私の母は私
ー再帰的プログラムー

岐阜経済大学　経営学部 経営情報学科　井戸　伸彦

来歴:

0.0版　2003年4月14日

1.0版　2005年11月4日:(3)(4)に追記

スライドの構成

はじめに

(1) ハノイの塔

(2) 再帰的手順

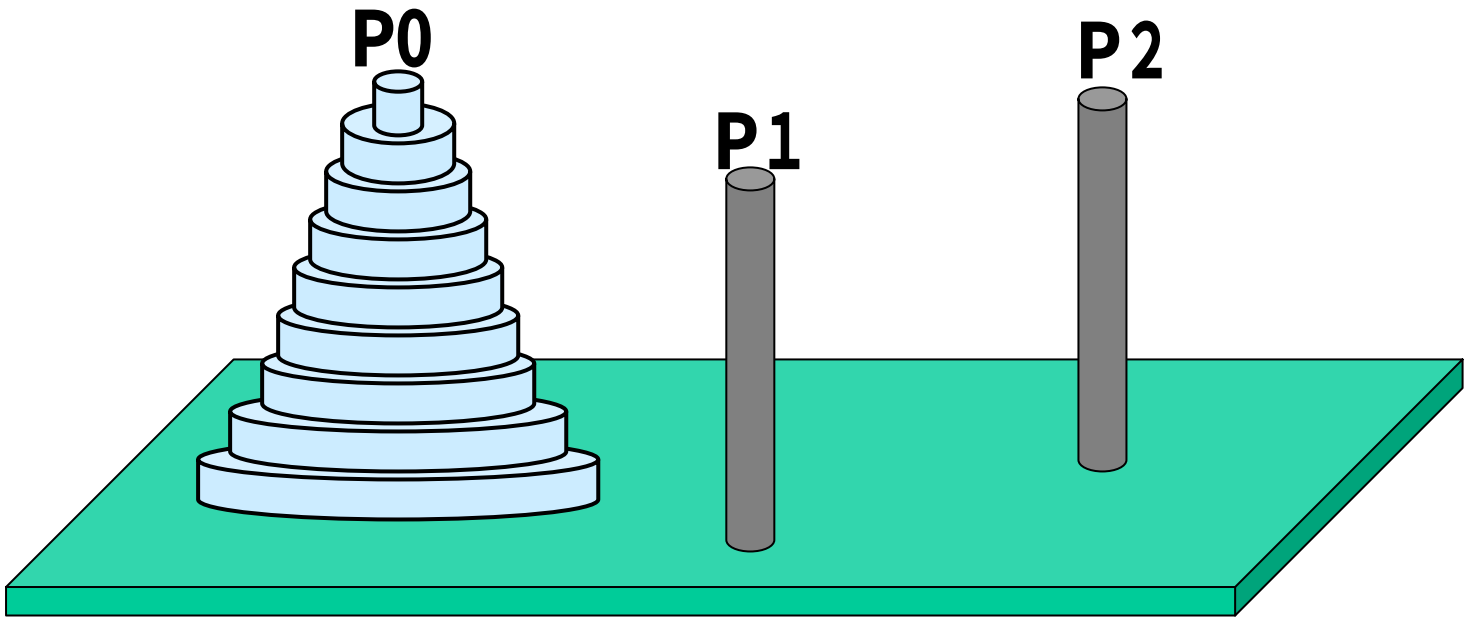
(3) 再帰的手順を使う分野(例題)

(4) 課題

(1) ハノイの塔　ー問題の説明ー

■3本の棒(pole)に、大きさの違う8枚の円盤(disk)が積まれている。

■最初は図のように、円盤はP0の棒に積まれている。



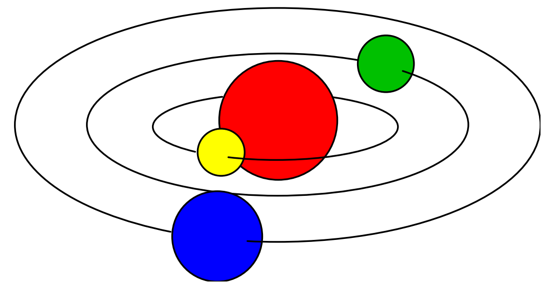
■円盤を一枚づつ移動して、すべての円盤をP2に移すには何回移動する必要があるか？　　但し、大きい円盤を小さい円盤の上に積んではいけない。

（１．１）物語

■黄金で出来た６４枚の円盤を３本のダイアモンドの棒に積んだ、ブラーマ（Brahma:梵天）の塔。

■この世の初めに、神は黄金の円盤を第一の棒に積んだ。

■一団の僧侶たちに、規則に従って円盤を第三の棒に移すよう命じた。

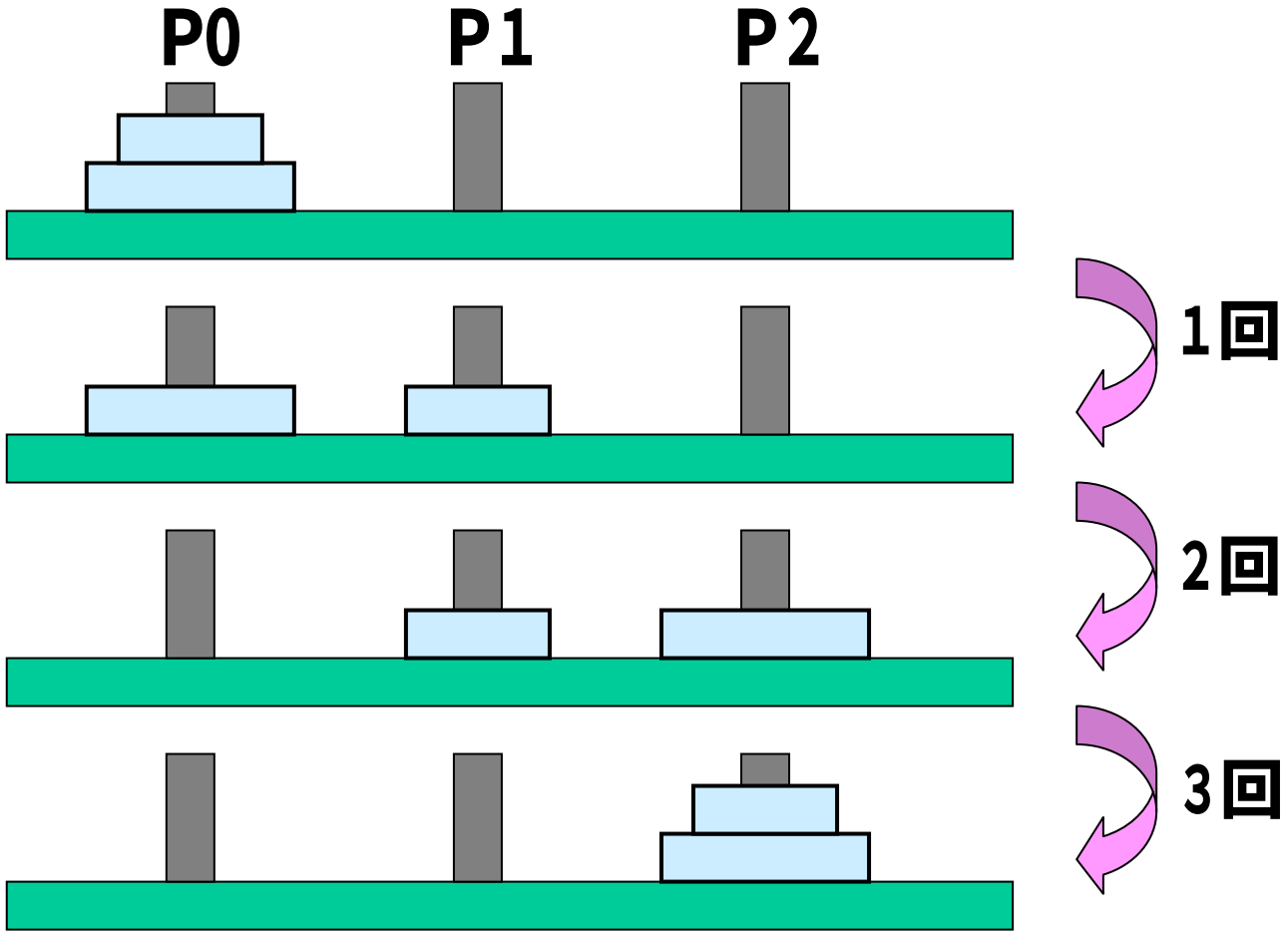


■僧侶たちは、昼夜をおかずその仕事に専念せねばならない。

■仕事が完成すると塔が崩壊して、この世は終焉する。
（１８８３年：フランスの数学者リュカ）

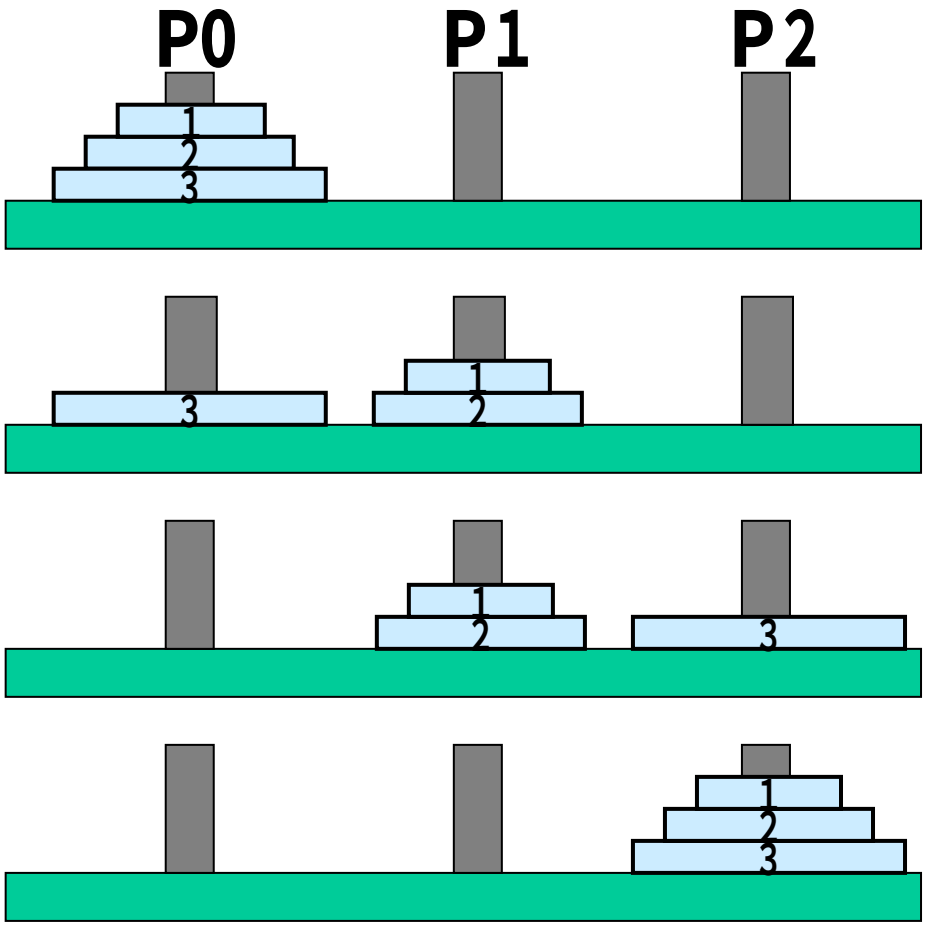
(1．2) やってみよう

■ 円盤2枚の場合



⇒答え：3回

(1．3) 円盤3枚の場合



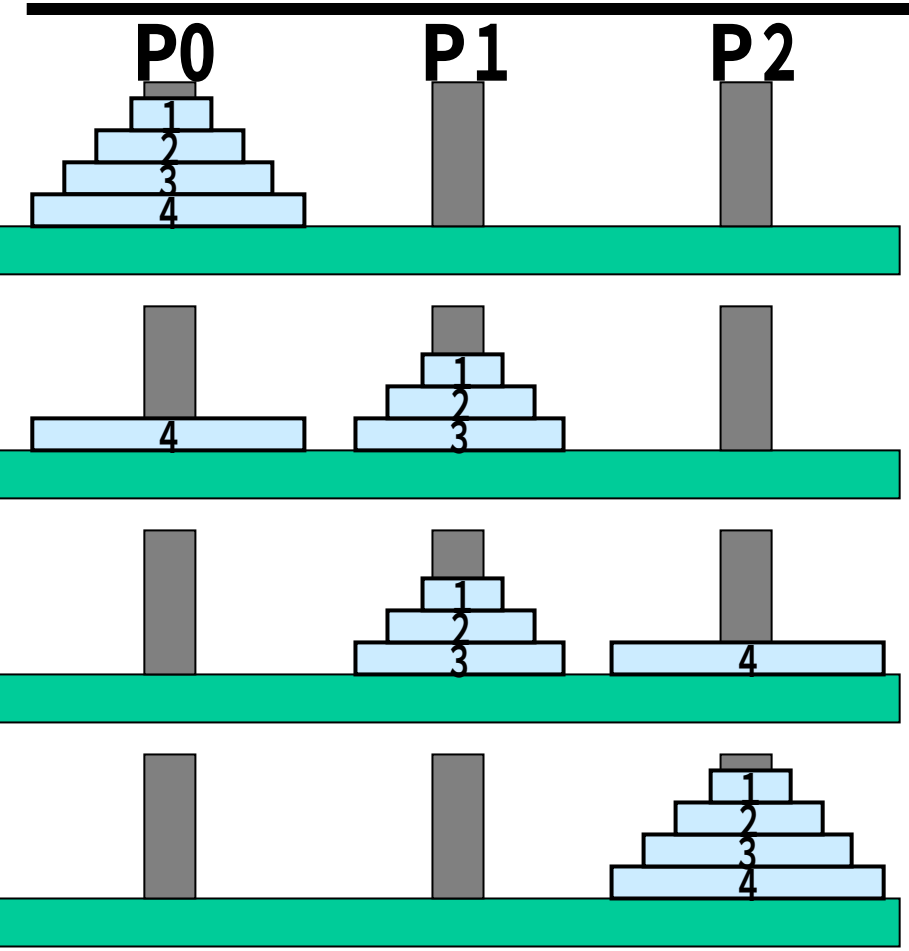
2枚を移す手順を使って、
1と2とを
P0からP1へ移す (3回)

3をP0からP2へ移す (1回)

2枚を移す手順を使って、
1と2とを
P1からP2へ移す (3回)

⇒答え：7回 (＝3＋1＋3)

(1．4) 円盤4枚の場合



3枚を移す手順を使って、
1,2,3を
P0からP1へ移す。(7回)

4をP0からP2へ移す。(1回)

3枚を移す手順を使って、
1,2,3を
P1からP2へ移す。(7回)

⇒答え：15回 (＝7＋1＋7)

■ 何か同じことをやっているようだ。

⇒何枚の場合でも通用する規則はなんだろう？ (⇒一般化)

(1．5) 命名統治 (name and conquer)

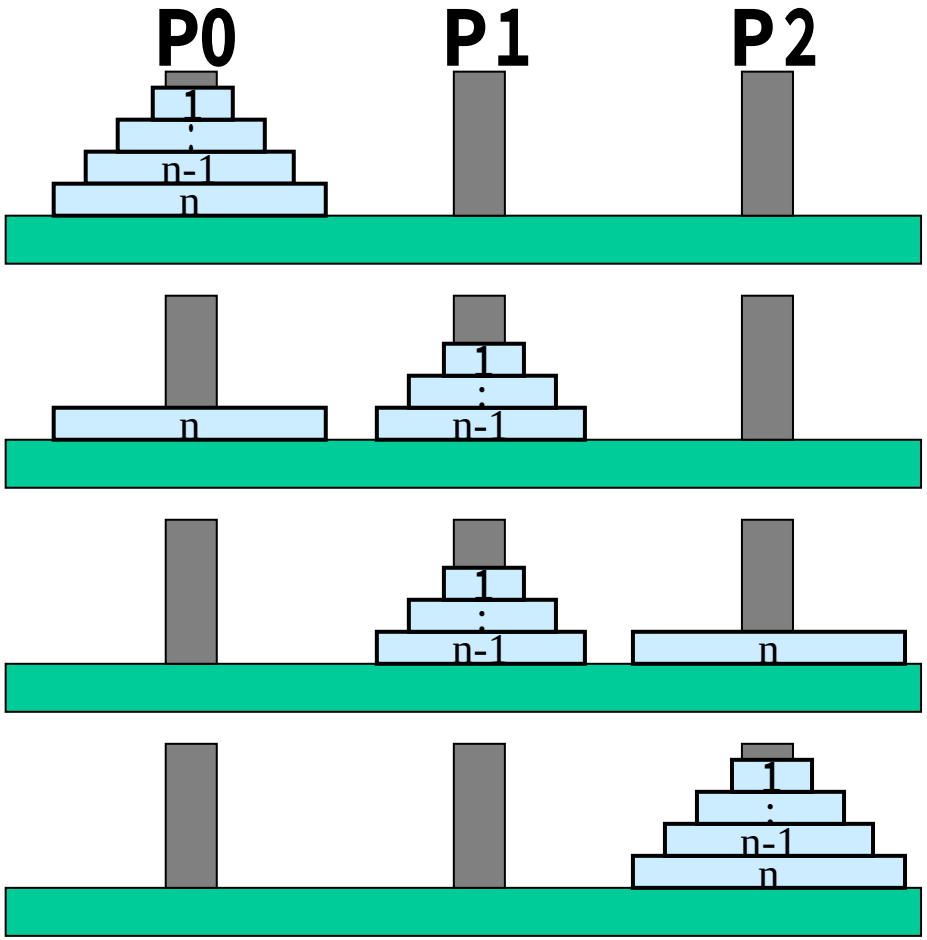
■ 発見したことは？

- (3枚を移す回数) ＝ (2枚を移す回数) ×2＋1
  ＝3×2＋1＝7
- (4枚を移す回数) ＝ (3枚を移す回数) ×2＋1
  ＝7×2＋1＝15

■ 上記のように書くのは煩わしいし、読みにくい。

- (2枚を移す回数) を、T_2と書く。
  (3枚を移す回数) を、T_3と書く。以下、同様。すると、
  - $T_3 = T_2 \times 2 + 1 = 7$
  - $T_4 = T_3 \times 2 + 1 = 15$
- 命名 (T) により、分かりやすくなった (？)

(1．6) 円盤n枚の場合



n-1枚を移す手順を使って、
1,2,..,n-1を
P0からP1へ移す (Tn-1回)

nをP0からP2へ移す (1回)

n-1枚を移す手順を使って、
1,2,..,n-1を
P1からP2へ移す (Tn-1回)

- ⇒答え：Tn＝ Tn-1＋1＋ Tn-1 ＝ 2Tn-1＋1 (←漸化式)
- これは、nがいくつであっても、成り立つ。一般化、OK.

(1．7) これで十分か？

■何枚の場合でも求めることが出来る。

- $T_n = 2T_{n-1} + 1$だから、
  - $T_4 = 2T_3 + 1 = 2 \cdot 3 + 1 = 7$
  - $T_5 = 2T_4 + 1 = 2 \cdot 7 + 1 = 15$
  - $T_6 = 2T_5 + 1 = 2 \cdot 15 + 1 = 31$

    ⋮

  - $T_{40} = 2T_{39} + 1 = 2 \cdot 549755813888 + 1$

    $= 1099511627776$
- 結構面倒だ。電卓使っても大変。

■一発で求める方法はないのか？

（1．8）漸化式を解く

■問題の定式化

◆$T_n = 2T_{n-1} + 1$、$T_0 = 0$

■解き方

◆$U_n = T_n + 1$（$T_n = U_n - 1$）とおく。

◆$T_n = 2T_{n-1} + 1$、$T_0 = 0$に、U_nを代入する。

$$U_n - 1 = 2(U_{n-1} - 1) + 1 = 2U_{n-1} - 2 + 1$$

$$\therefore \quad U_n = 2U_{n-1}$$

$$U_0 - 1 = T_0 = 0 \quad \therefore U_0 = 1$$

◆U_nを求める。

$$U_n = 2U_{n-1} = 2 \cdot 2U_{n-2} = 2 \cdot 2 \cdot 2U_{n-3}$$

$$= \ldots = \underbrace{2 \cdot 2 \ldots \ldots \cdot 2}_{n個} U_0 = 2^n$$

◆よって、$T_n = 2^n - 1$

(2) 再起的(recursive)手順

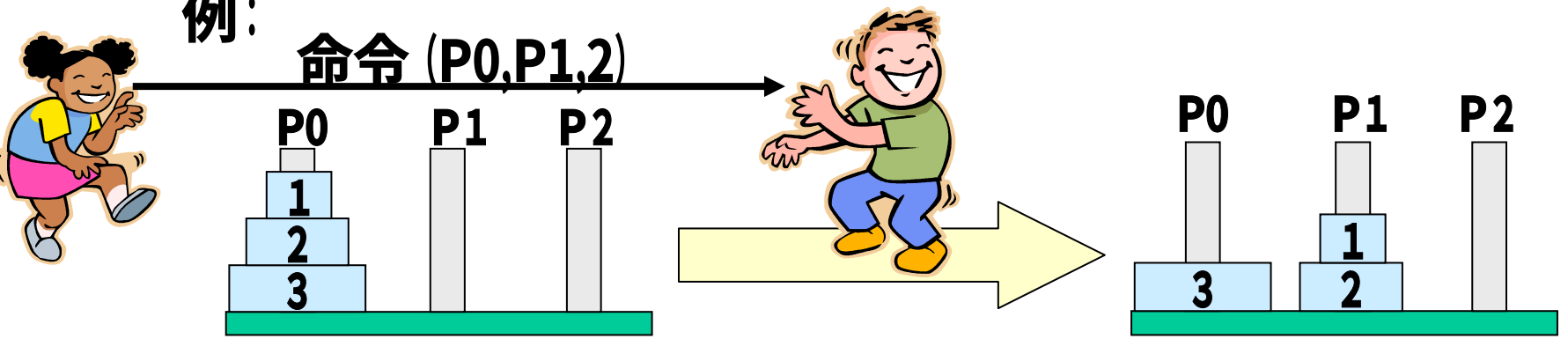
■何人かで協力して、ハノイの塔を解く。

■全員、次のシナリオに従うものとする。

- 命令の形式は (X,Y,z)

  ①どの棒 (X) から、②どの棒 (Y) へ、③何枚 (z) 移せ。

  例:

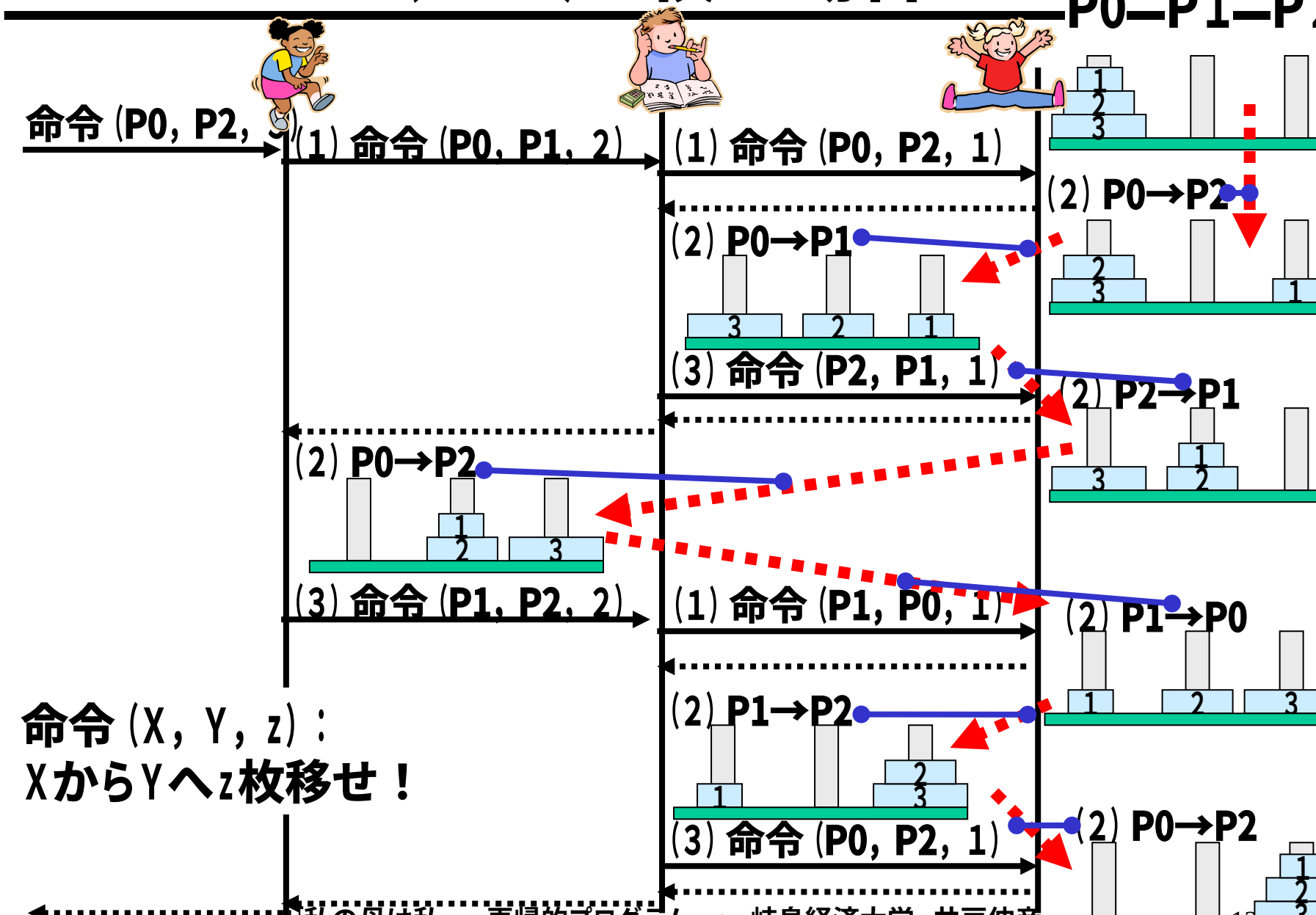


- シナリオ:

  命令 (X,Y,z) を受けたら、次の動作を行う。

  - (1) 誰かに、(X,「X,Y以外の残りの棒」,z－1) と命令する。
  - (2) XからYへ一枚移す。
  - (3) 誰かに、(「X,Y以外の残りの棒」,Y,z－1) と命令する。

(2．1) 3枚の場合



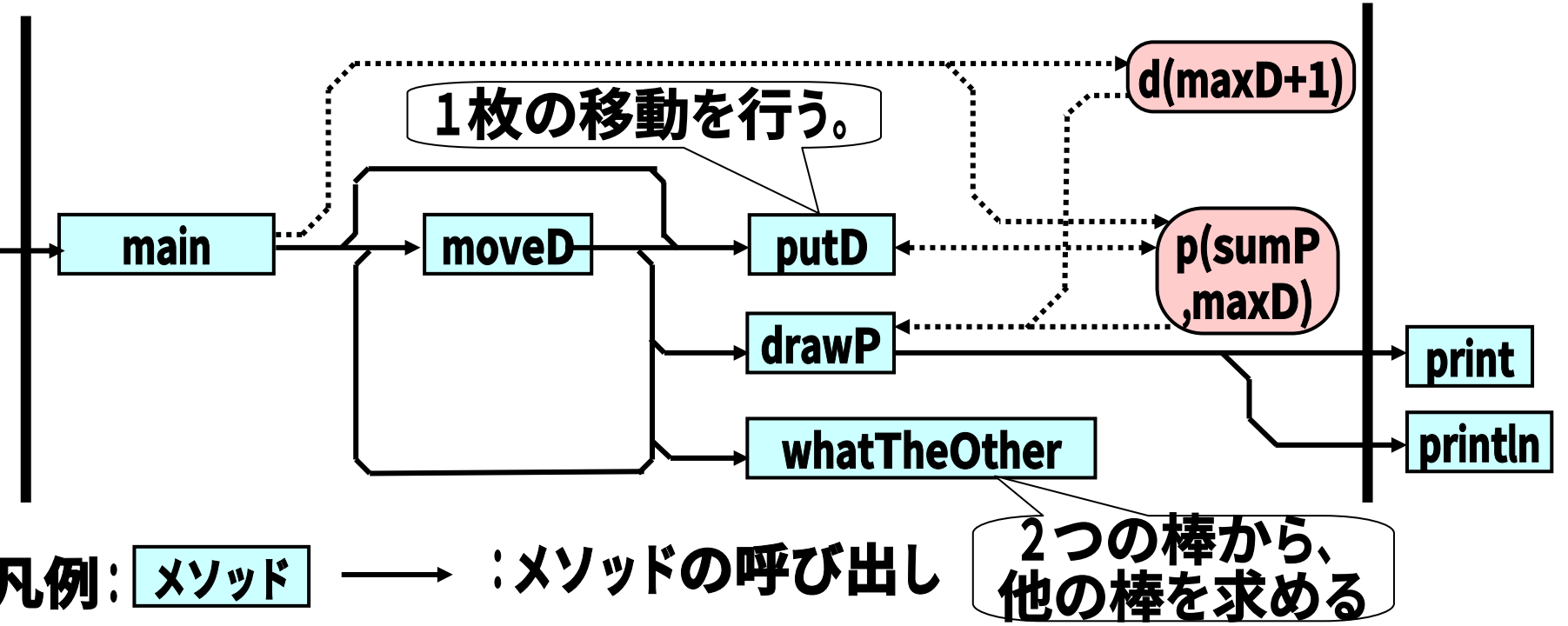
命令 (X, Y, z) :
XからYへz枚移せ！

(2．2) プログラム構成

■井戸のネットワークドライブより、次のプログラムをコピーして、実行してみてください。

- Hanoi.java

■プログラムでは、moveDが再帰的手順となっています。



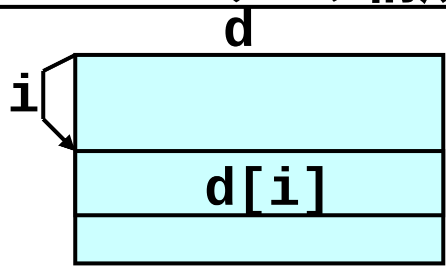
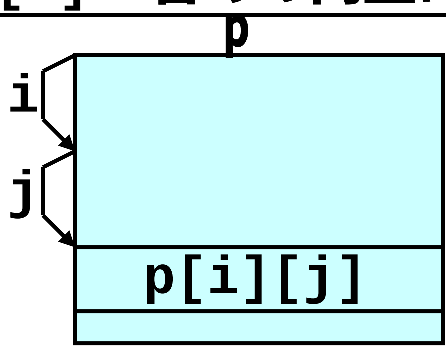
(2．3) メソッド仕様

■インターフェース仕様書

項番	メソッド名	パラメタ名	I/O	型	説明
1	main				データの初期設定を行う。 移動処理を起動する。
2	moveD				円盤の移動を指令する。
		iFrom	I	整数	移動元の棒の番号（0～2）。
		iTo	I	整数	移動先の棒の番号（0～2）。
		numD	I	整数	移動する円盤の枚数。
3	putD				円盤1枚の移動を実行する。
		iFrom	I	整数	移動元の棒の番号（0～2）。
		iTo	I	整数	移動先の棒の番号（0～2）。
4	drawP				棒の描画を行う。
5	whatTheOther				指定した2本の棒以外の棒の番号を計算する。
		fst	I	整数	1つめの棒の番号（0～2）。
		snd	I	整数	2つめの棒の番号（0～2）。
		what	O	整数	2本の棒以外の、棒の番号（0～2）

(2.4.1)データ構成

■データ構成図

項番	データ名	データ構成	説明・備考
1	円盤文字列テーブル	 i:円盤の小さい順の番号 d[i]:i番めの円盤に対応する文字列	最初に設定され、その後は読み出し専用。
2	棒内円盤テーブル	 i:棒の番号(0〜2) j:棒の下から数えた位置 p[i][j]:i番の棒の、j番の位置にある、円盤の番号	各棒の円盤にどのように円盤が載っているかを表す。

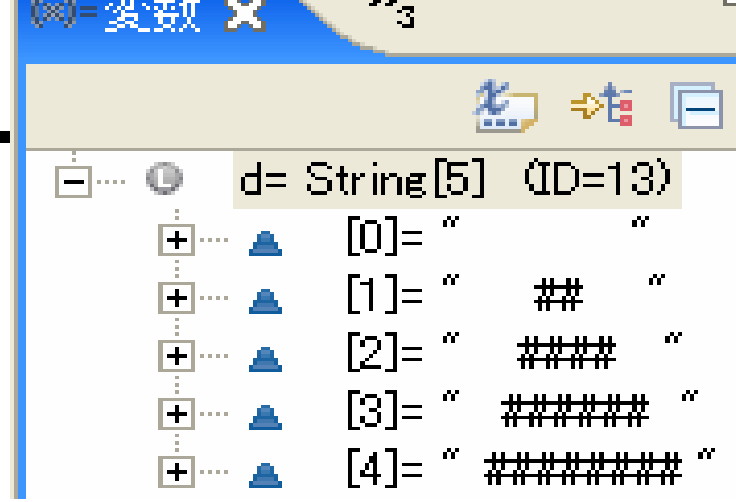
(2．4．2) 値の例示

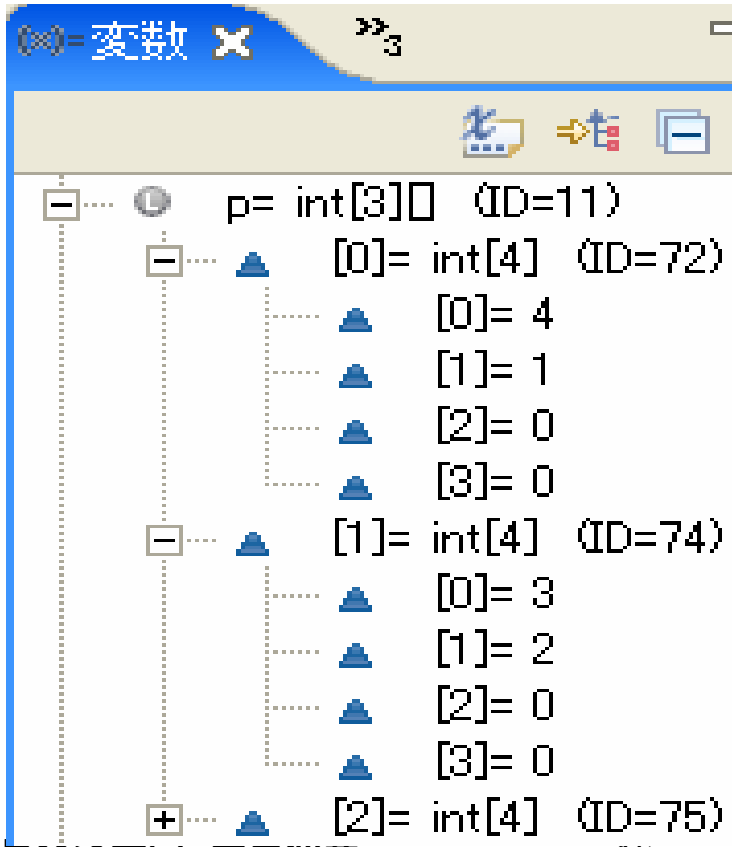
■円盤文字列テーブル“d”

- 右図のように、空白と“＃”とを連ねた文字列が入ります。これを用いて円盤を描画しています。

■棒内円盤テーブル“p”

- 例えば下図のように円盤がある際には、右図のような値が入っています。





(2．5) 再帰呼び出し

■Hanoi.javaで、円盤の移動のやり方を記述しているのは、次のメソッド“moveD”での数行だけです。

```
public static void
            moveD(int iFrom,int iTo,int numD){
  if(numD==0){
    return;
  }
  moveD(iFrom,whatTheOther(iFrom,iTo),numD-1);
  putD(iFrom,iTo);
  DrawP();

  moveD(whatTheOther(iFrom,iTo),iTo,numD-1);
}
```

元の棒から

先の棒へ

この枚数の円盤を移動させる

元の棒から

別の棒へ

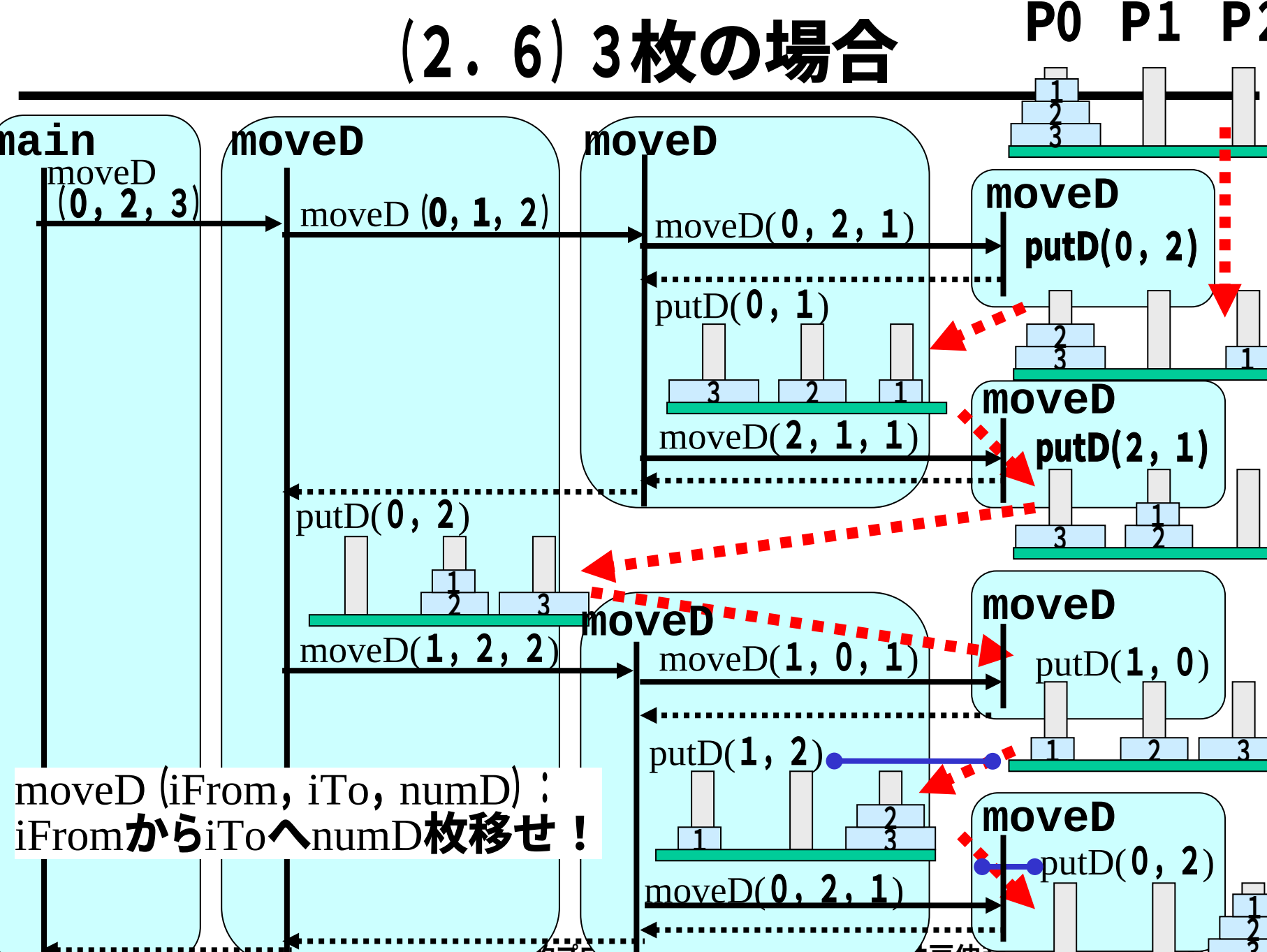
1枚移す

別の棒から

先の棒へ

■(2)に記したシナリオがそのまま示されています。

(2．6) 3枚の場合


P0 P1 P2
main
moveD
(0, 2, 3)
moveD
moveD (0, 1, 2)
moveD
moveD(0, 2, 1)
moveD
putD(0, 2)
putD(0, 1)
moveD(2, 1, 1)
moveD
putD(2, 1)
putD(0, 2)
moveD(1, 2, 2)
moveD
moveD(1, 0, 1)
moveD
putD(1, 0)
putD(1, 2)
moveD(0, 2, 1)
moveD
putD(0, 2)
moveD (iFrom, iTo, numD)：
iFromからiToへnumD枚移せ！

(3) 再帰的手順を使う分野 (例)

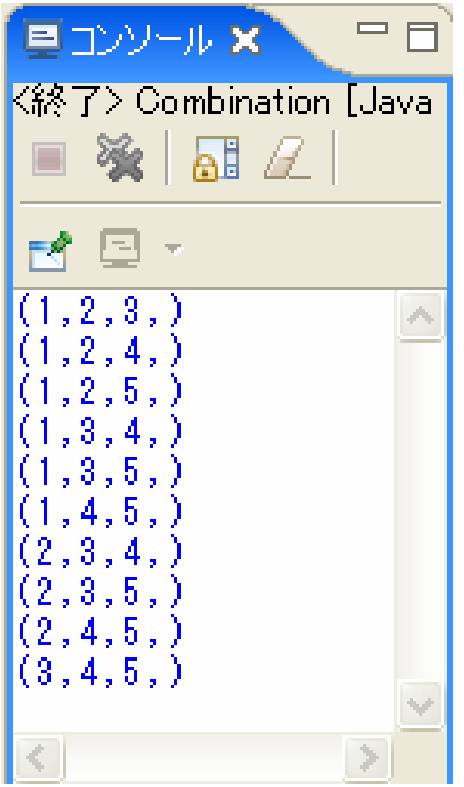
■組み合わせや、順列に関係することは、基本的に再帰的手順で行うことになります。

■次の問題を考えます。

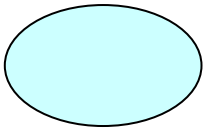
- 1からnまでの数から、k個を取り出したときのすべての組み合わせを挙げよ。

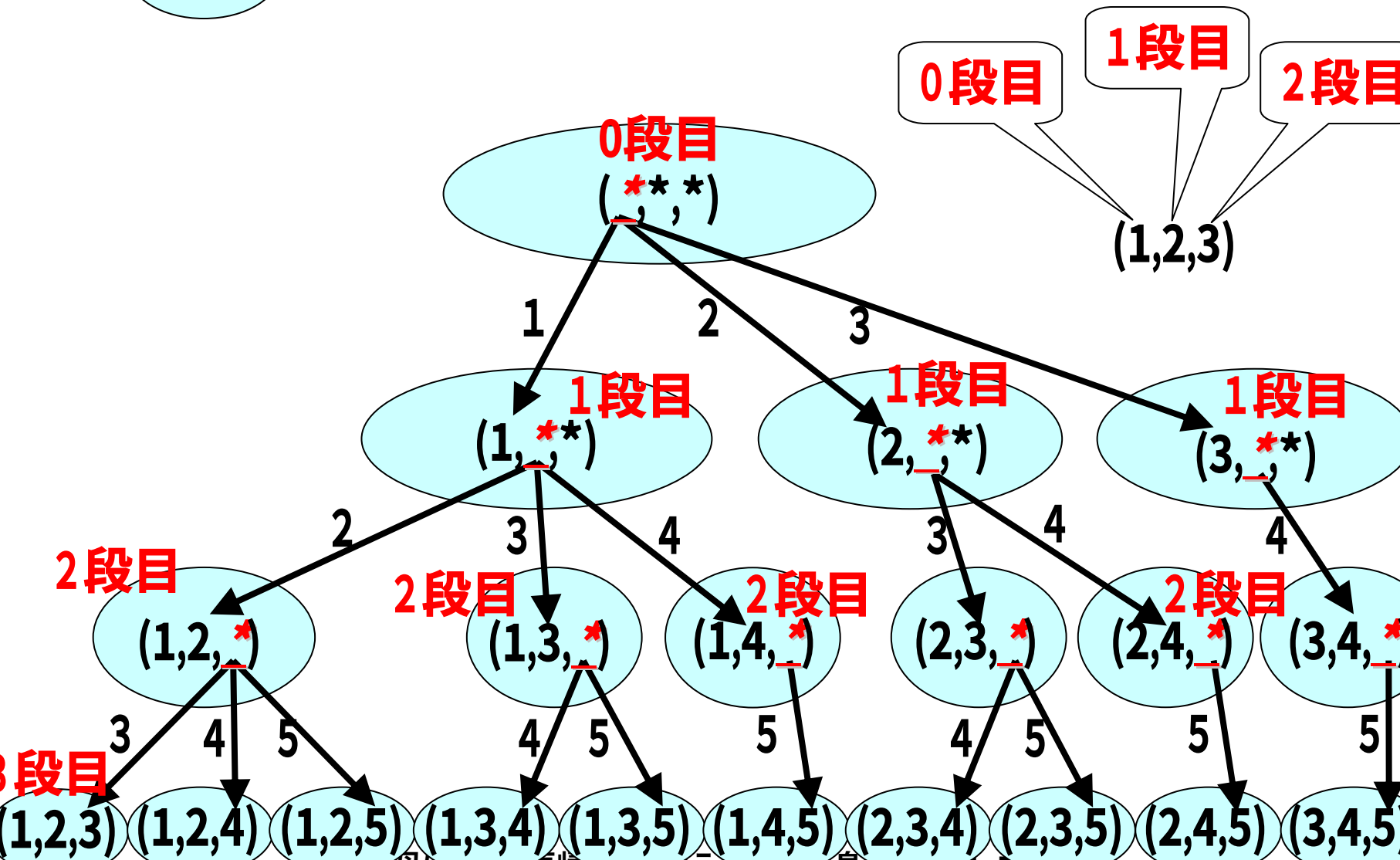
◆n=5, k=3とすると、

(1,2,3,)
(1,2,4,)
(1,2,5,)
(1,3,4,)
(1,3,5,)
(1,4,5,)
(2,3,4,)
(2,3,5,)
(2,4,5,)
(3,4,5,)



(3．1) 考え方

■  の手順が、再帰的な手順となります。



(3．2)プログラム

```
public class Combination {
  public static void main(String[] args) {
    int n = Integer.parseInt(args[0]);
    int k = Integer.parseInt(args[1]);
    int[] ar = new int[k];
    comb(n,0,1,ar);
  }
  public static void comb(int n,int loc,int start,int[] ar){
    if(loc==ar.length){
      System.out.print("(");
      for(int i=0;i<ar.length;i++){
        System.out.print(ar[i]+",");
      }
      System.out.println(")");
      return;
    }
    for(int i=start;i<=n;i++){
      ar[loc]=i;
      comb(n,loc+1,i+1,ar);
    }
  }
}
```

(3.3) プログラムの構成

■mainメソッドから呼び出された、combメソッドが再帰的呼び出しを行うことで組み合わせを列挙します。


```
public static void main(String[] args) {
    n = Integer.parseInt(args[0]);
    k = Integer.parseInt(args[1]);
    int[] ar = new int[k];
    comb(n,0,1,ar);
}

public static void
      comb(int n,int loc,int start,int[] ar)
  // 再起的手順
}
```

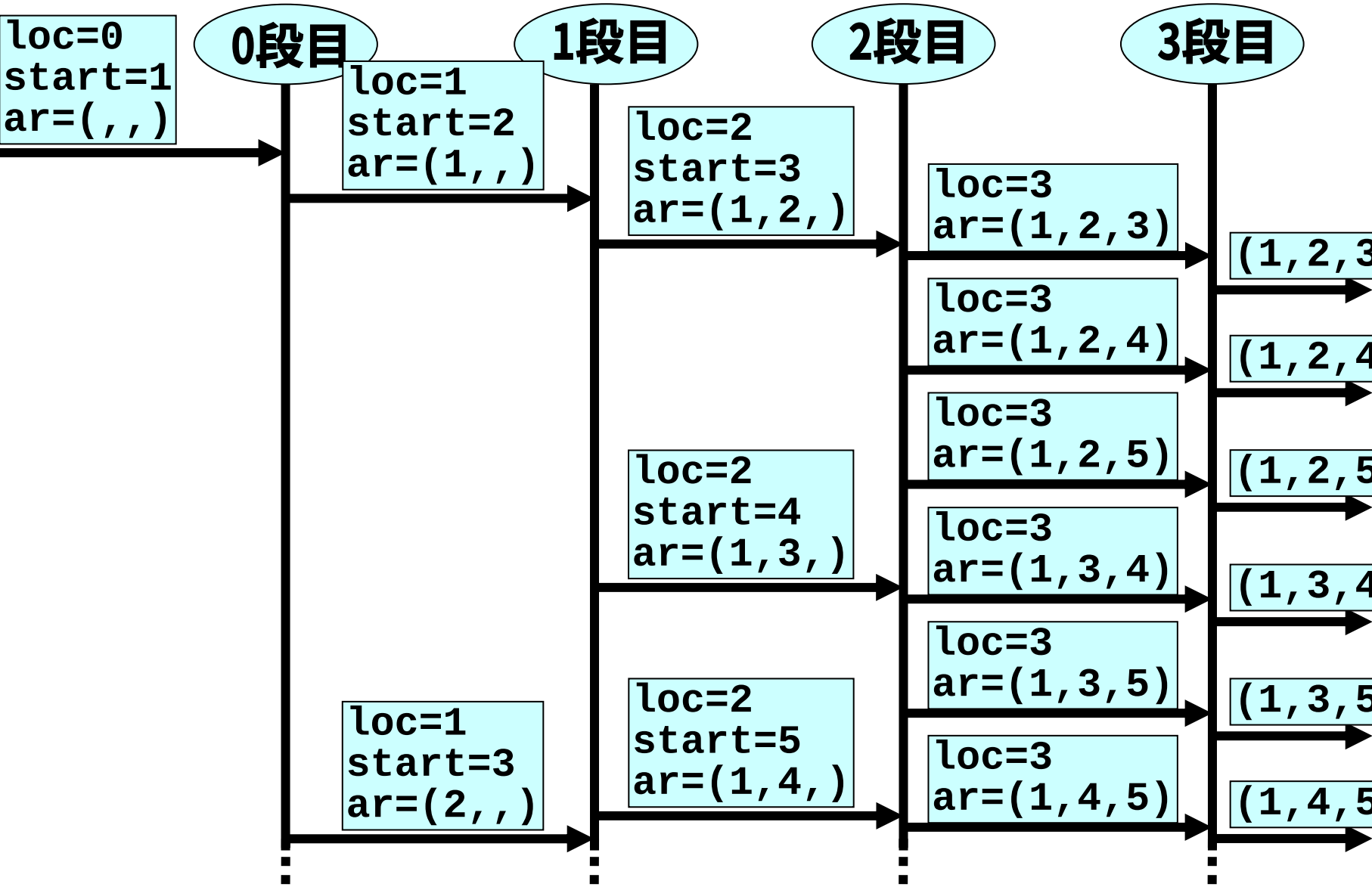
何段目か（loc）

使ってよい最初の数（start）

組み合わせの配列（ar）

“1からnまで”のn（n）

(3．4) 動作の概要


loc=0
start=1
ar=(,,)
0段目
1段目
2段目
3段目
loc=1
start=2
ar=(1,,)
loc=2
start=3
ar=(1,2,)
loc=3
ar=(1,2,3)
(1,2,3
loc=3
ar=(1,2,4)
(1,2,4
loc=3
ar=(1,2,5)
(1,2,5
loc=2
start=4
ar=(1,3,)
loc=3
ar=(1,3,4)
(1,3,4
loc=3
ar=(1,3,5)
(1,3,5
loc=1
start=3
ar=(2,,)
loc=2
start=5
ar=(1,4,)
loc=3
ar=(1,4,5)
(1,4,5

(3．5) 1段目の動き

■メソッドcombは次のようなパラメタを受け取ります。

```
...comb(int n,
        int loc,
        int start,
        int[] ar){
```

1の位置に入れる

5までを入れる

2から

(1, _, *)に入れる

■次のように動きます。

(_, *, *)

1

1の位置に入れる

(1,_,*)に入れる

1段目 (1,_,*)

2から

2 3 4 5

5までを入れる

2段目 (1,2,_) | 2段目 (1,3,_) | 2段目 (1,4,_) | 2段目 (1,5,_)

```
for(int i=start;i<=n;i++){
    ar[loc]=i;
    comb(n,loc+1,i+1,ar);
}
```

1の位置に入れる

2から

5までを入れる

(1,_,*)に入れる

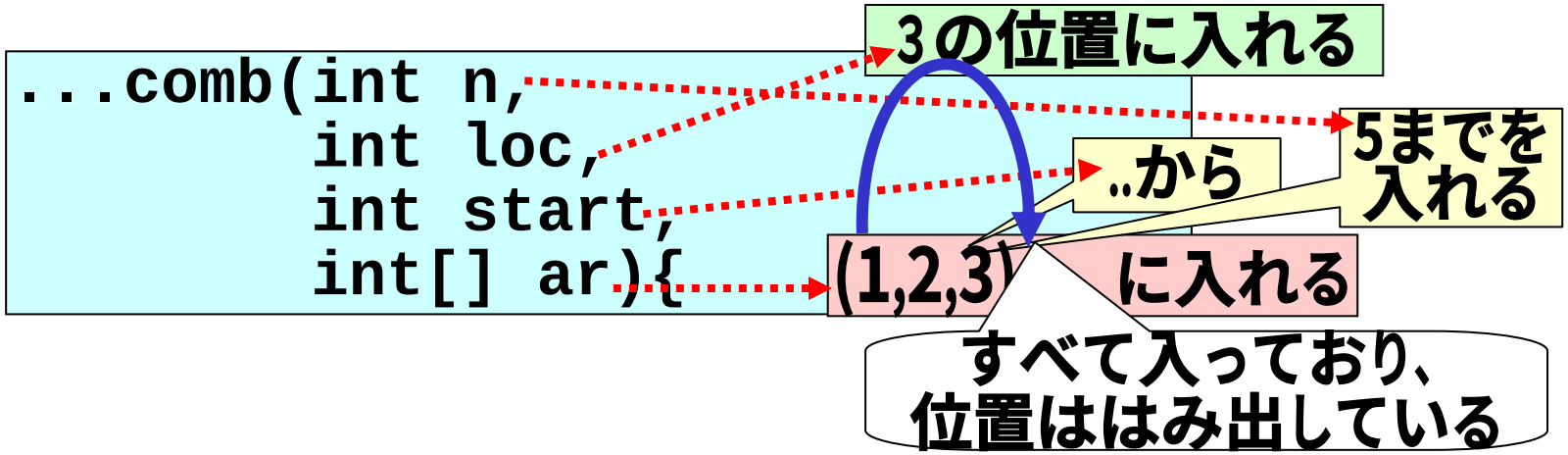
入れる位置を1つ進める

入れる数を1つ進める

start=5は、次の段を生成せずに終了する

(3．6) 3段目の動き

■メソッドcombは次のようなパラメタを受け取ります。



■既に配列(ar)のすべての要素に数値が入っていることを判定して、その場合、配列の内容を出力して終了します。

```
if(loc==ar.length){
    System.out.print("(");
    for(int i=0;i<ar.length;i++){
      System.out.print(ar[i]+",");
    }
    System.out.println(")");
   return;
}
```

すべて入っており、位置ははみ出しているか？

配列を出力する

終了する

（4．1）課題1、2、3

■課題1：ハノイの塔のプログラムで、次の変更を行ってください。

- 円盤の枚数を、5枚にする。
- 円盤を表す文字を、“＃”から“＊”へ変更する。
- 円盤の移動を、右端の棒から左端の棒にする。

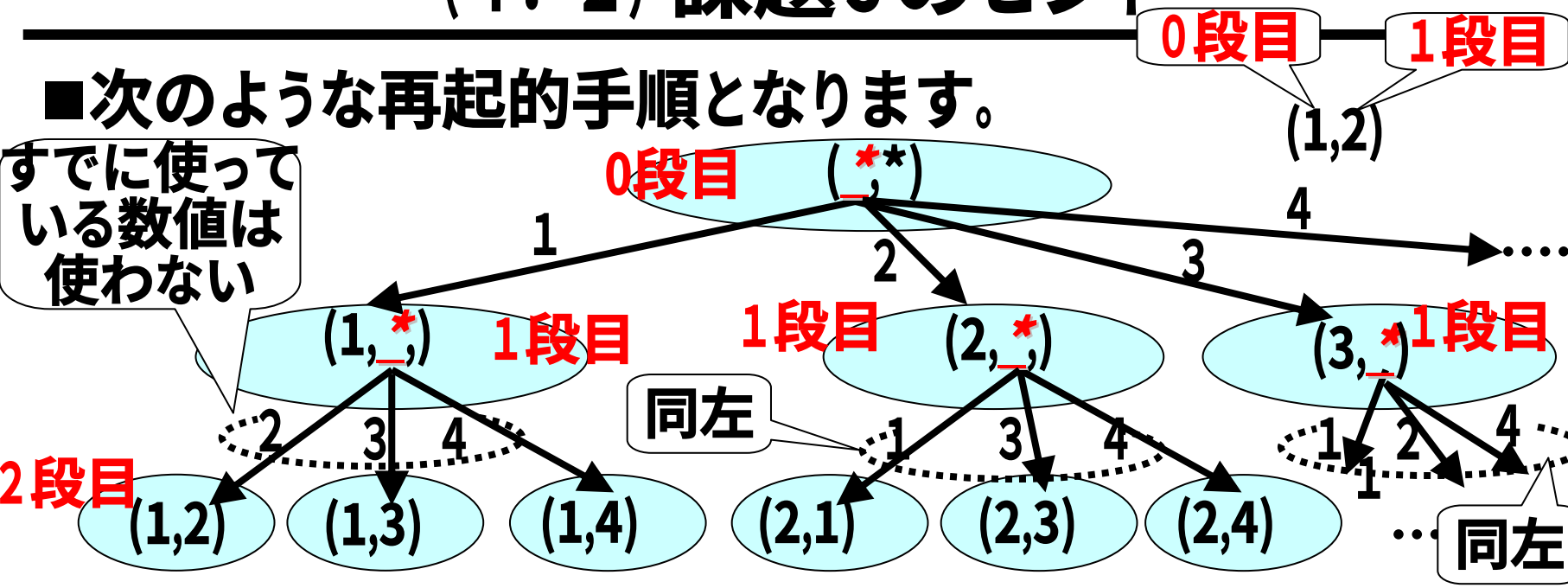
■課題2：スライド（3．2）のプログラムを作成し、動作を確認してください。

■課題3：1からnまでの数からk個を選んで並べる方法（順列）をすべてプリントアウトするプログラムを作ってください。

- 例　：　n＝4　k＝2
- (1,2),(1,3),(1,4),(2,1),(2,3),(2,4),(3,1),(3,2),(3,4),(4,1),(4,2),(4,3)

(4．2) 課題3のヒント

■次のような再起的手順となります。



■プログラム

```
public static void main(String[] args) {
  n = Integer.parseInt(args[0]);
  k = Integer.parseInt(args[1]);
  int[] ar = new int[k];
  perm(n,0,ar);
}
public static void
      perm(int n,int loc,int[] ar){
  // ここの部分を作成する
}
```