卒業研究報告書

題目

テトリスにおけるサポート機能の開発

指導教員

石水 隆　講師

報告者

17-1-037-0028

割石 皓太

近畿大学理工学部情報学科

令和 3 年 2 月 3 日提出

概要

テトリスは，4 つの正方形を組み合わせたテトリミノというブロックを積み重ねていく落ち物パズルゲームである．ルールは，ランダムに落下してくる 1 種類のテトリミノを積んでいき，横一列隙間なく埋めるとその列を消すことができ，一番上までテトリミノが積まれるとゲームオーバーである．落ちてくるテトリミノの次の形状は知ることができ，プレイヤーはテトリミノを左右に移動させ，90 度ごとに回転させることができる．

テトリスは，上級者であれば永久的にプレイできるが，初心者はどこかで詰まり列が積まれていきゲームオーバーになる．そこで，本研究では，初心者でもゲームオーバーにならないようにサポートする機能を開発する．

目次

1. 序論

1.1 テトリスとは

テトリスは，4 つの正方形を組み合わせたテトリミノというブロックを積み重ねていく落ち物パズルゲームである．ルールは，ランダムに落下してくる 1 種類のテトリミノを積んでいき，横一列隙間なく埋めるとその列を消すことができ，一番上までテトリミノが積まれるとゲームオーバーである．落ちてくるテトリミノの次の形状は知ることができ，プレイヤーはテトリミノを左右に移動させ，90 度ごとに回転させることができる．

テトリミノの置き方によっては，複数の段をまとめて消す多段消しができる．多段消しをすると得点が多く得られるため，得点を稼ぐためには多段消しができるようにテトリミノを積んでいく必要がある．しかし，多段消しを狙いすぎるとテトリミノの出現順によっては,いつまでも消すことができずテトリミノが積み上がってしまうため，テトリミノの形から，多段消しを狙うか，即座に消すことを狙うか判断しなければならない．

1.2 テトリスに関する既知の結果

テトリスは得点を競うゲームであり，高得点を取るには多段消しをする必要がある．また，長くゲームを続けるには，積み上がるテトリミノの高さをできるだけ低くしなければならない．評価計算方法を設定してコンピュータにテトリスをプレイさせる全自動テトリスでは，評価計算次第では 25000 個のブロックを落としてもゲームオーバーにはならない[3]．しかし，与えられたテトリミノの順番に対して，最も高い得点が得られる積み方を求める問題やテトリミノの高さが最も低くなる積み方を求める問題は NP 困難であることが Demaine により示されている[4]．NP 困難とは，問題のサイズの多項式時間で解を得ることが難しいとされてる問題のクラスである．すなわち，n 個のテトリミノが落ちてくるときに，n 時間で最も得点が高くなる積み方や，最も高さが低くなる積み方を求めることはできないとされる．

また，テトリミノの出現する順番によっては，永久にプレイすることはできず，必ずゲームオーバーになる．Burgiel は，Z 字型と S 字型のミノが交互に降ってる場合，70000 個以内に必ずゲームオーバーになることを示した[5]．

先行研究である「テトリスにおける AI の開発」は 1 段以上消せる，隙間なく置ける，置かない方が良い場所の 3 パターン候補表示のサポート AI であり，候補場所が重なったり，3 パターン以上の複数の候補が出現する[6]．

1.3 本研究の目的

テトリスは e スポーツの競技種目としても採用されており，注目されている競技で初心者から上級者であるプロまで存在する．上級者は対人戦でなければ半永久的にプレイできるが，初心者はどこかで詰まり列が積まれていきゲームオーバーになる．そこで，初心者でもゲームオーバーにならないようにサポートする機能を開発する．

本研究では Java を用いて，テトリスの初心者を対象としたサポート機能を作成する．初心者と上級者の主な違いは，半永久的にできるかできないかの差である．初心者でも半永久的にプレイできることを目標に，最適なテトリミノの置き場所を指示するサポート機能を開発する．

1.4 本報告書の構成

本報告書の構成は以下の通りである．2 章では本研究で作成したテトリスのサポート機能について述べる．3 章では，結果・考察を述べ，4 章で結論・今後の課題を述べる．

2. 作成したサポート機能について

2.1 テトリスのルール

本節では，テトリスのルールについて説明する．

テトリスは，画面上部から降ってくるテトリミノと呼ばれるブロックを操作し，配置していくゲームである．出現するテトリミノは 7 種類(I，O，S，Z，J，L，T)あり時間と共に落ちてくる．図 1 に各テトリミノの形を示す．プレイヤーはテトリミノを左右へ移動または回転，および即座に落下させることができる．フィールドの一番下またはすでに積まれているテトリミノの上に落下したテトリミノはその位置で固定され，画面上部から新たなテトリミノが出現する．フィールドのある横一列を全てのテトリミノで埋めると，その行のテトリミノは消去され，それより上にあるテトリミノが落下する．フィールドにテトリミノが積み上がり，それ以上テトリミノが置けなくなるとゲームオーバーである．

横一列をテトリミノで全て埋めて消去した時に得点が入る．この時，テトリミノの置き方によっては複数の行を一度に消すことができる．一度に消した行が多いほど高得点が得られるため，一度に複数行を消せるように考えてテトリミノを積む必要がある．ゲームが進むにつれて，テトリミノの落下速度が上がるため，より速い判断が必要となる．

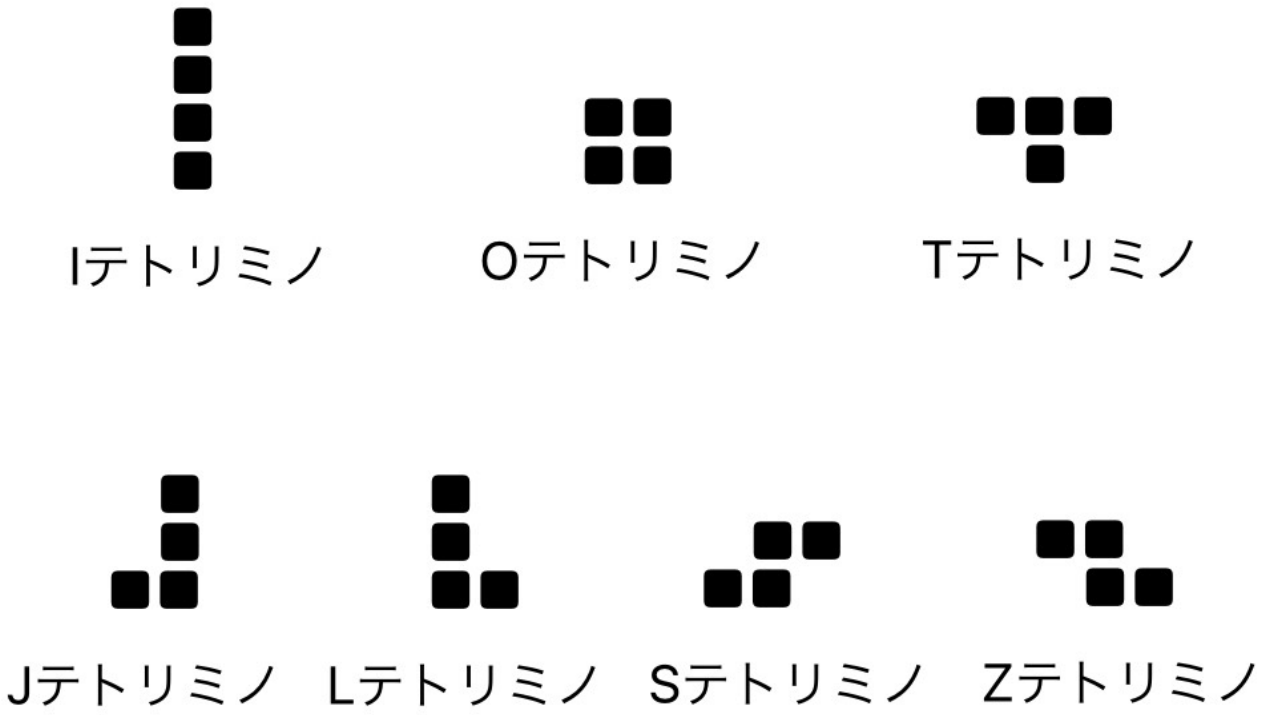

図 1テトリミノの種類

2.2 評価計算方法

半永久的にプレイするためには，列が消せる場合は列を消し，列が消せない場合はできるだけ低い位置にテトリミノを置くという戦略に従ってテトリミノを配置することで，ゲームオーバーになる確率を減らせる．そこで，この戦略に従い，以下の条件を満たすテトリミノの位置に評価を与える．

1. 積んだブロックの高さをなるべく低くする
2. 隙間(■の真下に□)がないようにする
3. 穴(■の左右どちらかに□)を作らないようにする

※■はテトリミノが置かれている状態を表し，□はテトリミノが置かれていない状態を示す．

1 においては，高さがあるとゲームオーバーに近づくので，積んだテトリミノの高さが一番上の時は最低点の 0，一番下の時は 19 点を加点するように高さによって加点を判定する．2 では，隙間の 1 つにつき 1 点減点する．3 は，穴の 1 つにつき 1 点減点する．

2.3 サポート機能の使い方

図 2 起動時に落ちてくるミノが問われる．I，O，S，Z，J，L，T のいずれかを入力すると評価計算結果による最適な位置が図 3 のように*で表示される．

図 2 起動時の画面

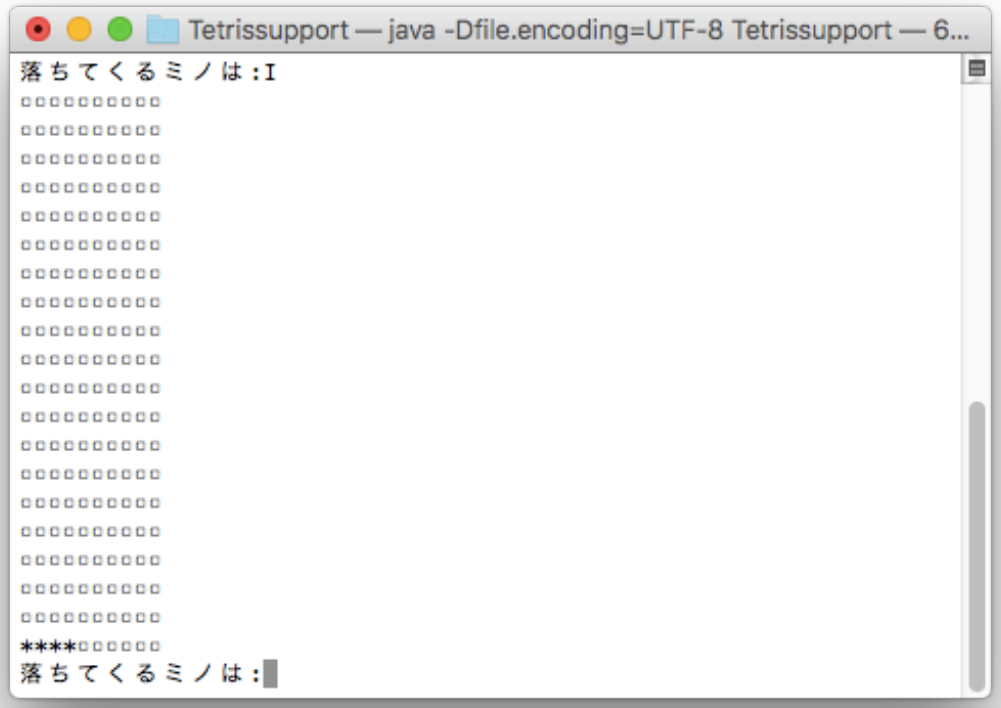

図 3 入力した結果

2.4 プログラム

付録に本研究で作成したサポート機能プログラムのソースを示す. 本研究で作成したプログラムは Tetrissupport クラス 1 つから成る．図 4 に Tetrissupport クラスのクラス図を示す.

subTet メソッドでは現在のフィールドを判定用のフィールドにコピーする．maxTet メソッドでは，現在のフィールドを評価の高いフィールドにコピーする．resTet メソッドは評価の高いフィールドを現在のフィールドにコピーする．calculation メソッドでは，判定用フィールドにあるテトリミノを評価計算する．テトリミノの種類を入力し，テトリミノを置ける場所を左上から検索し，置ける場所を見つけ次第計算して，評価値を比較していき評価が高い方を最適位置として表示させる.

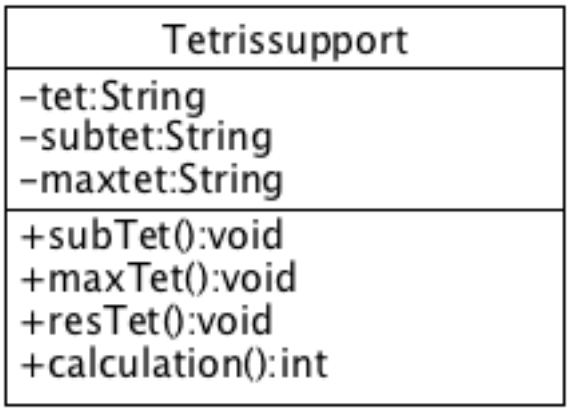

図 4

3. 結果・考察

図 5 にテトリミノを本サポート機能で指示される位置に置くことを繰り返した場合の実行例を示す．図 5 に示されるように段は消されることがあるが，徐々に積まれていった．原因は高さの得点に依存しすぎて，段を消すことよりも，高さが低い方を優先したからであると考えられる．したがって，高さによる加点よりも段を消すことを優先させるように，得点の配分を考えるべきである．

今回は初心者用のサポート機能ということで，できるだけ長くゲームができるように，半永久的にゲームをプレイできることを目指した．半永久プレイは結果的には不可能だったが，熟練者はほぼゲームオーバーすることなくプレイすることができるので，評価計算方法次第ではできるのではないかと考える．例えば，実際のテトリスでは次の落ちてくるミノが表示されるので，そのミノも含んだ評価計算を行うことや，なるべく複数段で消すことを意識したものを用いることで，目標の実現が可能ではないかと考える．

先行研究である「テトリスにおける AI の開発」と比較すると自身のプログラムは途中から用いることができず，サポート機能を用いる場合は，最初からやり直す必要がある．途中から用いるために盤面を自動判別できるようにできるようにする必要があると考える[6].

図 5 サポート機能を用いた結果

4. 結論・今後の課題

本研究ではテトリス初心者用のサポート機能を作成した．ゲームオーバーしない条件を探すのは難しく，評価計算方法の見直しをする必要がある．今後の課題は考察にて考えた評価計算方法を導入したり，初心者に対して，視覚的にわかりやすく色別にミノを表示する機能や，テトリス自体にサポート機能を組み込み，プレイ中に最適位置を表示させたり，ミノを自動判別したりする機能を搭載したい．

謝辞

本論文を進めるにあたり，指導教官である石水隆講師から多大なる御助言を賜りました．この場を借りて厚く御礼申し上げます．

参考文献

[1] 村山要司，楽しく学べる Java ゲーム・アプレット，工学社，(2002).

[2] 長久勝，Java ゲームプログラミング，SB クリエイティブ，(2007).

[3] トモカズ，コンピュータにテトリスをプレイさせる，(2014)，http://85data.world.coocan.jp/02-info-tetris03.html

[4] E.D.Demaine, S.Hohenberger, D.Liben-Nowell, Tetris is Hard, Even to Approximate, Computer Science Vol.2002, No.20 pp,1-56,Cornell University Library,(2002)

[5] H.Burgiel:How to lose at Tetris, The Mathematical Gazette, Vol.81, No.491, pp.194-200, (1997)

[6] 川原翔太，テトリスにおけるＡＩの開発，近畿大学理工学部情報学科　2016 年度卒業論文　(2017)，https://www.info.kindai.ac.jp/~takasi-i/thesis/2016_13-1-037-0113_S_Kawahara_thesis.pdf

付録

本研究で作成したサポート機能のソースを以下に示す.

```
import java.util.Scanner;

public class Tetrissupport{

    private static String[][] tet = new String[20][10];

    private static String[][] subtet = new String[20][10];

    private static String[][] maxtet = new String[20][10];









    public Tetrissupport(){


    }


    /**現在のフィールドを別のフィールドにコピー*/
    public static void subTet(){
         for(int i = 0;i < 20;i++){
             for(int j = 0;j < 10;j++){
                  subtet[i][j] = tet[i][j];
             }
         }
    }

    /**現在のフィールドを評価の高いフィールドにコピー*/
    public static void maxTet(){
         for(int i = 0;i < 20;i++){
             for(int j = 0;j < 10;j++){
                  maxtet[i][j] = tet[i][j];
             }
         }
    }

    /**評価の高いフィールドを現在のフィールドにコピー*/
    public static void resTet(){
         for(int i = 0;i < 20;i++){
             for(int j = 0;j < 10;j++){
                  tet[i][j] = maxtet[i][j];
             }
         }
    }



    /**評価計算*/
    public static int calculation(){
```

```
        int result = 0;

        //ブロックの高さが低ければ高評価(一番上の段で 0, 一番下の段で 19 加点)
        outside:for(int i = 0;i < 20;i++){
            for(int j = 0;j < 10;j++){
                if(subtet[i][j].equals("■")){
                    result += i;
                    break outside;
                }
            }
        }




        //隙間(■の真下に□)があれば減点(1 つにつき 1 減点)
        for(int i = 0;i < 19;i++){
            for(int j = 0;j < 10;j++){
                if(subtet[i+1][j].equals("□") && subtet[i][j].equals("■")){
                    result -= 1;
                }
            }
        }



        //穴(■の左右どちらかに□)を作らないようにする(1 つにつき 1 減点)
        for(int i = 0;i < 20;i++){
            for(int j = 0;j < 9;j++){
                if(subtet[i][j+1].equals("□") && subtet[i][j].equals("■")){
                    result -= 1;
                }
                if(subtet[i][j].equals("□") && subtet[i][j+1].equals("■")){
                    result -= 1;
                }
            }
        }



        return result;
    }







    public static void main(String[] args){
        //ミノを初期化
        for(int i = 0;i < 20;i++){
            for(int j = 0;j < 10;j++){
                tet[i][j] = "□";
                subtet[i][j] = "□";
                maxtet[i][j] = "□";
            }
        }
```

```
        while(true){
            Scanner keyBoardScanner = new Scanner(System.in);
            System.out.print("落ちてくるミノは:");
            String input = keyBoardScanner.nextLine();
            if(input.equals("I") || input.equals("O") || input.equals("S") || input.equals("Z")
               || input.equals("J") || input.equals("L") || input.equals("T")){


                //最適配置（*）を積まれたミノ（■）に変換
                for(int i = 0;i < 20;i++){
                    for(int j = 0;j < 10;j++){
                        if(tet[i][j].equals("*")){
                            tet[i][j] = "■";
                        }
                    }
                }

                //消える列があれば消す
                for(int i = 19;i >= 0;i--){
                    if(tet[i][0].equals("■") && tet[i][1].equals("■") && tet[i][2].equals("■")
                       && tet[i][3].equals("■") && tet[i][4].equals("■")
                       && tet[i][5].equals("■") && tet[i][6].equals("■")
                       && tet[i][7].equals("■") && tet[i][8].equals("■")
                       && tet[i][9].equals("■")){

                        for(int j = 0;j < 10;j++){
                            for(int i2 = i;i2 > 0;i2--){
                                tet[i2][j] = tet[i2-1][j];
                                if(i2 == 1){
                                    tet[0][j] = "□";
                                }
                            }
                        }
                        i = 20;
                    }
                }



                if(input.equals("I")){
                    //全17通り
                    int intMax = -400;//現在の最大評価
                    int judge[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

                    //I90度回転:行1列4(7通り)
                    for(int i = 0;i < 20;i++){
                        for(int j = 0;j < 7;j++){
                            if(tet[i][j].equals("□") && tet[i][j+2].equals("□")
                               && tet[i][j+1].equals("□") && tet[i][j+3].equals("□")
                               && (i != 19 && (tet[i+1][j].equals("■") || tet[i+1][j+2].equals("■")
                                               || tet[i+1][j+1].equals("■") || tet[i+1][j+3].equals("■
"))
                                   || i == 19)){

                                if(judge[j] == 0){
                                    subTet();
```

```
                        subtet[i][j] = "■"; subtet[i][j+2] = "■";
                        subtet[i][j+1] = "■"; subtet[i][j+3] = "■";
                        judge[j] = 1;
                        if(calculation() > intMax){
                            intMax = calculation();
                            maxTet();
                            maxtet[i][j] = "*"; maxtet[i][j+2] = "*";
                            maxtet[i][j+1] = "*"; maxtet[i][j+3] = "*";
                        }
                    }

                }
            }
        }

        //I:行4列1(10通り)
        for(int i = 3;i < 20;i++){
            for(int j = 0;j < 10;j++){
                if(tet[i][j].equals("□") && tet[i-1][j].equals("□")
                   && tet[i-3][j].equals("□") && tet[i-2][j].equals("□")
                   && (i != 19 && tet[i+1][j].equals("■")
                       || i == 19)){

                    if(judge[j+7] == 0){
                        subTet();
                        subtet[i][j] = "■"; subtet[i-1][j] = "■";
                        subtet[i-3][j] = "■"; subtet[i-2][j] = "■";
                        judge[j+7] = 1;
                        if(calculation() > intMax){
                            intMax = calculation();
                            maxTet();
                            maxtet[i][j] = "*"; maxtet[i-1][j] = "*";
                            maxtet[i-3][j] = "*"; maxtet[i-2][j] = "*";
                        }

                    }
                }
            }
        }

        //結果を反映
        resTet();


    }else if(input.equals("O")){
        //全9通り
        int intMax = -400;
        int judge[] = {0,0,0,0,0,0,0,0,0};

        //O:行2列2(9通り)
        for(int i = 1;i < 20;i++){
            for(int j = 0;j < 9;j++){
                if(tet[i][j].equals("□") && tet[i-1][j].equals("□")
                   && tet[i-1][j+1].equals("□") && tet[i][j+1].equals("□")
                   && (i != 19 && (tet[i+1][j].equals("■") || tet[i+1][j].equals("■"))
                       || i == 19)){

                    if(judge[j] == 0){
```

```
                        subTet();
                        subtet[i][j] = "■"; subtet[i-1][j] = "■";
                        subtet[i-1][j+1] = "■"; subtet[i][j+1] = "■";
                        judge[j] = 1;
                        if(calculation() > intMax){
                            intMax = calculation();
                            maxTet();
                            maxtet[i][j] = "*"; maxtet[i-1][j] = "*";
                            maxtet[i-1][j+1] = "*"; maxtet[i][j+1] = "*";
                        }

                    }
                }
            }
        }

        //結果を反映
        resTet();


    }else if(input.equals("S")){
        //全17通り
        int intMax = -400;
        int judge[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

        //S:行2列3(8通り)
        for(int i = 1;i < 20;i++){
            for(int j = 0;j < 8;j++){
                if(tet[i][j].equals("□") && tet[i][j+1].equals("□")
                   && tet[i-1][j+1].equals("□") && tet[i-1][j+2].equals("□")
                   && (i != 19 && (tet[i+1][j].equals("■") || tet[i+1][j+1].equals("■")
                                   || tet[i][j+2].equals("■"))
                       || i == 19)){

                    if(judge[j] == 0){
                        subTet();
                        subtet[i][j] = "■"; subtet[i][j+1] = "■";
                        subtet[i-1][j+1] = "■"; subtet[i-1][j+2] = "■";
                        judge[j] = 1;
                        if(calculation() > intMax){
                            intMax = calculation();
                            maxTet();
                            maxtet[i][j] = "*"; maxtet[i][j+1] = "*";
                            maxtet[i-1][j+1] = "*"; maxtet[i-1][j+2] = "*";
                        }
                    }

                }
            }
        }

        //S90度回転:行3列2(9通り)
        for(int i = 2;i < 20;i++){
            for(int j = 0;j < 9;j++){
                if(tet[i][j+1].equals("□") && tet[i-1][j].equals("□")
                   && tet[i-1][j+1].equals("□") && tet[i-2][j].equals("□")
                   && (i != 19 && (tet[i][j].equals("■") || tet[i+1][j+1].equals("■"))
                       || i == 19)){
```

```
                    if(judge[j+8] == 0){
                        subTet();
                        subtet[i][j+1] = "■"; subtet[i-1][j] = "■";
                        subtet[i-1][j+1] = "■"; subtet[i-2][j] = "■";
                        judge[j+8] = 1;
                        if(calculation() > intMax){
                             intMax = calculation();
                             maxTet();
                             maxtet[i][j+1] = "*"; maxtet[i-1][j] = "*";
                             maxtet[i-1][j+1] = "*"; maxtet[i-2][j] = "*";
                        }
                    }

                }
            }
        }



        //結果を反映
        resTet();



    }else if(input.equals("Z")){
        //全17通り
        int intMax = -400;
        int judge[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

        //Z:行2列3(8通り)
        for(int i = 1;i < 20;i++){
             for(int j = 0;j < 8;j++){
                 if(tet[i-1][j].equals("□") && tet[i][j+1].equals("□")
                    && tet[i-1][j+1].equals("□") && tet[i][j+2].equals("□")
                    && (i != 19 && (tet[i][j].equals("■") || tet[i+1][j+1].equals("■")
                                    || tet[i+1][j+2].equals("■"))
                        || i == 19)){

                    if(judge[j] == 0){
                        subTet();
                        subtet[i-1][j] = "■"; subtet[i][j+1] = "■";
                        subtet[i-1][j+1] = "■"; subtet[i][j+2] = "■";
                        judge[j] = 1;
                        if(calculation() > intMax){
                             intMax = calculation();
                             maxTet();
                             maxtet[i-1][j] = "*"; maxtet[i][j+1] = "*";
                             maxtet[i-1][j+1] = "*"; maxtet[i][j+2] = "*";
                        }
                    }

                }
             }
        }

        //Z90度回転:行3列2(9通り)
        for(int i = 2;i < 20;i++){
```

```
                    for(int j = 0;j < 9;j++){
                        if(tet[i][j].equals("□") && tet[i-1][j].equals("□")
                           && tet[i-1][j+1].equals("□") && tet[i-2][j+1].equals("□")
                           && (i != 19 && (tet[i+1][j].equals("■") || tet[i][j+1].equals("■"))
                                || i == 19)){

                            if(judge[j+8] == 0){
                                subTet();
                                subtet[i][j] = "■"; subtet[i-1][j] = "■";
                                subtet[i-1][j+1] = "■"; subtet[i-2][j+1] = "■";
                                judge[j+8] = 1;
                                if(calculation() > intMax){
                                     intMax = calculation();
                                     maxTet();
                                     maxtet[i][j] = "*"; maxtet[i-1][j] = "*";
                                     maxtet[i-1][j+1] = "*"; maxtet[i-2][j+1] = "*";
                                }
                            }

                        }
                    }
                }

                //結果を反映
                resTet();


            }else if(input.equals("J")){
                //全34通り
                int intMax = -400;
                int judge[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                              ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

                //J右90度回転:行2列3(8通り)
                for(int i = 1;i < 20;i++){
                     for(int j = 0;j < 8;j++){
                         if(tet[i][j].equals("□") && tet[i-1][j].equals("□")
                            && tet[i][j+1].equals("□") && tet[i][j+2].equals("□")
                            && (i != 19 && (tet[i+1][j].equals("■") || tet[i+1][j+1].equals("■")
                                            || tet[i+1][j+2].equals("■"))
                                || i == 19)){

                            if(judge[j] == 0){
                                subTet();
                                subtet[i][j] = "■"; subtet[i-1][j] = "■";
                                subtet[i][j+1] = "■"; subtet[i][j+2] = "■";
                                judge[j] = 1;
                                if(calculation() > intMax){
                                     intMax = calculation();
                                     maxTet();
                                     maxtet[i][j] = "*"; maxtet[i-1][j] = "*";
                                     maxtet[i][j+1] = "*"; maxtet[i][j+2] = "*";
                                }
                            }

                        }
                     }
                }
```

```
//J:行3列2(9通り)
for(int i = 2;i < 20;i++){
     for(int j = 0;j < 9;j++){
         if(tet[i][j].equals("□") && tet[i][j+1].equals("□")
            && tet[i-1][j+1].equals("□") && tet[i-2][j+1].equals("□")
            && (i != 19 && (tet[i+1][j].equals("■") || tet[i+1][j+1].equals("■"))
                 || i == 19)){

            if(judge[j+8] == 0){
                subTet();
                subtet[i][j] = "■"; subtet[i][j+1] = "■";
                subtet[i-1][j+1] = "■"; subtet[i-2][j+1] = "■";
                judge[j+8] = 0;
                if(calculation() > intMax){
                     intMax = calculation();
                     maxTet();
                     maxtet[i][j] = "*"; maxtet[i][j+1] = "*";
                     maxtet[i-1][j+1] = "*"; maxtet[i-2][j+1] = "*";
                }
            }

         }
     }
}

//J90度左回転:行2列3(8通り)
for(int i = 1;i < 20;i++){
     for(int j = 0;j < 8;j++){
         if(tet[i][j+2].equals("□") && tet[i-1][j].equals("□")
            && tet[i-1][j+1].equals("□") && tet[i-1][j+2].equals("□")
            && (i != 19 && (tet[i+1][j+2].equals("■") || tet[i][j].equals("■")
                              || tet[i][j+1].equals("■"))
                || i == 19)){

            if(judge[j+17] == 0){
                subTet();
                subtet[i][j+2] = "■"; subtet[i-1][j] = "■";
                subtet[i-1][j+1] = "■"; subtet[i-1][j+2] = "■";
                judge[j+17] = 1;
                if(calculation() > intMax){
                     intMax = calculation();
                     maxTet();
                     maxtet[i][j+2] = "*"; maxtet[i-1][j] = "*";
                     maxtet[i-1][j+1] = "*"; maxtet[i-1][j+2] = "*";
                }
            }

         }
     }
}

//J180度回転:行3列2(9通り)
for(int i = 2;i < 20;i++){
     for(int j = 0;j < 9;j++){
         if(tet[i][j].equals("□") && tet[i-1][j].equals("□")
            && tet[i-2][j].equals("□") && tet[i-2][j+1].equals("□")
            && (i != 19 && (tet[i+1][j].equals("■") || tet[i-1][j+1].equals("■"))
```

```
                        || i == 19)){

                    if(judge[j+25] == 0){
                        subTet();
                        subtet[i][j] = "■"; subtet[i-1][j] = "■";
                        subtet[i-2][j] = "■"; subtet[i-2][j+1] = "■";
                        judge[j+25] = 1;
                        if(calculation() > intMax){
                            intMax = calculation();
                            maxTet();
                            maxtet[i][j] = "*"; maxtet[i-1][j] = "*";
                            maxtet[i-2][j] = "*"; maxtet[i-2][j+1] = "*";
                        }
                    }

                }
            }
        }

        //結果を反映
        resTet();


    }else if(input.equals("L")){
        //全34通り
        int intMax = -400;
        int judge[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                      ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

        //L90度左回転:行2列3(8通り)
        for(int i = 1;i < 20;i++){
            for(int j = 0;j < 8;j++){
                if(tet[i][j].equals("□") && tet[i-1][j+2].equals("□")
                   && tet[i][j+1].equals("□") && tet[i][j+2].equals("□")
                   && (i != 19 && (tet[i+1][j].equals("■") || tet[i+1][j+1].equals("■")
                                   || tet[i+1][j+2].equals("■"))
                       || i == 19)){

                    if(judge[j] == 0){
                        subTet();
                        subtet[i][j] = "■"; subtet[i-1][j+2] = "■";
                        subtet[i][j+1] = "■"; subtet[i][j+2] = "■";
                        judge[j] = 1;
                        if(calculation() > intMax){
                            intMax = calculation();
                            maxTet();
                            maxtet[i][j] = "*"; maxtet[i-1][j+2] = "*";
                            maxtet[i][j+1] = "*"; maxtet[i][j+2] = "*";
                        }
                    }

                }
            }
        }

        //L:行3列2(9通り)
        for(int i = 2;i < 20;i++){
            for(int j = 0;j < 9;j++){
```

```
                if(tet[i][j].equals("□") && tet[i-1][j].equals("□")
                   && tet[i][j+1].equals("□") && tet[i-2][j].equals("□")
                   && (i != 19 && (tet[i+1][j].equals("■") || tet[i+1][j+1].equals("■"))
                        || i == 19)){

                    if(judge[j+8] == 0){
                        subTet();
                        subtet[i][j] = "■"; subtet[i-1][j] = "■";
                        subtet[i][j+1] = "■"; subtet[i-2][j] = "■";
                        judge[j+8] = 1;
                        if(calculation() > intMax){
                             intMax = calculation();
                             maxTet();
                             maxtet[i][j] = "*"; maxtet[i-1][j] = "*";
                             maxtet[i][j+1] = "*"; maxtet[i-2][j] = "*";
                        }
                    }

                }
            }
        }

        //L90 度右回転:行 2 列 3(8 通り)
        for(int i = 1;i < 20;i++){
             for(int j = 0;j < 8;j++){
                if(tet[i][j].equals("□") && tet[i-1][j].equals("□")
                   && tet[i-1][j+1].equals("□") && tet[i-1][j+2].equals("□")
                   && (i != 19 && (tet[i+1][j].equals("■") || tet[i][j+1].equals("■")
                                   || tet[i][j+2].equals("■"))
                        || i == 19)){

                    if(judge[j+17] == 0){
                        subTet();
                        subtet[i][j] = "■"; subtet[i-1][j] = "■";
                        subtet[i-1][j+1] = "■"; subtet[i-1][j+2] = "■";
                        judge[j+17] = 1;
                        if(calculation() > intMax){
                             intMax = calculation();
                             maxTet();
                             maxtet[i][j] = "*"; maxtet[i-1][j] = "*";
                             maxtet[i-1][j+1] = "*"; maxtet[i-1][j+2] = "*";
                        }
                    }

                }
             }
        }

        //L180 度回転:行 3 列 2(9 通り)
        for(int i = 2;i < 20;i++){
             for(int j = 0;j < 9;j++){
                if(tet[i][j+1].equals("□") && tet[i-2][j].equals("□")
                   && tet[i-1][j+1].equals("□") && tet[i-2][j+1].equals("□")
                   && (i != 19 && (tet[i-1][j].equals("■") || tet[i+1][j+1].equals("■"))
                        || i == 19)){

                    if(judge[j+25] == 0){
                        subTet();
```

```
                        subtet[i][j+1] = "■"; subtet[i-2][j] = "■";
                        subtet[i-1][j+1] = "■"; subtet[i-2][j+1] = "■";
                        judge[j+25] = 1;
                        if(calculation() > intMax){
                            intMax = calculation();
                            maxTet();
                            maxtet[i][j+1] = "*"; maxtet[i-2][j] = "*";
                            maxtet[i-1][j+1] = "*"; maxtet[i-2][j+1] = "*";
                        }
                    }

                }
            }
        }

        //結果を反映
        resTet();


    }else if(input.equals("T")){
        //全34通り
        int intMax = -400;
        int judge[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
                      ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

        //T180度回転:行2列3(8通り)
        for(int i = 1;i < 20;i++){
            for(int j = 0;j < 8;j++){
                if(tet[i][j].equals("□") && tet[i][j+2].equals("□")
                   && tet[i-1][j+1].equals("□") && tet[i][j+1].equals("□")
                   && (i != 19 && (tet[i+1][j].equals("■") || tet[i+1][j+2].equals("■")
                                   || tet[i+1][j+1].equals("■"))
                       || i == 19)){

                    if(judge[j] == 0){
                        subTet();
                        subtet[i][j] = "■"; subtet[i][j+2] = "■";
                        subtet[i-1][j+1] = "■"; subtet[i][j+1] = "■";
                        judge[j] = 1;
                        if(calculation() > intMax){
                            intMax = calculation();
                            maxTet();
                            maxtet[i][j] = "*"; maxtet[i][j+2] = "*";
                            maxtet[i-1][j+1] = "*"; maxtet[i][j+1] = "*";
                        }
                    }

                }
            }
        }

        //T:行2列3(8通り)
        for(int i = 1;i < 20;i++){
            for(int j = 0;j < 8;j++){
                if(tet[i][j+1].equals("□") && tet[i-1][j].equals("□")
                   && tet[i-1][j+1].equals("□") && tet[i-1][j+2].equals("□")
                   && (i != 19 && (tet[i+1][j+1].equals("■") || tet[i][j].equals("■")
                                   || tet[i][j+2].equals("■"))
```

```
                        || i == 19)){

                    if(judge[j+8] == 0){
                        subTet();
                        subtet[i][j+1] = "■"; subtet[i-1][j] = "■";
                        subtet[i-1][j+1] = "■"; subtet[i-1][j+2] = "■";
                        judge[j+8] = 1;
                        if(calculation() > intMax){
                             intMax = calculation();
                             maxTet();
                             maxtet[i][j+1] = "*"; maxtet[i-1][j] = "*";
                             maxtet[i-1][j+1] = "*"; maxtet[i-1][j+2] = "*";
                        }
                    }

                }
            }
        }

        //T90 度左回転:行 3 列 2(9 通り)
        for(int i = 2;i < 20;i++){
             for(int j = 0;j < 9;j++){
                 if(tet[i][j].equals("□") && tet[i-1][j].equals("□")
                    && tet[i-1][j+1].equals("□") && tet[i-2][j].equals("□")
                    && (i != 19 && (tet[i+1][j].equals("■") || tet[i][j+1].equals("■"))
                         || i == 19)){

                    if(judge[j+16] == 0){
                        subTet();
                        subtet[i][j] = "■"; subtet[i-1][j] = "■";
                        subtet[i-1][j+1] = "■"; subtet[i-2][j] = "■";
                        judge[j+16] = 1;
                        if(calculation() > intMax){
                             intMax = calculation();
                             maxTet();
                             maxtet[i][j] = "*"; maxtet[i-1][j] = "*";
                             maxtet[i-1][j+1] = "*"; maxtet[i-2][j] = "*";
                        }
                    }

                }
             }
        }

        //T90 度右回転:行 3 列 2(9 通り)
        for(int i = 2;i < 20;i++){
             for(int j = 0;j < 9;j++){
                 if(tet[i][j+1].equals("□") && tet[i-1][j].equals("□")
                    && tet[i-1][j+1].equals("□") && tet[i-2][j+1].equals("□")
                    && (i != 19 && (tet[i][j].equals("■") || tet[i+1][j+1].equals("■"))
                         || i == 19)){

                    if(judge[j+25] == 0){
                        subTet();
                        subtet[i][j+1] = "■"; subtet[i-1][j] = "■";
                        subtet[i-1][j+1] = "■"; subtet[i-2][j+1] = "■";
                        judge[j+25] = 1;
                        if(calculation() > intMax){
```

```
                                        intMax = calculation();
                                        maxTet();
                                        maxtet[i][j+1] = "*"; maxtet[i-1][j] = "*";
                                        maxtet[i-1][j+1] = "*"; maxtet[i-2][j+1] = "*";
                                    }
                                }
                            }
                        }
                    }

                    //結果を反映
                    resTet();


                }



                //結果を表示
                for(int i = 0;i < 20;i++){
                    for(int j = 0;j < 10;j++){
                        System.out.print(tet[i][j]);
                    }
                    System.out.println();
                }


            }else{
                keyBoardScanner.close();
                break;
            }
        }



    }
}
```