JAPLA研究会資料 2008/7/26

Ｊによる迷路の木の探索
－木構造を基本から考えてみる－

西川　利男

１．木構造の表現

ここ数回パズルの木の探索のプログラムについて報告してきた[1,2,3]。

あらためて木構造とは何かをコンピュータ科学の教科書を見てみると、通常は、Pascal、最近ではＣによりポインタ機能を用いてつぎのようになされる[4]。

まず、

```
type
    branch = ^tree ;
    tree = record
             low :  branch ;
             high : branch ;
             data : integer ;
           end ;
```

のようにポインタを使ってデータ型treeを定義し、値を

```
var :
    node : branch ;
```

とした上で、実行プログラムとしては

```
new(node) ;
node^.low := A ;
node^.high := B ;
node^.data := V ;
```

のようにプログラミングされる。つまり、システム関数newにより初期値nodeを生成して、ポインタ使ってリンクを張り、次々と木構造を作りあげていく。これはプログラム実行にあたって、動的に木構造が作られアルゴリズムとしもエレガントである。しかし、Ｊなどで実現するには、必ずしも実用的ではない。

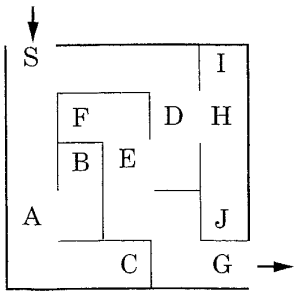
実はＪでもこれに倣って、木構造を作ってパズルの問題を解決しようと、いろいろやってみたがうまくいかない。いっそのこと木構造の考え方を基本に戻って、その言語にあった方法で行えば良いのではないか、との反省に立っての筆者の試みである。

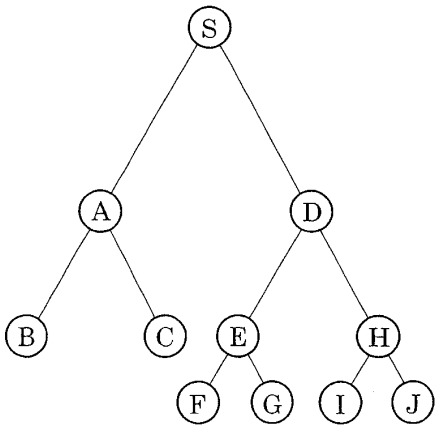
今回はパズルというよりもっと簡単な次のような迷路(maze)脱出の問題[5]を例としてＪによる木構造の探索を検討した。

[1] 西川利男「Ｊによる８クィーンーその１」JAPLA研究会資料2008/4/26
[2] 西川利男「Ｊによる８クィーンーその２、解の探索プログラム」JAPLA研究会資料2008/5/24
[3] 西川利男「ＪによるWhite&Blackパズルとその探索プログラム」JAPLA研究会資料2008/6/28
[4] L.Mazlack, “ Structured Problem Soving With Pascal , p.311 Hold Rinehold & Winston(1983).
[5] 西原清一「Ｃで学ぶデータ構造とアルゴリズム」p.43, オーム社(2008).

２．迷路と探索の木

つぎのような迷路の脱出を例にとる。これに対する検索の木は次のようになる[5]。





ここでは最も簡単に、探索の木の枝に対してＪで文字列のボックスで表す。枝の終点は文字(_)で示す。これは単にリンクの状態を示したものだがこれで充分である。

```
   DA
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|RS|SA|SD|AB|AC|B_|C_|DE|EF|EG|F_|G_|DH|HI|HJ|I_|J_|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

訪問した枝をDMとして、ルート(R)からリンクをたどり、すでに訪問した枝は取り除きこれから訪問すべき枝をDNとして次々と更新していく。かつ枝の終点(_)に出会ったときはバックトラックで戻る。そしてゴール(G)に達したとき解が得られる。
途中経過も示したプログラムの実行は次のようになる。

```
     tra ''
i=0 R
+---+--+
|DM:|RS|
+---+--+
+---+--+
|DN:|RS|
+---+--+

S
i=1 S
+---+--+--+
|DM:|SA|RS|
+---+--+--+
+---+--+--+
|DN:|SA|SD|
+---+--+--+
```

A

i=2 A

```
+---+--+--+--+
|DM:|AB|SA|RS|
+---+--+--+--+
+---+--+--+--+
|DN:|AB|AC|SD|
+---+--+--+--+
```

B

i=3 B

```
+---+--+--+--+--+
|DM:|B_|AB|SA|RS|
+---+--+--+--+--+
+---+--+--+--+
|DN:|B_|AC|SD|
+---+--+--+--+
```

_

back_truck:

```
+----+--+--+
|DNX:|AC|SD|
+----+--+--+
```

i=4 C

```
+---+--+--+--+--+--+
|DM:|C_|B_|AB|SA|RS|
+---+--+--+--+--+--+
+---+--+--+--+
|DN:|C_|AC|SD|
+---+--+--+--+
```

_

back_truck:

```
+----+--+
|DNX:|SD|
+----+--+
```

i=5 D

```
+---+--+--+--+--+--+--+
|DM:|DE|C_|B_|AB|SA|RS|
+---+--+--+--+--+--+--+
+---+--+--+--+--+
|DN:|DE|DH|AC|SD|
+---+--+--+--+--+
```

E

i=6 E

```
+---+--+--+--+--+--+--+--+
|DM:|EF|DE|C_|B_|AB|SA|RS|
+---+--+--+--+--+--+--+--+
+---+--+--+--+--+--+
|DN:|EF|EG|DH|AC|SD|
+---+--+--+--+--+--+
```

F

```
i=7 F
+---+--+--+--+--+--+--+--+
|DM:|F_|EF|DE|C_|B_|AB|SA|RS|
+---+--+--+--+--+--+--+--+--+
+---+--+--+--+--+--+--+
|DN:|F_|EG|DH|AC|SD|
+---+--+--+--+--+--+--+


_
back_truck:
+----+--+--+--+--+--+
|DNX:|EG|DH|AC|SD|
+----+--+--+--+--+--+
i=8 G
Route to Goal:
+--+--+--+--+--+--+--+--+--+
|R |S |A |B |C |D |E |F |G |
+--+--+--+--+--+--+--+--+--+
```

プログラムリスト

```
NB. tree maze traversal
DA =:
'RS';'SA';'SD';'AB';'AC';'B_';'C_';'DE';'EF';'EG';'F_';'G_';'DH';'HI';'HJ';'I
_';'J_'

tra =: 3 : 0
i =. 0
DM =. ''
DMA =. ''
DN =. ''
D =. 'R'
label_LOOP.
  wr 'i=', (":i), ' ', D
  if. 'G' = D do. goto_FIN. end.
  DM =. (((> {. L:0 DA) i. D){DA), DM
  wr 'DM:';DM
  DN =. ( ((D = > {. L:0 DA)#i.#DA){DA), DN
  DN =. DN -. DMA
  wr 'DN:';DN
  YN =. rd 1
  if. 0<#YN do. return. end.
  wr D =. {: > {. DN
  if. '_' = D do.
    wr 'back_truck:'
    DN =. }. DN
NB.     wr 'DN:';DN
    DX =. {. > 0{DM
```

```
    DNX =. (-. > DX e. L:0 DN)#DN
    wr 'DNX:';DNX
    D =. {: > {. DNX
  end.
  DMA =. 0{DM
  i =. i + 1
goto_LOOP.
label_FIN.
  wr 'Route to Goal:'
  |. ('G'; {. L:0 DM) , L:0 ' '
)
```