コイン問題

そのプログラム手法の解法

初出典 1996 年
JAVA 改良 2015 年 6 月 3 日
ドキュメント作成 2017 年 1 月 8 日

久保覚

命題

1. 10 円玉と 100 円玉それぞれ 3 枚以上のコインを下図のように並べる。
   ●●●○○○　(n=3 と定義した場合)
2. 隣り合うコイン 2 枚をセットでそのまま並んだコインの延長線上に移動させる。
   以下は n=3 とした場合の移動ルール(・はコインの置いてない場所を明示的に表示)
   ●・・○○○●●　　(左から 2 番目と 3 番目をセットで一番右に移動)
   あるいは
   ●●・・○○●○　　(左から 3 番目と 4 番目をセットで一番右に移動)
   もちろん
   ○○●●●○・・　　(右から 1 番目と 2 番目をセットで一番左に移動)
   も可能。
   ※追加考察の結果
   「移動先は必ず動かすコインの左右どちらかが元のコインに接すること」
   を追加ルールしても構わない(後述)。
3. 「2」のルールに従い、続けて移動を繰り返す。以下は移動例。
   移動 0 回目：●●●●○○○○　(n=4 とした場合)
   移動 1 回目：●・・●○○○○●●
   移動 2 回目：●○○●・・○○●●
   移動 3 回目：●○○●○●○・・●
   移動 4 回目：・・○●○●○●○●
4. 「3」の例のように 10 円と 100 円のコインが 4 枚の場合、4 回の移動で交互にできる。
   以上が 10 円玉と 100 円玉を 4 枚使用したコイン問題として余興として出題できる。
   整理すると、
   1)テーブルにコイン 4 枚ずつを「移動 0 回目」のように並べ
   2)隣り合うコイン 2 枚をセットで回転させることなく、空いている場所に移動させる。
   3)その際移動先のコインと接する場所に移動させること。
   4)「2-3」のルールを 4 回繰り返す。

5)「移動 4 回目」のように連続した交互になれば成功。
というゲームができる。

拡張命題

1. さて、コインが 4 枚ずつの場合は 4 回の移動で交互にできた。
   では、コインが 3 枚ずつの場合はどうか？
   移動 0 回目：●●●○○○
   移動 1 回目：・・●○○○●●
   移動 2 回目：・・●○○・・●○●
   移動 3 回目：・・・・○●○●○●
   3 回でできた。
2. それではコインが 5 枚ずつの場合はどうか？
   移動 0 回目：●●●●●○○○○○
   移動 1 回目：●・・●●○○○○○●●
   移動 2 回目：●○○●●○○・・○●●
   移動 3 回目：●○○●・・○●○○●●
   移動 4 回目：●○○●○●○●○・・●
   移動 5 回目：・・○●○●○●○●○●
   5 回でできた。
3. 以上を一般化してコインが n 枚ずつの場合は n 回でできるか検証のためプログラム化して法則発見の一助とする(ただし n は 3 以上)。
4. JAVA 言語で完成したプログラム。
   プログラムするにあたり、以下の条件を課した。
   1)移動パターンを総当たりして解を求める手法でプログラムすること。
   2)再起処理を使用してロジックを考えること。
   3)コインの枚数を入力できるようにすること。
   4))正解のパターンを表示させること(なければ『無し』を表示)。
5. 当初「コインの延長線上のどこでも移動可能」としたルールでプログラムしたが、総当たりのパターン数が膨大になったため「移動先をコインの隣接のみ」に限定した。しかし、限定しても「n 回で交互になる(しかも最終形は空間が無い)」ことが n=10 までは確認できた。
   結果、余興として「コイン問題－命題 4」のようなルールが成立しゲーム化できた。

プログラムリスト

```
////////////////////////////////////////////
//        coin Change
//
//        coded by S.Kubo       1996 mod.2015
////////////////////////////////////////////
import java.util.Scanner;
import java.util.Date;

//////////////////////
//        enum
//
enum markType{
        BLACK("X"),
        WHITE("O"),
        NULL(".");;
        private final String text;

        private markType( final String text ){ this.text=text; }
        @Override
        public String toString(){ return this.text; }
}


//////////////////////
//coinBase
//
class coinBase{
        markType arry[];
        int num;

//============
//construct
        public coinBase( int n ){
                num = n;
                arry = new markType[num*6];
        }
        public coinBase( coinBase coins ){
                this.num=coins.num;
                this.arry = new markType[this.num*6];
                for( int x=0; x<this.num*6; x++ ){
                        this.arry[x]=coins.arry[x];
                }
        }
//============
//Initialize
        public void init(){
                int cnt=0;
                for( int x=0; x<this.num*2; x++ ){
                        this.arry[cnt++] = markType.NULL;
                }
                for( int x=0; x<this.num; x++ ){
                        this.arry[cnt++] = markType.BLACK;
                }
                for( int x=0; x<this.num; x++ ){
                        this.arry[cnt++] = markType.WHITE;
                }
                for( int x=0; x<this.num*2; x++ ){
                        this.arry[cnt++] = markType.NULL;
                }
        }
```

```
//============
//Print
	public void print(){
		for( int x=0; x<this.num*6; x++ ){
			System.out.print( this.arry[x] );
		}
		System.out.println();
	}
//============
//Is Get 2-Coins
	public boolean isGet( int n ){
		if( n<0 || n>this.num*6-2 ){ return false; }
		if( this.arry[n]==markType.NULL || this.arry[n+1]==markType.NULL ){
			return false;
		}else{
			return true;
		}
	}
//============
//Is Put 2-Coins Null-Case
	public boolean isPutNull( int n ){
		if( n<0 || n>this.num*6-2 ){ return false; }
		if( this.arry[n]==markType.NULL && this.arry[n+1]==markType.NULL ){
			return true;
		}else{
			return false;
		}
	}
//============
//Is Put 2-Coins Touch-Case
	public boolean isPutTouch( int n ){
		if( n<1 || n>this.num*6-3 ){ return false; }
		if( this.arry[n+2]!=markType.NULL || this.arry[n-1]!=markType.NULL ){
			return true;
		}else{
			return false;
		}
	}
//============
//Move 2-Coins
	public void move( int i, int j){
		this.arry[j] = this.arry[i];
		this.arry[j+1] = this.arry[i+1];
		this.arry[i] = this.arry[i+1] = markType.NULL;
	}
//============
//Is Alternate Coins
	public boolean isAlternate(){
		for( int i=0; i< this.num*6; i++ ){
			if( this.arry[i]!=markType.NULL && this.arry[i]!=this.arry[i+1] ){
				for( int j=i; j< i+this.num*2; j+=2 ){
					if( this.arry[j]!=this.arry[i] || this.arry[j+1]!=this.arry[i+1] ){
						return false;
					}
				}
				return true;
			}
		}
		return false;
	}
```

```
//============
//Recursive Coin-Patterns
	public boolean run( int n ){
	//print();
		if( n>=this.num ){
			return isAlternate();
		}
		for( int i=0; i< this.num*6; i++ ){
			if( isGet( i ) ){
				for( int j=0; j<this.num*6; j++ ){
					if( isPutNull( j ) && isPutTouch( j ) ){

						coinBase coins = new coinBase(this);
						coins.move( i, j );

						if( coins.run( n+1 )==true ){
							coins.print();
							return true;
						}
					}
				}
			}
		}
		return false;
	}
}

//////////////////////
//	coinChange Main Process
//
public class coinChange{
	public static void main(String[] args) {
		Scanner in = new Scanner( System.in );
		int maxN;
		do{
			System.out.println( "Please enter the number of Your coins(3-n)" );
			maxN = in.nextInt();
		}while( maxN<3 );
		coinBase coins = new coinBase(maxN);
		coins.init();
		Date start = new Date();
		if( coins.run( 0 )==true ){
			coins.print();
		}else{
			System.out.print( "Can Not" );
			System.out.println();
		}
		Date end = new Date();
		long t=end.getTime() - start.getTime();
		System.out.print( t+"ms" );
	}
}
```

プログラム構造

プログラムは単純である。

1. コインが移動するのに十分な空間(配列を)用意。
   coinBase Class
2. 完成条件「コインの間が空かないで交互になる」まで再帰で総当たりをする。
   isAlternate()　　完成条件。
   run(int n)　　再帰。n は再起の深さ。
3. 移動できるコインを探す。
   つまり「隣り合うコインは空間以外」が移動可能なコイン。
   isGet(int n)　　n は調べる最初の配列番号。
4. 移動場所を探す。条件１「置ける場所は空間」である。
   isPutNull(int n)　　n は調べる最初の配列番号。
5. 移動場所を探す。条件２「置ける場所はコインに隣接」する。
   isPutTouch(int n)　　n は調べる最初の配列番号。

実行結果

・Windows10 core-i7 3.4GHz 32Mem

・n=3～n=5　　処理時間は 6ms、13ms、330ms

・n=6　　処理時間は 23945ms　システム上、解法の表示は下から上へとなる。

```
.............OXOXOXOXOXOX.........
............XOOXOXOXOXO..X.........
............XOOXOXOX..OOXX.........
............XOO..XOXXOOOXX.........
............XOOXXXO..OOOXX.........
............X..XXXOOOOOOXX.........
............XXXXXXOOOOOO...........
```

・n=7　　処理時間は 3427066ms

```
..............OXOXOXOXOXOXOX.........
.............XOOXOXOXOXOXO..X.........
.............XOOXOX..OXOXOOXX.........
.............XOOXOXXOO..XOOXX.........
.............XOOX..XOOOXXOOXX.........
.............XOOXXXXOOO..OOXX.........
.............X..XXXXOOOOOOOXX.........
.............XXXXXXXOOOOOOO...........
```

実行結果を表にする

・3 枚

●●●○○○

・・●○○○●●

・・●○○・・●○●

・・・・○●○●○●

or

・・・・●●●○○○

・・○○●●●○・・

○●○・・●●○・・

○●○●○●・・・・

3 枚は特別？

・4 枚

●●●●○○○○

●・・●○○○○●●

●○○●・・○○●●

●○○●○●○・・●

・・○●○●○●○●

・5 枚

●●●●●○○○○○

●・・●●○○○○○●●

●○○●●○○・・○●●

●○○●・・○●○○●●

●○○●○●○●○・・●

・・○●○●○●○●○●

・6 枚

●●●●●●○○○○○○

●・・●●●○○○○○○●●

●○○●●●○・・○○○●●

●○○・・●○●●○○○●●

●○○●○●○●・・○○●●

●○○●○●○●○●○・・●

・・○●○●○●○●○●○●

・7 枚

●●●●●●●○○○○○○○

●・・●●●●○○○○○○○●●

●○○●●●●○○○・・○○●●

●○○●・・●○○○●●○○●●

●○○●○●●○○・・●○○●●

●○○●○●・・○●○●○○●●

●○○●○●○●○●○●○・・●

・・○●○●○●○●○●○●○●

パターンを推測する

先のプログラムでコンピュータによる 7 枚の実行結果は約 57 分であった。

当たり前だが、本アルゴリズムでの検証は以降指数的に時間がかかる。

ただし、過去に 10 枚までプログラムで検証済みで、10 枚の時は 10 回で交互にできる。

従って、ここまで(7 枚)の結果を踏まえて 8 枚以降を推測する。

・8 枚を面パターンから推測してみる。

1. パターン表から以下の部分はパターンで推測できる

●●●●●●●●○○○○○○○○
●・・●●●●●○○○○○○○○●●
●○○●？？？？？？？？？○○●●
●○○？？？？？？？？？？○○●●
●○○？？？？？？？？？？○○●●
●○○？？？？？？？？？？○○●●
●○○●○●○●○●○●○●○●○・・●
・・○●○●○●○●○●○●○●○●○●

2. もう少しパターンを見つける

●●●●●●●●○○○○○○○○
●・・●●●●●○○○○○○○○●●
●○○●●●●●○？？？？？○○●●
●○○？○？？●○？？？？？○○●●
●○○●○●？●○？？？？？○○●●
●○○●○●？●○？？？？？○○●●
●○○●○●○●○●○●○●○●○・・●
・・○●○●○●○●○●○●○●○●○●

3. 以降、どの手順でパターンを当てはめれば解が得られるか？
   また、別のアプローチがあるか検証する。

実行結果を基に処理の速度アップを行う

処理の雰囲気から、

n=3 の時、配列数はコイン枚数×2+5 あれば処理が破たんしない。

n=3 以外の時、配列数はコイン枚数×2+3 あれば処理が破たんしない。

以上のように配列数をダイエットして再帰数を抑制してみる(修正プログラムは別頁)。

結果、n=3～7 の時、5ms、6ms、13ms、33ms、305ms、で終了したので n=8 以降を試す。

```
Please enter the number of Your coins(3-n)
8
..OXOXOXOXOXOXOX.
XOOXOXOXOXOXOX0..X.
XOOXOXOXOX..OXOOXX.
XOOXO..XOXXOOXOOXX.
XOOXOOXXOXX0..OOXX.
XOOX..XXOXXOOOOOXX.
XOOXXXXX0..OOOOOXX.
X..XXXXXOOOOOOOXX.
XXXXXXXXOOOOOOOO...
4286ms
```

n=8　4286ms

●●●●●●●○○○○○○○○
●・・●●●●●○○○○○○○○●●
●○○●●●●●○・・○○○○○●●
●○○●・・●●○●●○○○○○●●
●○○●○○●●○●●○・・○○●●
●○○●○・・●○●●○○●○○●●
●○○●○●○●○●・・○●○○●●
●○○●○●○●○●○●○●○・・●
・・○●○●○●○●○●○●○●○●

とりあえず、先の推測パターンと一致するか検証。

ちなみに、

n=9　268230ms

●●●●●●●●●○○○○○○○○
●・・●●●●●●○○○○○○○○○●●
●○○●●●●●●○○・・○○○○○●●
●○○●・・●●●○○●●○○○○○●●
●○○●○○●●●○○●●○○・・○●●
●○○●○○●●・・○●●○○●○○●●
●○○●○・・●○●○●●○○●○○●●
●○○●○●○●○●○●・・○●○○●●
●○○●○●○●○●○●○●○●○・・●
・・○●○●○●○●○●○●○●○●○●

n=10　8707475ms

●●●●●●●●●○○○○○○○○○
●・・●●●●●●●○○○○○○○○○●●
●○○●●●●●●●○・・○○○○○○○●●
●○○●・・●●●●○●●○○○○○○○●●
●○○●○○●●●●○●●○○・・○○○●●
●○○●○○●・・●○●●○○●●○○○●●
●○○●○○●●○●○●・・○●●○○○●●
●○○●○・・●○●○●○●○●●○○○●●
●○○●○●○●○●○●○●○●・・○○●●
●○○●○●○●○●○●○●○●○●○・・●
・・○●○●○●○●○●○●○●○●○●○●

これ以上の n を試すにはプログラムの効率化を行うか解法パターンを見つけるしかない。従って以降は数学的解法を試してみる。

数学的検証

このゲームで「コイン(n≧3)枚ずつの時、n 回で必ず交互になるか否か」を証明せよ。また、「必ず n 回でできる場合の解法手順」を説明せよ。

コラム

さらにこの問題を発展させ、コイン数を増加させたり、移動セット数を増価させたり、セットの法則を複雑化させることにより遺伝子の組み換えとの関連やヒモ理論との関連などを考察すると何か新発見が潜んでいるかもしれない。思考の連携とも言える。
特にセットを作る際の隣との連携を指数化しリスト構造を構築し、移動させれば線形構造の複雑系が見えてくる。また、線形だけでなく円形の場合はどのように振舞うか考察できる。
別の見方として本問題はエントロピーの増加、つまり整列から混沌へ向かう法則の一般化モデルとして体系化できる。

コンピュータ的解釈を行うと、本問題は「最も効率良くバラバラにするパターンの発見」への訓練だと言える。一般論として、どんなシステムでも上手いコンピュータシステムを構築する訓練はパターンの発見と楽をする訓練でもあり、ある程度ひらめきが必要だ。その一助として数学的手法も必要であることは言うまでもない。

本問題は「コラッツの問題(3n+1 問題)」に触発され、コンピュータを使用して力ずくで解法(5×2^{60})まで証明されているが、数学的に未解決の問題をヒントにコンピュータ教育に生かせないかと考えた研修用考察である。この課題で新人プログラマーのプログラム能力、推察能力、数学素養をかつて計ることができた。

n=3～10 までの一覧表

●●●○○○ ・・●○○○●● ・・●○○・・●○● ・・・・○●○●○●	●●●●○○○○ ●・・●○○○○●● ●○○●・・○○●● ●○○●○●○・・● ・・○●○●○●○●
●●●●●○○○○○ ●・・●●○○○○○○●● ●○○●●○○・・○●● ●○○●・・○●○○●● ●○○●○●○●○・・● ・・○●○●○●○●○●	●●●●●●○○○○○○○ ●・・●●●○○○○○○○●● ●○○●●●●○・・○○○○●● ●○○●・・●●○●●○○○○○●● ●○○●○○●●○●●○・・○○●● ●○○●○・・●○●●○○●○○●● ●○○●○●○●○●・・○●○○●● ●○○●○●○●○●○●○・・● ・・○●○●○●○●○●○●○●
●●●●●●●●○○○○○○○○ ●・・●●●●○○○○○○○○●● ●○○●●●●○○○・・○○●● ●○○●・・●○○○●●○○○●● ●○○●○●●○○・・●○○○●● ●○○●○●・・○●○●○○●● ●○○●○●○●○●○●○・・● ・・○●○●○●○●○●○●○●	●●●●●●●●○○○○○○○○○ ●・・●●●●●●○○○○○○○○○●● ●○○●●●●●●○・・○○○○○●● ●○○●・・●●○●●○○○○○●● ●○○●○○●●○●●○・・○○●● ●○○●○・・●○●●○○●○○●● ●○○●○●○●○●・・○●○○●● ●○○●○●○●○●○●○●○・・● ・・○●○●○●○●○●○●○●○●
●●●●●●●●●○○○○○○○○○ ●・・●●●●●●●○○○○○○○○○●● ●○○●●●●●●●○○・・○○○○○●● ●○○●・・●●●●○○●●○○○○○●● ●○○●○○●●●○○●●○○・・○●● ●○○●○○●●・・○●●○○●○○●● ●○○●○・・●○●○●●○○●○○●● ●○○●○●○●○●・・○●○○●● ●○○●○●○●○●○●○●○・・● ・・○●○●○●○●○●○●○●○●	●●●●●●●●●○○○○○○○○○○ ●・・●●●●●●●●○○○○○○○○○○●● ●○○●●●●●●●●○・・○○○○○○○●● ●○○●・・●●●●○●●○○○○○○○●● ●○○●○○●●●●○●●○○・・○○○●● ●○○●○○●・・●○●●○○●●○○○●● ●○○●○○●○●○●・・○●●○○○●● ●○○●○●○●○●○●○●○・・● ・・○●○●○●○●○●○●○●○●

```
//////////////////////////////////////////
//        coin Change  (最終バージョン)
//
//        coded by S.Kubo    1996 mod.2015
//////////////////////////////////////////
import java.util.Scanner;
import java.util.Date;

//////////////////////
//        enum
//
enum markType{
          BLACK("X"),
          WHITE("O"),
          NULL(".");
          private final String text;

          private markType( final String text ){ this.text=text; }
          @Override
          public String toString(){ return this.text; }
}


//////////////////////
//coinBase
//
class coinBase{
          markType arry[];
          int numMax;
          int numCoin;

//============
//construct
          public coinBase( int n ){
                    numCoin= n;
                    if( numCoin==3 ){
                              numMax=numCoin*2+5;
                              arry = new markType[numMax];
                    }else{
                              numMax=numCoin*2+3;
                              arry = new markType[numMax];
                    }
          }
          public coinBase( coinBase coins ){
                    numCoin=coins.numCoin;
                    numMax=coins.numMax;
                    arry = new markType[numMax];
                    for( int i=0; i<numMax; i++ ){
                              arry[i]=coins.arry[i];
                    }
          }
//============
//Initialize
          public void init(){
                    int cnt=0;
                    for( int i=0; i<numCoin; i++ ){
                              arry[cnt++] = markType.BLACK;
                    }
                    for( int i=0; i<numCoin; i++ ){
                              arry[cnt++] = markType.WHITE;
                    }
                    for( int i=cnt; i<numMax; i++ ){
                              arry[i] = markType.NULL;
                    }
          }
```

```
//============
//Print
        public void print(){
                for( int i=0; i<numMax; i++ ){
                        System.out.print( arry[i] );
                }
                System.out.println();
        }
//============
//Is Get 2-Coins
        public boolean isGet( int n ){
                if( n<0 || n>numMax-2 ){ return false; }
                if( arry[n]==markType.NULL || arry[n+1]==markType.NULL ){
                        return false;
                }else{
                        return true;
                }
        }
//============
//Is Put 2-Coins Null-Case
        public boolean isPutNull( int n ){
                if( n<0 || n>numMax-2){ return false; }
                if( arry[n]==markType.NULL && arry[n+1]==markType.NULL ){
                        return true;
                }else{
                        return false;
                }
        }
//============
//Is Put 2-Coins Touch-Case
        public boolean isPutTouch( int n ){
                if( n<1 || n>numMax-3 ){ return false; }
                if( arry[n+2]!=markType.NULL || arry[n-1]!=markType.NULL ){
                        return true;
                }else{
                        return false;
                }
        }
//============
//Move 2-Coins
        public void move( int i, int j){
                arry[j] = arry[i];
                arry[j+1] = arry[i+1];
                arry[i] = arry[i+1] = markType.NULL;
        }
//============
//Is Alternate Coins
        public boolean isAlternate(){
                for( int i=0; i< numMax-1; i++ ){
                        if( arry[i]!=markType.NULL && arry[i]!=arry[i+1] ){
                                for( int j=i; j< i+numCoin*2; j+=2 ){
                                        if( arry[j]!=arry[i] || arry[j+1]!=arry[i+1] ){
                                                return false;
                                        }
                                }
                                return true;
                        }
                }
                return false;
        }
```

```
//============
//Recursive Coin-Patterns
        public boolean run( int n ){
        //print();
                if( n>=numCoin ){
                        return isAlternate();
                }
                for( int i=0; i< numMax-1; i++ ){
                        if( isGet( i ) ){
                                for( int j=0; j<numMax-1; j++ ){
                                        if( isPutNull( j ) && isPutTouch( j ) ){

                                                coinBase coins = new coinBase(this);
                                                coins.move( i, j );

                                                if( coins.run( n+1 )==true ){
                                                        coins.print();
                                                        return true;
                                                }
                                        }
                                }
                        }
                }
                return false;
        }
}

//////////////////////
//      coinChange Main Process
//
public class coinChange{
        public static void main(String[] args) {
                Scanner in = new Scanner( System.in );
                int maxN;
                do{
                        System.out.println( "Please enter the number of Your coins(3-n)" );
                        maxN = in.nextInt();
                }while( maxN<3 );
                coinBase coins = new coinBase(maxN);
                coins.init();
                Date start = new Date();
                if( coins.run( 0 )==true ){
                        coins.print();
                }else{
                        System.out.print( "Can Not" );
                        System.out.println();
                }
                Date end = new Date();
                long t=end.getTime() - start.getTime();
                System.out.print( t+"ms" );
                System.out.println();
        }
}
```