Algorithm 1 Deep df-pn (part I)

```
1: // At the root
2: procedure DEEPDFPN(r)
3:     r.φ = ∞; r.δ = ∞;
4:     MID(r);
5: end procedure
6:
7: // Exploration of node n
8: procedure MID(n)
9:     // 1. Look up transposition table
10:    LookUpTranspositionTable(n,φ,δ);
11:    if n.φ ≤ φ || n.δ ≤ δ then
12:        n.φ = φ; n.δ = δ;
13:        return ;
14:    end if
15:
16:    // 2. Generation of legal moves
17:    if n is a terminal node then
18:        if (n is an AND node && Eval(n) = true) ||
19:        (n is an OR node && Eval(n) = false) then
20:            n.φ = ∞; n.δ = 0;
21:        else
22:            n.φ = 0; n.δ = ∞;
23:        end if
24:        PutInTranspositonTable(n,n.φ,n.δ);
25:        return ;
26:    end if
27:    GenerateLegalMoves();
28:
29:    // 3. Avoidance of cycle by using transposition table
30:    PutInTranspositonTable(n,n.φ,n.δ);
31:
32:    // 4. Multiple Iterative Deepening
33:    while 1 do
34:        // Stop searching if φ or δ is above or equal to
35:        its threshold
36:        if n.φ ≤ ΔMin(n) || n.δ ≤ ΦSum(n) then
37:            n.φ = ΔMin(n); n.δ = ΦSum(n);
38:            PutInTranspositonTable(n,n.φ,n.δ);
39:            return ;
40:        end if
41:        n_c = SelectChild(n,φ_c,δ_c,δ_2);
42:        n_φ = n_δ + φ_c − ΦSum(n);
43:        n_δ = min(n.φ, δ_2 + 1);
44:        MID(n_c);
45:    end while
46: end procedure
47:
48: // Record into the transposition table
49: procedure PUTINTRANSPOSTIONTABLE(n,φ,δ)
50:    Table[n].φ = φ; Table[n].δ = δ;
51: end procedure
```

Algorithm 2 Deep df-pn (part II)

```
52:  // Look up the transposition table
53:  procedure LOOKUPTRANSPOSTIONTABLE(n,&φ,&δ)
54:      if n is already recorded then
55:          φ =Table[n].φ; δ =Table[n].δ;
56:      else
57:          // In df-pn, φ = 1 and δ = 1
58:          if E = 0 then
59:              φ = 0; δ = 0;
60:          else if D ≤ n.depth then
61:              φ = 1; δ = 1;
62:          else
63:              φ = E^{D−n.depth}; δ = E^{D−n.depth};
64:          end if
65:      end if
66:  end procedure
67:
68:  // Selection of the child
69:  procedure SELECTCHILD(n,&φ_c,&δ_c,&δ_2)
70:      δ_c = ∞; δ_2 = ∞;
71:      for each child node n_child do
72:          LookUpTranspositionTable(n_child,φ,δ);
73:          if δ < δ_c then
74:              n_best = n_child;
75:              δ_2 = δ_c; φ_c = φ; δ_c = δ;
76:          else if δ < δ_2 then
77:              δ_2 = δ;
78:          end if
79:          if φ = ∞ then
80:              return n_best;
81:          end if
82:      end for
83:      return n_best;
84:  end procedure
85:
86:  // Calculation of the minimum δ of all the children
87:  procedure ΔMIN(n)
88:      min = ∞
89:      for each child node n_child do
90:          LookUpTranspositionTable(n_child,φ,δ);
91:          min = min(min, δ);
92:      end for
93:      return min;
94:  end procedure
95:
96:  // Calculation of the summation of φ of all the children
97:  procedure ΦSUM(n)
98:      sum = 0
99:      for each child node n_child do
100:         LookUpTranspositionTable(n_child,φ,δ);
101:         sum = sum + φ;
102:     end for
103:     return sum;
104: end procedure
```