疑似３クイーン問題を解く ScratchJr プログラム

a ScratchJr Program to Solve Quasi-Three Queens Problem

田近　一郎
Ichiro TAJIKA

本稿ではタブレット端末 iPad を利用したオブジェクト指向プログラミング理解のためのビジュアルプログラミングの活用について紹介する．具体的には疑似３クイーン問題を解く ScratchJr プログラムを示す．

In this report we show the advanced understanding for object-oriented programming（OOP）using visual programming. Specifically, we provide a ScratchJr program to solve quasi-three queens problem.

キーワード：プログラミング教育，タブレット端末，iPad，ビジュアルプログラミング，
オブジェクト指向プログラミング，責務分散，メッセージ送受信，人工知能，
探索アルゴリズム，再帰
programming education, tablet device, iPad, visual programming, object-oriented programming,
distributed responsibility, message passing, aritificial intelligence, search algorithm, recursion

１．はじめに

名古屋文理大学では平成23年度以来，iPad を教育に活用するさまざまな試みに取り組んできた[1)-9)]．特に著者らは平成24年度からタブレット端末特有のインターフェースを利用したプログラミング教育の可能性について実践と方法論の両面で研究を進めてきた[10)-16)]．特に，iPad 上で利用可能なビジュアルプログラミングツール ScratchJr[17)] がプログラミング初学者のためのツールとしてだけではなく，デザインパターンやポリモーフィズムなどのオブジェクト指向プログラミング（OOP）の教育にも活用できる可能性について言及した[15), 16)]．本稿ではそれを推し進めて，Java 等の OOP 言語の文法を学んだ初～中級者を対象にしたクラスへの責務分散やメッセージ送信など OOP で重視される内容の教育のために，ScratchJr を活用して３クイーン問題の変型版を解くプログラムを実装する例を紹介する．

本稿の構成は以下の通りである．２章では，N クイーン問題および ScratchJr で実装することを念頭に３クイーン問題を修正した疑似３クイーン問題を定義する．また，疑似３クイーン問題を解くための Java プログラムを，ティモシー・バッドによる８クイーン問題を解くためのオブジェクト指向に基づく Java プログラム[18), 19)]を修正したものとして提示する．３章では，疑似３クイーン問題を解くための ScratchJr プログラムを作成し，このプログラムの機能と構造がともに２章の Java プログラムとほぼ同等であることを示す．これにより Java の基本文法を習得した学生はクラスへの責務分散やメッセージ送信の考え方に基づいたプログラム作成技能に習熟することが容易になると考えられる．４章はまとめである．

２．疑似３クイーン問題とその Java プログラム

２．１　N クイーン問題と疑似３クイーン問題

N クイーン問題は，N × N のチェス盤上の N 個のクイーンの配置に関してどのクイーンも互いに相手を攻撃しない配置を解として探索する問題であり，人工知能分野の代表的な問題の一つである．クイーンの利き筋は列方向、行方向、対角線方向であるので，解となる配置は３個の制約条件：どの２個のクイーンも（1）チェス盤の同じ列にない，（2）同じ行にない，（3）同じ対角線上にない，を満たす．

一方，本稿では疑似３クイーン問題を，３×３のチェ

ス盤上の３個のクイーンのどの２個のクイーンも

・チェス盤の同じ列になく，同時に

・同じ対角線上にない

という制約条件の問題として定義する．この探索問題は従来のＮクイーン問題を定義づける制約条件 (1)，(2)，(3) から，(2) を除いている．この定義により疑似３クイーン問題は，解を持たない３クイーン問題とは異なり解を持ち，その解法プログラムを実行しているときのプログラムの振る舞いに多様性を持たせることができる．また，探索の規模が小さいため SctarchJr のプログラミング言語としての表現能力の範囲内で解法プログラムを SctarchJr 実装することが可能になる．

２．２　疑似３クイーン問題の OOP に基づく Java 実装

Ｎクイーン問題の解法プログラムを手続型のプログラムとして実装する場合，チェス盤上の部分解を構成しているクイーンの配置にマスタープログラムがもう一つのクイーンを配置することを試行する深さ優先探索に基づくアルゴリズムを実装することが多い．各試行では今対象にしているクイーンの配置の候補が他のクイーンから攻撃される配置であるかをテストし，攻撃されない場合はクイーンの配置をこの候補に確定して次の試行に進む．しかし，攻撃される場合は次の配置の候補に１行ずつ下りて移動し，もしＮ行目まで下りても部分解となる配置が見つからない場合は，この試行を中止し1段階前の試行に戻るというバックトラック探索を用いる．

一方，ティモシー・バッドによる OOP に基づくＮクイーン問題の解法プログラムでは，マスタープログラムによるバックトラック探索と同様の探索をマスタープログラムなしでおこなう．ここでＮクイーン問題と疑似Ｎクイーン問題はクイーンの配置に関する制約条件が異なるだけで解法プログラムの構造と動作はほとんど同じであることから，以下ではＮクイーン問題の解法プログラムの代わりに疑似３クイーン問題の解法プログラムの構造と動作を概観する．なお，ティモシー・バッドによるＮクイーン問題の解法プログラムは参考文献[18), 19)]に記載されている．

	１列	２列	３列
１行	Q1 (1, 1)	Q2 (1, 2)	Q3 (1, 3)
２行			
３行			

図1　3×3のチェス盤と盤上の3個のクイーン

図1のように左端から１～３列目のクイーンをそれぞれ Q1，Q2，Q3と書く．プログラムは初期化時に，Q1，Q2，Q3の順にクイーンオブジェクトを１個ずつそれぞれ１行目に生成し，生成の度にチェス盤上のクイーンの配置が部分解であるかをチェックする（図3，図4）．つまり，チェス盤に (Q1)，(Q1，Q2)，(Q1，Q2，Q3) の順にクイーンが配置され，((1, 1))，((1, 1), (1, 2))，((1, 1), (1, 2), (1, 3)) という部分解が順次得られる．カッコ内の数値はチェス盤の各クイーンの座標である．

次の解を探索するため Q3は自らを２～３行と１行ずつ下に移動し，その度に部分解であるかをチェックする．しかし Q3の移動で部分解は得られず，Q3は左に隣接する Q2に移動を依頼する advance メソッドの呼出をおこなう．これを受けて Q2は自らを２～３行と下に移動し，３行目のときに部分解 ((1, 1), (3, 2)) を得る．その直後 Q2は Q3に部分解が得られたことを示す true を返し，Q3は１行目に戻り，（全体）解 ((1, 1), (3, 2), (1, 3)) を得る．

以上を含めすべての解の探索プロセスを下図に示す．図中で実線は解探索のプロセスで部分解または解であることを確認した配置であり，点線は部分解ではないことを確認した配置である．

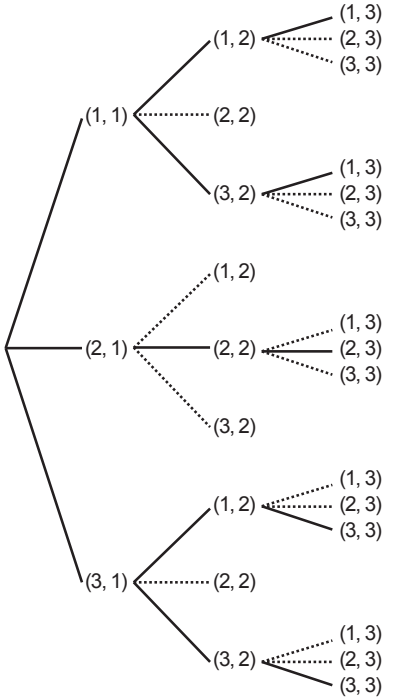


図2　３×３のチェス盤における解の探索プロセス

このように各クイーンは自律的に動作するエージェントとして振る舞い，問題を解くプロセスはエージェントたちがメッセージを送受信しながら部分解から全体解へ

と解を形成するプロセスとみなせる．各クイーンは部分解を構成する配置を自ら探索するので，全体解となる配置を見つけるという責務をクイーン間で分担していると解釈できる．

次に疑似３クイーン問題の解法プログラムにおける「クイーン」クラス,「メイン」クラスの機能を主なメソッドのしくみから概観する．

《クイーンの生成と変数 row，column，neighbor》

「メイン」クラスで３個のクイーンが順次生成される（図4）とき，それぞれのコンストラクタ内で変数 row にクイーンのいる行番号の初期値1，変数 column に各クイーンの不変な列番号 c，変数 neighbor に左に隣接するクイーンオブジェクト n をそれぞれ格納する（図3）．

```
Queen (int c, Queen n) {  //コンストラクタ
  row = 1;       //クイーンを1行目に配置
  column = c;    //クイーンの列はc列目に固定
  neighbor = n;  //左隣りのクイーンnへの参照を保持
}
```

図3　Queen クラスのコンストラクタ

```
public static void main(String[ ] args){//メインメソッド
  ThreeQueenSolver solver = new ThreeQueenSolver();
}
private Queen lastQueen = null;
public ThreeQueenSolver() {
  for (int i = 1; i <= 3; i++) {
    //Q1, Q2, Q3の順に新規クイーンを生成する。
    //ただし、新規クイーンは左隣りのクイーンへの参照を保持
    lastQueen = new Queen(i, lastQueen);
    lastQueen.findSolution();  //部分解であるかをチェック
  }
}
```

図4 「メイン」クラスにおけるクイーンの初期化

《canAttack メソッド》

クイーンは自身より右側の座標（testRow, testColumn）を攻撃可能であれば true を出力する（図5）．攻撃できない場合，左隣りのクイーンの canAttack メソッドを再帰呼出する．再帰呼出されたいずれのクイーンの canAttack メソッドでも座標（testRow, testColumn）を攻撃できないと判定される場合に限り false を出力する（図5）．疑似 N クイーン問題では，ある座標がそれより左側にあるクイーンたちから攻撃される座標であるかをチェックするにはそれらのクイーンたちの対角線上にその座標があるかを計算する。

```
private boolean canAttack(int testRow, int testColumn) {
  int columnDifference = testColumn - column;
  //座標(testRow, testColumn)を攻撃可能ならtrueを出力
  if ((row + columnDifference == testRow) ||
    (row - columnDifference == testRow))
    return true;
  if (neighbor != null)
    //左隣りのクイーンに、攻撃可能であるかのチェックを依頼
    return neighbor.canAttack(testRow, testColumn);
  //どのクイーンも攻撃できないならfalseを出力
  return false;
}
```

図5　Queen クラスの canAttack メソッド

《findSolution メソッド》

findSolution メソッドはこのメソッドを実行するクイーンとその左側のクイーンたちで部分解を構成する場合，true を出力する（図6）．

メソッドの if 文内で呼出する canAttack メソッドの実行により自身の座標が左側のクイーンたちのいずれかから攻撃されることが確定した場合，部分解は得られないので自身を移動するため advance メソッドを呼出する．advance メソッド（図7）呼出では自身およびその左側のクイーンたちの移動と部分解の探索が再帰的におこなわれ，自身とその左側のクイーンたちで構成される部分解が見つかったことを意味する true が返される．それが findSolution メソッドの出力となる．

```
public boolean findSolution() {
//座標(row, column)が左隣りのクイーンたちから攻撃可能なら、
  if (neighbor != null &&
        neighbor.canAttack(row, column))
    advance();  //advanceメソッドで自身の座標を移動
  return true;  //部分解が見つかるとtrueを出力する
}
```

図6　Queen クラスの findSolution メソッド

《advance メソッド》

クイーンは１，２行目にいる場合（図7の①），自身を１行下に移動して部分解を構成しているかを findSolution メソッドを呼出してチェックする．一方，これ以上移動できない3行目にいる場合（図7の②），自身の左隣りのクイーンに移動を依頼する advance メソッド呼出をおこなう．左隣りのクイーンから移動および左隣りのクイーンまでの部分解ができていることを示す true を受け取ると自身を１行目に戻し，部分解を構成しているかを findSolution メソッドを呼出してチェックする．①，②いずれの場合でも findSolution メソッド実行時に部分解ができていない場合，findSolution メソッドを実行して

いるクイーン自身までの部分解が見つかるまで再帰的にadvance メソッドを実行する．このメソッド呼出の連鎖を次の（全体）解が見つかるまで続ける．

なお，Q1は左隣りにクイーンがいないので３行目に達している場合，自らの advance メソッドを実行するとfalse を出力する。その連鎖として同じく３行目に既に達している Q2，Q3も自らの advance メソッドを実行するとfalse を出力し，これにより解の探索が終了していることを判定できる．

```
public boolean advance() {
  if (row < 3) {  //①
    row++;  //自身を次の行に移動し部分解であるかをチェック
    return findSolution();
  } else {                    //②クイーンが3行目にいれば
    if (neighbor != null) {      //左隣りのクイーンの
      if (! neighbor.advance()) {//advanceを呼出
        return false;
      } else {  //左隣りのクイーンを移動後、
        row = 1; //自身を1行目に戻し部分解であるかをチェック
        return findSolution();
      }
    } else {  //左隣りがいないQ1が3行目にいればfalseを出力
      return false;
    }
  }
}
```

図7　Queen クラスの advance メソッド

３．疑似３クイーン問題の ScratchJr による実装

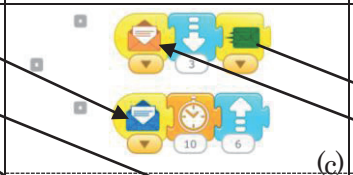
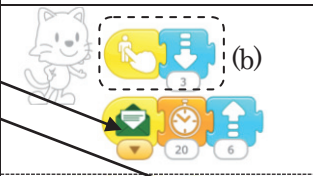
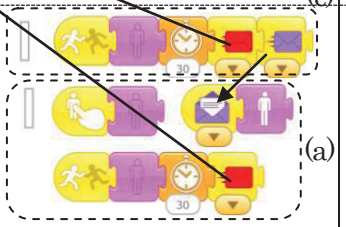
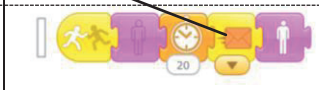
２章の疑似３クイーン問題の解法プログラムとほぼ同等の機能を持ち，実行時にもほぼ同様の振る舞いをするScratchJr によるプログラムを以下で示す．ただし，クイーン間のメッセージ送受信に関するプログラムの実装に ScratchJr の機能の大半を費やしているため，探索に関するプログラムの実装はバックトラック探索としてではなく，単純な深さ優先探索で探索木をすべて探索するプログラムとして実装する．

《チェス盤上での Q1，Q2，Q3の実現と解の表示》

３×３のチェス盤上に３個のクイーン Q1，Q2，Q3を駒のキャラクタとして配置し，キャラクタどうしの当たり判定により解が構成できたか否かをチェックするしくみも同時に実現するため以下のしくみを用いた．

１列目の Q1は，その対角線方向の利き筋に他の駒がいるかどうかを当たり判定で判断するため，Q1本体以外に対角線方向の２～３列目にも四角形を並べて３列にわたる全体を駒とみなす（表1）．同様に Q2も，その対角線方向の利き筋に他の駒がいるかどうかを当たり判定で判断するため，対角線方向の３列目に四角形を並べて全体を駒とする（表1）．なお，Q2の１列目での利き筋は，Q1，Q2が互いに相手を攻撃する配置にある場合，Q1の利き筋と重複することから設ける必要はない．Q3も同様の理由で１～２列目の利き筋は設ける必要はなく１個の「ネコ」キャラクタを駒とする（表1）．図8はScratchJr プログラム実行時に（(2, 1), (2, 2), (2, 3)）という解が見つかったときの画面である．この場合も含め３個のクイーンの配置が解を構成している場合，プログラムは一時的に実行を停止し解を表示する．「ネコ」キャ

表1　疑似３クイーン問題の ScratchJr による解法プログラム

クイーン Q1，Q2，Q3 に対応するキャラクタ	Q1	Q2 と と	Q3 と
advance に対応する ScratchJr プログラム 上段：キャラクタを１行下に移動するプログラム 下段：キャラクタを１行目に戻すプログラム			
左隣りのクイーンの advance メソッドを呼出することに対応する ScratchJr プログラム			
findSolution に対応する ScratchJr プログラム			

ラクタをタップすると次の解の探索を再開する．

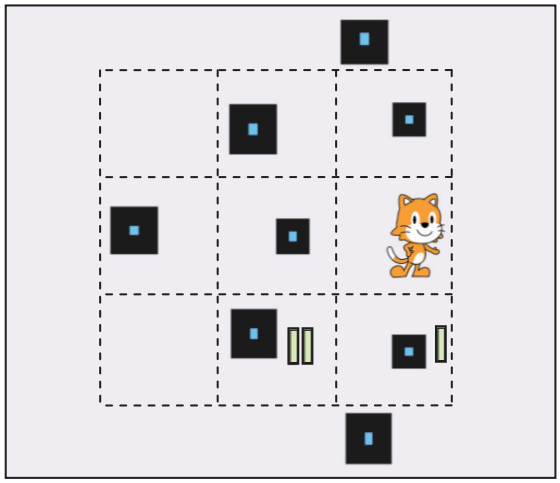


図8 解 ((2, 1), (2, 2), (2, 3)) における Q1，Q2，Q3の配置

《プログラム起動の方法と解探索の進め方について》

プログラムの起動時に３行２列目にある「棒」キャラクタの一つ（表1の (a)，図8の３行２列目の左側の「棒」キャラクタ）をあらかじめタップにより「不可視」にしておく．この理由は後述する．その後，解を探索するために Q3に対応する「ネコ」キャラクタをタップし１行目から２行目に，また２行目から３行目に移動させる（表1の (b)）．「ネコ」キャラクタは３行目に至ると Q2に移動を依頼するためのオレンジ色メッセージを「棒」キャラクタを介して送信する．Q2はオレンジ色メッセージを受け取ると1行下に移動し「ネコ」キャラクタに緑色メッセージを送信する．「ネコ」キャラクタは緑色メッセージを受信し再び1行目に戻る．以上のプロセスを繰り返すことにより解探索が進められる．

《advance メソッドに対応する ScratchJr プログラム》

Q3が自身の advance メソッドにより１行下へ移動する Java プログラムにおけるしくみは，「ネコ」キャラクタをタップして１行下へ移動するという ScratchJr プログラムで実現される．また，Q3が３行目に至ったとき，自身の advance メソッドにより１行目に戻る Java プログラムにおけるしくみは，Q2キャラクタからの緑色メッセージを受信して「ネコ」キャラクタを１行目に戻すという ScratchJr プログラムで実現される．

次に Q2が自身の advance メソッドにより１行下へ移動する Java プログラムにおけるしくみは，「ネコ」キャラクタから「棒」キャラクタを介して送信されたオレンジ色メッセージを受信して Q2キャラクタを1行下へ移動するという ScratchJr プログラムで実現される．また，Q2が３行目に至ったとき，自身の advance メソッドにより１行目に戻る Java プログラムにおけるしくみは，Q1キャラクタからの青色メッセージを受信して Q2キャラクタを１行目に戻すという ScratchJr プログラムで実現される．

最後に Q1が自身の advance メソッドにより１行下へ移動する Java プログラムにおけるしくみは，Q2キャラクタから「棒」キャラクタを介して送信された赤色メッセージを受信して Q1キャラクタを１行下へ移動するという ScratchJr プログラムで実現される．

《findSolution メソッドに対応する ScratchJr プログラム》

Q1と Q2の２個のクイーンには他のキャラクタに当たると点滅を繰り返す命令ブロックを組み込んであり，３個のクイーンが解以外の配置の場合は必ずクイーンのキャラクタどうしが重なり点滅を繰り返す．解を構成する配置では，Q1，Q2，「ネコ」キャラクタは互いに重ならず，その結果点滅せずに解が表示される．

《２個の「棒」キャラクタによる Q2の３行目到達判定》

表1の「棒」キャラクタ (c) は Q2が最初に３行目に至ったことを判定するプログラムである．この「棒」キャラクタは Q2に当たると直ちに「不可視」になる．これは３個のクイーンが ((1, 1), (3, 2), (1, 3)) という解を構成しているときにこの「棒」キャラクタが存在すると，Q2がこのキャラクタと重なり点滅し，解であることを表示する妨げになることによる．「棒」キャラクタ (c) は一定時間の後，Q2が２度目に３行目に至ったことを判定する第２の「棒」キャラクタ (a) に紫色メッセージを送信する．この「棒」キャラクタ (a) も Q2に当たると直ちに「不可視」になる．なお，「棒」キャラクタ (a) は Q2が１度目に３行目に至る時点では「不可視」である必要があるので，プログラム起動時にタップにより「不可視」にしておく．

４．まとめ

本稿では OOP でプログラムを設計する際に指針を与えてくれるメッセージ送受信と責務分散の考え方を包含している疑似３クイーン問題を定義付けし，その Java プログラムを具体的に示した．さらにこの Java プログラムに対応するより簡易な実装として ScratchJr プログラムを示した．

現状の ScratchJr のプログラミング言語としての仕様

ではこれ以上に複雑なアルゴリズムを実装することは困難であるが，OOP に基づくプログラムを簡易にプログラミングする方策として，また Java のようなテキストベースのより煩雑な OOP へと接続する導入のための方策として，ScratchJr のようなビジュアルプログラミング言語によりプログラムを作成することは重要であると考えている．ビジュアルプログラミングによる見通しのよい OOP の例と教育への適用について考察を続けたい．

参考文献

1）本多一彦：「モバイル機器の変遷から情報教育機器としての iPad を考察する」，名古屋文理大学紀要，11，97-104（2011）

2）森博，田近一郎，杉江晶子：「タブレット PC を活用したマルチメディア教育の試み」，名古屋文理大学紀要，12, 97-104（2012）

3）佐原理，大橋平和，長谷川旭，長谷川聡，KAISER Meagan：「タブレット端末による学校教育現場向け多言語情報配信システム」，名古屋文理大学紀要，12, 105-112（2012）

4）長谷川旭，佐原理，尾崎志津子，本多一彦，山住冨也，長谷川聡：「名古屋文理大学における iPad 導入とアクティブラーニング」，モバイル学会研究報告集，7(2), 45-48（2011）

5）長谷川旭，長谷川聡，本多一彦，山住富也，佐原理：「大学教育でのタブレット端末の利用とその効果―iPad を無償配布した名古屋文理大学における学生意識」，コンピュータ＆エデュケーション，31, 70-73,（2011）

6）尾崎志津子：「iPad を活用したオンライン英語多読の導入―名古屋文理大学情報メディア学科における事例―」，コンピュータ＆エデュケーション．; 32, 49-52（2012）

7）長谷川聡：「ソーシャルリーディングとソーシャルラーニング」，現代の図書館，50(2), 114-120（2012）

8）長谷川旭，小橋一秀，山住富也，長谷川聡：「タブレット端末の教育利用と情報インフラ：名古屋文理大の iPad 無償配布と大学図書館」，医学図書館，59(3), 186-191,（2012）

9）斉藤徹，河原潤，高下義弘：「教える！名古屋文理大学」，『iPad で現場を変える !』，日本経済新聞出版，132-147（2011）

10）本多一彦，田近一郎，杉江晶子，森博：「タブレット端末を活用したプログラミング教育」，名古屋文理大学紀要，13, 85-92（2013）

11）田近一郎，本多一彦，杉江晶子，森博：「タブレット端末を活用したプログラミング教育（2)」，名古屋文理大学紀要，14, 75-86（2014）

12）田近一郎，本多一彦，杉江晶子，森博：「タブレット端末を活用したプログラミング教育（3)」，名古屋文理大学紀要，15, 17-27（2015）

13）田近一郎，本多一彦，杉江晶子，森博：「IT 教育のためのプログラミングオンモバイル」，モバイル学会研究報告集，11, 143-146（2015）

14）本多一彦，田近一郎，杉江晶子，森博：「プログラミング・オン・モバイル」，メディア教育シンポジウム（2016/2/6）

15）田近一郎，本多一彦，杉江晶子，森博：「タブレット端末を活用したプログラミング教育（4)」，名古屋文理大学紀要，17, 11-23（2017）

16）田近一郎，本多一彦，杉江晶子，森博：「タブレット端末を活用したプログラミング教育」，情報文化学会第25回全国大会，講演予稿集53-56（2017/10/7）

17）ScratchJr，https://itunes.apple.com/jp/app/scratchjr/id895485086 より2017年7月31日検索

18）ティモシイ・A・バッド：「オブジェクト指向プログラミング入門」，ピアソンエデュケーション 第2版（2002）

19）Timothy A Budd: An Introduction to Object-Oriented Programming Third Edition，http://web.engr.oregonstate.edu/~budd/Books/oopintro3e/info/ReadMe.html より6章2017年10月22日検索

補遺　ScratchJr の命令ブロックの一覧表