C プログラミング

— 再帰 (2) —

早稲田大学

本日の目標

- 再帰呼び出しの例
  - ハノイの塔
  - トーナメント方式の総和

ハノイの塔（問題設定）

- ハノイの塔は 3 本の棒（A, B, C），中央に穴の開いた大きさの異なる n 枚の円盤を用いるゲーム.
- すべての円盤が左端の棒 A に小さいものが上になるように順に積み重ねられている状態からスタートし，これを移動ルールに従って，右の棒 C に小さいものが上になるように積み重ねる.
- 円盤は一回に一枚ずつどれかの棒に移動させることができるが，小さな円盤の上に大きな円盤を乗せることはできない.

- すべての円盤を移動させるには少なくとも $2^n - 1$ 回の手数がかかる.

ハノイの塔（解法）

- 一番最初の手は，棒 A の頂上に積まれた一番小さい円盤を移動すること.
- しかし，棒 B，棒 C のどちらに積んだら良いか，この時点では決定できない.
- ループでは書くことができないため，再帰呼び出しを用いて書く.

↓

分かっている操作を探す.

ハノイの塔（解法）

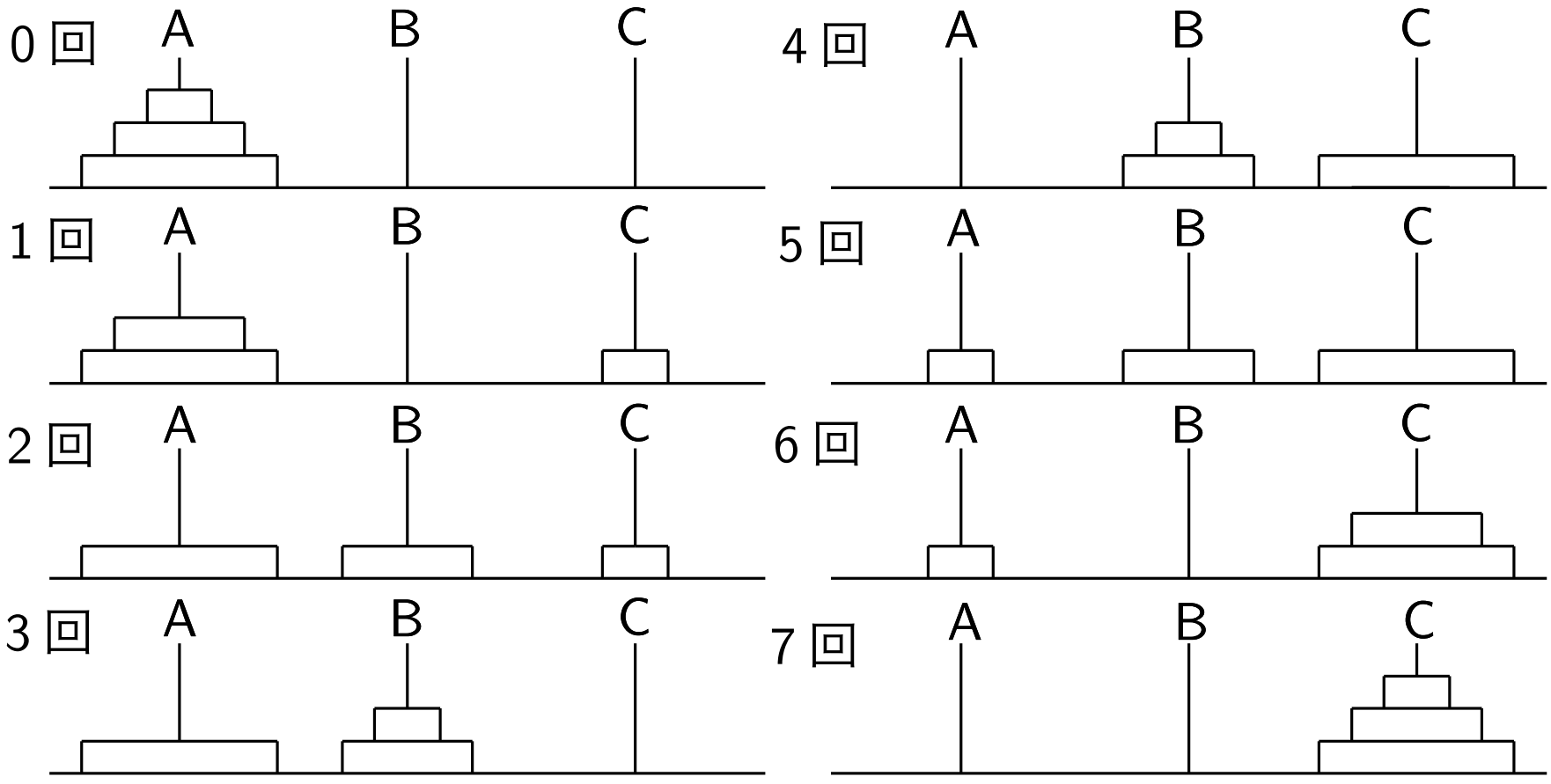
- 一番最初の手は，棒 A の頂上に積まれた一番小さい円盤を移動すること.
- しかし，棒 B，棒 C のどちらに積んだら良いか，この時点では決定できない.
- ループでは書くことができないため，再帰呼び出しを用いて書く.

↓

分かっている操作を探す.

ハノイの塔（解法）

- 3 枚の円盤を移動する手順



ハノイの塔（解法）

ハノイの塔を攻略するためには…

- 移動させる円盤が n 枚のとき，まずは $n-1$ 枚を棒 A から棒 B に移動しておく必要がある.
- 次に，棒 A の一番下にある円盤を棒 C に移動させる.
- 今度は移動させる円盤の数が $n-1$ 枚になる.

・円盤の数が 3 枚の時には,

- 始めに，$3-1$ 枚を棒 A から棒 B に移動する（1 回〜3 回）
- 次に，棒 A の円盤を棒 C に移動させる（4 回）
- この段階で移動させる円盤の数は $3-1$ 枚になる.
- 残り 2 枚を棒 C に移動させる（5 回〜7 回）

ハノイの塔（解法）

ハノイの塔を攻略するためには…

$n-1$ 枚の円盤を棒 A から棒 B へ移動する場合と棒 B から棒 A へ移動する場合は交互に行う.

- 始めの目標は「一番下の円盤を棒 C に移動する」こと.
- その為には上に載っている全ての円盤を，空いている棒へ移動させなければならない.
- 棒 A に円盤があるならば，棒 B が空いているので上に載っている円盤を全て棒 B へ移動させる.
- 次に棒 B の「一番下の円盤を棒 C に移動する」ことを考える.
- その為には上に載っている全ての円盤を，空いている棒 A へ移動させる.

  以降も同じ事を繰り返す.

ハノイの塔（プログラムでの表現）

関数の原型

```
void Hanoi(int n, char *from, char *work, char *dest)
```

※　この関数は,「第 2 引数（from）から第 4 引数（dest) に n 枚の円盤を移動させる」ということを意味している.

- n　…　移動させる円盤の枚数
- from …　移動元の棒の名前
- work …　作業用に使う棒の名前
- dest …　移動先の棒の名前

ハノイの塔（プログラムでの表現）

main 関数からの呼び出し

- 移動させる円盤は N 枚.
- 移動元 (from) を棒 A，移動先 (dest) を棒 C，作業棒 (work) を棒 B とすれば,

```
Hanoi(N,"A","B","C");
```

円盤を一枚移動させる（移動元：from，移動先：dest）

```
printf("Move a disk from %s to %s",from,dest);
```

一番下の円盤を移動させるまで

- $n-1$ 枚の円盤を移動元 (from) から作業棒 (work) に移動させる

```
Hanoi(n-1,from, dest, work);
```

演習問題 1

ファイル名: hanoi.c

ハノイの塔を攻略するプログラムを作成せよ.

- 表記は以下のようにせよ.

```
How many disks ? 3
Move the disc from A to C.
Move the disc from A to B.
Move the disc from C to B.
Move the disc from A to C.
Move the disc from B to A.
Move the disc from B to C.
Move the disc from A to C.
```

課題のヒント

```
#include <stdio.h>

void Hanoi(int n, char *from, char *work, char *dest){
    // Move n-1 desks from "from" to "work" via "dest".
    if(n>=2) Hanoi( ... , ... , ... , ...);

    printf("Move the disc from %s to %s.\n",from,dest);

    // Move n-1 desks from "work" to "dest" via "from".
    if(n>=2) Hanoi( ... , ... , ... , ...);
}

int main(void){
    int N;
    printf("How many disks ? ");
    scanf("%d",&N);
    Hanoi(N,"A","B","C");
    return 0;
}
```

解答例

```
#include <stdio.h>

void Hanoi(int n, char *from, char *work, char *dest){
    // Move n-1 desks from "from" to "work" via "dest".
    if(n>=2) Hanoi(n-1, from, dest, work);

    printf("Move the disc from %s to %s.\n",from,dest);

    // Move n-1 desks from "work" to "dest" via "from".
    if(n>=2) Hanoi(n-1, work, from, dest);
}

int main(void){
    int N;
    printf("How many disks ? ");
    scanf("%d",&N);
    Hanoi(N,"A","B","C");
    return 0;
}
```

解答例

```
#include <stdio.h>

void Hanoi(int n, char *from, char *work, char *dest){
    // Move n-1 desks from "from" to "work" via "dest".
    if(n>=2) Hanoi(n-1, from, dest, work);

    printf("Move the disc from %s to %s.\n",from,dest);

    // Move n-1 desks from "work" to "dest" via "from".
    if(n>=2) Hanoi(n-1, work, from, dest);
}

int main(void){
    int N;
    printf("How many disks ? ");
    scanf("%d",&N);
    Hanoi(N,"A","B","C");
    return 0;
}
```

解答例

```
#include <stdio.h>

void Hanoi(int n, char *from, char *work, char *dest){
    // Move n-1 desks from "from" to "work" via "dest".
    if(n>=2) Hanoi(n-1, from, dest, work);

    printf("Move the disc from %s to %s.\n",from,dest);

    // Move n-1 desks from "work" to "dest" via "from".
    if(n>=2) Hanoi(n-1, work, from, dest);
}

int main(void){
    int N;
    printf("How many disks ? ");
    scanf("%d",&N);
    Hanoi(N,"A","B","C");
    return 0;
}
```

演習問題 2

ファイル名: vecsum.c

64 個の要素を持つベクトルの総和をトーナメント方式（ペアワイズ）で計算するプログラムを作成せよ.

$$\mathrm{Sum} = (x_0 + x_1) + (x_2 + x_3) + \cdots + (x_{60} + x_{61}) + (x_{62} + x_{63})$$

- ベクトルを 1 次元配列で宣言し，それぞれの要素に 0 以上 1 未満の乱数を代入せよ.
- 再帰呼び出しを用いて，トーナメント方式でベクトルの総和を計算せよ.
- 端末の表記は以下のようにせよ.

```
Sum is ??
```

課題のヒント

例えば，8 個の要素を持つベクトルの総和をトーナメント方式（ペアワイズ）で計算すると，

$$\mathrm{Sum} = (x_0 + x_1) + (x_2 + x_3) + (x_4 + x_5) + (x_6 + x_7).$$

- それぞれのペアで足し算を繰り返し，総和を求める．
- これを再帰呼び出しでプログラムすると...

vecsum 関数

```
double vecsum(double *x, int n1, int n2)
```

※　この関数は「配列 x の n1 番目から n2 番目までを加える」．

- n1=n2 のとき値 x[n1] を返す．
- それ以外は「n1 から (n1+n2)/2 までの和」＋「(n1+n2)/2+1 から n2 までの和」を返す．

課題のヒント

例えば，8 個の要素を持つベクトルの総和をトーナメント方式（ペアワイズ）で計算すると，

$$\mathrm{Sum} = y_0 + y_1 + y_2 + y_3, \quad (y_k = x_{2k} + x_{2k+1}).$$

- それぞれのペアで足し算を繰り返し，総和を求める．
- これを再帰呼び出しでプログラムすると...

vecsum 関数

```
double vecsum(double *x, int n1, int n2)
```

※　この関数は「配列 x の n1 番目から n2 番目までを加える」．

- n1=n2 のとき値 x[n1] を返す．
- それ以外は「n1 から (n1+n2)/2 までの和」＋「(n1+n2)/2+1 から n2 までの和」を返す．

課題のヒント

例えば，8 個の要素を持つベクトルの総和をトーナメント方式（ペアワイズ）で計算すると，

$$\mathrm{Sum} = (y_0 + y_1) + (y_2 + y_3).$$

- それぞれのペアで足し算を繰り返し，総和を求める．
- これを再帰呼び出しでプログラムすると...

vecsum 関数

```
double vecsum(double *x, int n1, int n2)
```

※　この関数は「配列 x の n1 番目から n2 番目までを加える」．

- n1=n2 のとき値 x[n1] を返す．
- それ以外は「n1 から (n1+n2)/2 までの和」＋「(n1+n2)/2+1 から n2 までの和」を返す．

課題のヒント

例えば，8 個の要素を持つベクトルの総和をトーナメント方式（ペアワイズ）で計算すると，

$$\mathrm{Sum} = z_0 + z_1, \quad (z_k = y_{2k} + y_{2k+1}).$$

- それぞれのペアで足し算を繰り返し，総和を求める．
- これを再帰呼び出しでプログラムすると...

vecsum 関数

```
double vecsum(double *x, int n1, int n2)
```

※　この関数は「配列 x の n1 番目から n2 番目までを加える」．

- n1=n2 のとき値 x[n1] を返す．
- それ以外は「n1 から (n1+n2)/2 までの和」＋「(n1+n2)/2+1 から n2 までの和」を返す．

課題のヒント

例えば，8 個の要素を持つベクトルの総和をトーナメント方式（ペアワイズ）で計算すると，

$$\mathrm{Sum} = z_0 + z_1, \quad (z_k = y_{2k} + y_{2k+1}).$$

- それぞれのペアで足し算を繰り返し，総和を求める．
- これを再帰呼び出しでプログラムすると...

vecsum 関数

```
double vecsum(double *x, int n1, int n2)
```

※　この関数は「配列 x の n1 番目から n2 番目までを加える」．

- n1=n2 のとき値 x[n1] を返す．
- それ以外は「n1 から (n1+n2)/2 までの和」＋「(n1+n2)/2+1 から n2 までの和」を返す．

課題のヒント

例えば，8 個の要素を持つベクトルの総和をトーナメント方式（ペアワイズ）で計算すると，

$$\mathrm{Sum} = z_0 + z_1, \quad (z_k = y_{2k} + y_{2k+1}).$$

- それぞれのペアで足し算を繰り返し，総和を求める．
- これを再帰呼び出しでプログラムすると...

vecsum 関数

```
double vecsum(double *x, int n1, int n2)
```

※　この関数は「配列 x の n1 番目から n2 番目までを加える」．

- n1=n2 のとき値 x[n1] を返す．
- それ以外は「n1 から (n1+n2)/2 までの和」＋「(n1+n2)/2+1 から n2 までの和」を返す．

解答例

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

double vecsum(double* x, int n1, int n2){
    if(n1==n2){
        return x[n1];
    }
    else{
        return vecsum(x,n1,(n1+n2)/2)+vecsum(x,(n1+n2)/2+1,n2);
    }

}

int main(void){
    int i,n=64;
    double x[64];
    srand(2015);
    for(i=0;i<n;i++) x[i]=(double) rand()/RAND_MAX;

    printf("sum is %f\n",vecsum(x,0,n-1));
    return 0;
}
```