while changed:
67            changed = False
68
69            for r in range(0,SIZE,SUBSIZE):     # 3x3 Blocok Check
70                  for c in range(0,SIZE,SUBSIZE):
71                        ret = checkBlock(c,r)
72                        if ret == EXCEPTION_CODE:
73                              return EXCEPTION_CODE
74                        if ret:
```

```
75                                changed = True
76            if changed:
77                  continue
78
79            for r in range(SIZE):         # HLine check
80                  ret = checkHline(r)
81                  if ret == EXCEPTION_CODE:
82                        return EXCEPTION_CODE
83                  if ret:
84                        changed = True
85                  if changed:
86                        continue
87
88            for c in range(SIZE):         # VLine check
89                  ret = checkVline(c)
90                  if ret == EXCEPTION_CODE:
91                        return EXCEPTION_CODE
92                  if ret:
93                        changed = True
94            if changed:
95                        continue
96
97            for r in range(SIZE):         # Cell check
98                  for c in range(SIZE):
99                        ret = checkCell(c,r)
100                       if ret == EXCEPTION_CODE:
101                             return EXCEPTION_CODE
102                       if ret:
103                             changed = True
104
105           if blankCount() == 0:
106                 break
107
108     return changed
109
110 ## --------------------     set value   --------------------
111
112 cdef bint setValue( int r, int c, int v ) except? False:
113     cdef int i, n, r0, c0
114
115     if not candidate[r][c][v]:
116           return False
117
118     if board[r][c]:
119           if board[r][c] != v:
120                 return False
121           return True
122
123     board[r][c] = v
124     for n in range(1,SIZE+1):
125           candidate[r][c][n] = False
126
127     r0 = (r//SUBSIZE)*SUBSIZE
```

```
128     c0 = (c//SUBSIZE)*SUBSIZE
129
130     for i in range(SIZE):
131           candidate[r][i][v] = False
132           candidate[i][c][v] = False
133           candidate[r0+(i//SUBSIZE)][c0+i%SUBSIZE][v] = False
134
135     return True
136
137 ## --------------------     get value   --------------------
138
139 def getAnswer():
140     ans = np.zeros((SIZE,SIZE)).astype(int)
141     NP.copyBoard( board, ans )
142     return ans
143
144 def getValue( int r, int c ):
145     return board[r][c]
146
147 def getCandidate( int r, int c ):
148     return candidate[r][c]
149
150 ## --------------------     print   --------------------
151
152 def printCandidate():
153     cdef int r, c, n
154
155     print("Solver.candidate:")
156
157     for r in range(SIZE):
158           for c in range(SIZE):
159                 for n in range(1,SIZE+1):
160                       if candidate[r][c][n]:
161                             h = n
162                       else:
163                             h = 0
164                       if h!=0:
165                             print(h,end='')
166                       else:
167                             print('-',end='')
168                 print(' ',end='')
169     print()
170
171
172 def printBoard():
173     sys.stdeerr.write("Solver.board:\n")
174     NP.printBoard(sys.stderr,board)
175
176 ## ------------   check box/line/cell & set   ----------------
177
178 cdef bint checkBlock( int c0, int r0 ) except? EXCEPTION_CODE:
179     cdef:
180     int   n, cnt, col, row, r, c, can
```

```
181       bint   changed, exist
182
183       changed = False
184       for n in range(1,SIZE+1):
185             exist = False
186             cnt = 0
187             col = 0
188             row = 0
189             for r in range(r0,r0+SUBSIZE):
190                   for c in range(c0,c0+SUBSIZE):
191                         if board[r][c] == n:
192                               exist = True
193                               break
194                         if candidate[r][c][n]:
195                               cnt += 1
196                               col = c
197                               row = r
198                         if exist:
199                               break
200
201             if not exist:
202                   if cnt == 1:
203                         if not setValue( row, col, n ):
204                               return EXCEPTION_CODE
205                         changed = True
206                   elif cnt == 0:
207                         return EXCEPTION_CODE
208
209       return changed
210
211
212   cdef bint checkHline( int r ) except? EXCEPTION_CODE:
213       cdef:
214       int   n, cnt, col, c
215       bint   changed, exist
216
217       changed = False
218       for n in range(1,SIZE+1):
219             exist = False
220             cnt = 0
221             col = 0
222             for c in range(SIZE):
223                   if board[r][c] == n:
224                         exist = True
225                   if candidate[r][c][n]:
226                         cnt += 1
227                         col = c
228
229       if not exist:
230                   if cnt == 1:
231                         setValue( r, col, n )
232                         changed = True
233                   elif cnt == 0:
```

```
                        return EXCEPTION_CODE

    return changed


cdef bint checkVline( int c ) except? EXCEPTION_CODE:
    cdef:
    int   n, cnt, col, r
    bint   changed, exist

    changed = False
    for n in range(1,SIZE+1):
          exist = False
          cnt = 0
          row = 0
          for r in range(SIZE):
                if board[r][c] == n:
                      exist = True
                if candidate[r][c][n]:
                      cnt += 1
                      row = r

          if not exist:
                if cnt == 1:
                      setValue( row, c, n )
                      changed = True
                elif cnt == 0:
                      return EXCEPTION_CODE

    return changed


cdef bint checkCell( int c, int r ) except? EXCEPTION_CODE:
    cdef int    n, cnt, v

    if board[r][c]:
    return False

    cnt = 0
    v = 0
    for n in range(1,SIZE+1):
          if candidate[r][c][n]:
                cnt += 1
                v = n

    if cnt == 1:
          setValue( r, c, v )
          return True
    elif cnt == 0:
          return EXCEPTION_CODE

    return False
```

```
287  ## -------------------- blank count     --------------------
288
289  cdef int blankCount():
290      cdef int cnt, r, c
291
292      cnt = 0
293      for r in range(SIZE):
294            for c in range(SIZE):
295                  if not board[r][c]:
296                        cnt += 1
297
298      return cnt
```