eight queen puzzle

Osada Yuma

2022 年 4 月 3 日

目次

1 エイトクイーンパズル

チェスの盤上にたくさんクイーンを置いて, 各々が各々に取られないようにするパズル.

2 Fortran での実装

カモノハシ本のアルゴリズムをそのまま実装した.

2.1 module

neighbor ポインタに, 左のクイーンを指させてオブジェクト同士の通信で左のやつに取られないような位置へ移動する. チェス盤は `max_row*max_rox` の大きさ.

```
1 module queen_m
2   use, intrinsic :: iso_fortran_env
```

```
3     implicit none
4
5     type queen_t
6        private
7        integer                :: row, col
8        type(queen_t), pointer :: neighbor
9        integer                :: max_row
10     contains
11       procedure, pass :: find_solution => find_solution_q
12       procedure, pass :: can_attack    => can_attack_q
13       procedure, pass :: advance       => advance_q
14       procedure, pass :: print         => print_q
15       final :: destroy_queen
16    end type queen_t
17
18    interface queen_t
19       module procedure :: initialize_left, initialize_q
20    end interface queen_t
21
22  contains
23
24    impure function initialize_left(col, max_row) result(res_q)
25      type(queen_t)       :: res_q
26      integer, intent(in) :: col, max_row
27      res_q%row     = 1
28      res_q%col     = col
29      res_q%max_row = max_row
30      res_q%neighbor => null()
31      return
32    end function initialize_left
33
34    impure function initialize_q(col, max_row, queen) result(res_q)
35      type(queen_t)                    :: res_q
36      integer              , intent(in) :: col, max_row
37      type(queen_t), target, intent(in) :: queen
38      res_q%row     = 1
39      res_q%col     = col
40      res_q%max_row = max_row
41      res_q%neighbor => queen
```

```
42      return
43    end function initialize_q
44
45    subroutine destroy_queen(this)
46      type(queen_t), intent(inout) :: this
47      write(error_unit, '(a, i0, a, i0, a)') "destroyed: (", this%row, ", ", this%col,
        ↪  ")"
48    end subroutine destroy_queen
49
50    impure recursive logical function find_solution_q(this)
51      class(queen_t), intent(inout) :: this
52      do
53         if (.not. associated(this%neighbor)) exit
54         if (.not. this%neighbor%can_attack(this%row, this%col)) exit
55         if (.not. this%advance()) then
56            find_solution_q = .false.
57            return
58         end if
59      end do
60      find_solution_q = .true.
61      return
62    end function find_solution_q
63
64    pure recursive logical function can_attack_q(this, test_row, test_col)
      ↪  result(attackable)
65      class(queen_t), intent(in) :: this
66      integer       , intent(in) :: test_row, test_col
67      integer                    :: column_diff
68      if (this%row == test_row) then
69         attackable = .true.
70         return
71      end if
72
73      column_diff = test_col - this%col
74      if ( this%row + column_diff == test_row .or.&
75           this%row - column_diff == test_row ) then
76         attackable = .true.
77         return
78      end if
```

```
79
80      if (associated(this%neighbor)) then
81         attackable = this%neighbor%can_attack(test_row, test_col)
82      else
83         attackable = .false.
84      end if
85      return
86    end function can_attack_q
87
88    impure recursive logical function advance_q(this)
89      class(queen_t), intent(inout) :: this
90      if (this%row < this%max_row) then
91         this%row = this%row + 1
92         advance_q = this%find_solution()
93         return
94      end if
95
96      if (.not. this%neighbor%advance()) then
97         advance_q = .false.
98         return
99      end if
100     this%row = 1
101     advance_q = this%find_solution()
102     return
103   end function advance_q
104
105   recursive subroutine print_q(this)
106     class(queen_t), intent(in) :: this
107     if (associated(this%neighbor)) then
108        call this%neighbor%print()
109     end if
110     write(output_unit, *) this%row, this%col
111   end subroutine print_q
112
113 end module queen_m
```

2.2 本体

10x10 のチェス盤にクイーンを置く．Fortran には new 演算子みたいなものが (多分) なくて, 変数に新しいオブジェクトを代入してもアドレスが変わらないっぽい? `queen_t` オブジェクトを配列で宣言した.

```
program find_sol
  use, intrinsic :: iso_fortran_env
  use queen_m
  implicit none
  integer, parameter :: num_queen = 10
  integer            :: i
  logical            :: can_find
  type(queen_t)      :: queen(num_queen)

  queen(1) = queen_t(i, num_queen)
  do i = 2, num_queen
     queen(i) = queen_t(i, num_queen, queen(i-1))
     can_find = queen(i)%find_solution()
  end do

  call queen(num_queen)%print()
end program find_sol
```

2.3 実行結果

```
$ ./eight_queen.out
           1           0
           2           2
           6           3
           8           4
          10           5
           4           6
           9           7
           3           8
```

10	5	9
11	7	10