Intermediate Programming

— Recursion (2) —

Waseda Univ.

Today's Topics

- Examples of the recursive call
  - Tower of Hanoi
  - Pairwise summation

Tower of Hanoi

- The Tower of Hanoi is a mathematical game. It consists of three rods (A, B, C), and n disks of different sizes that can slide onto any rod.
- The game starts with the disks in a neat stack in ascending order of size on the rod A. The objective of the game is to move the entire stack to the rod C, obeying the following simple rules:
  - Only one disk can be moved at a time.
  - Each disk can only be moved if it is the uppermost disk on a stack.
  - No disk can be placed on top of a smaller disk.

- The minimum number of moves is $2^n - 1$.

Tower of Hanoi

- The first move is taking the upper disk on the rod A and placing it on top of another rod, but...
- We cannot determine which rod (B or C) is better for moving this disk.
- The recursive procedure is effective for this game.

↓

Breaking the problem down into smaller problems.

Tower of Hanoi

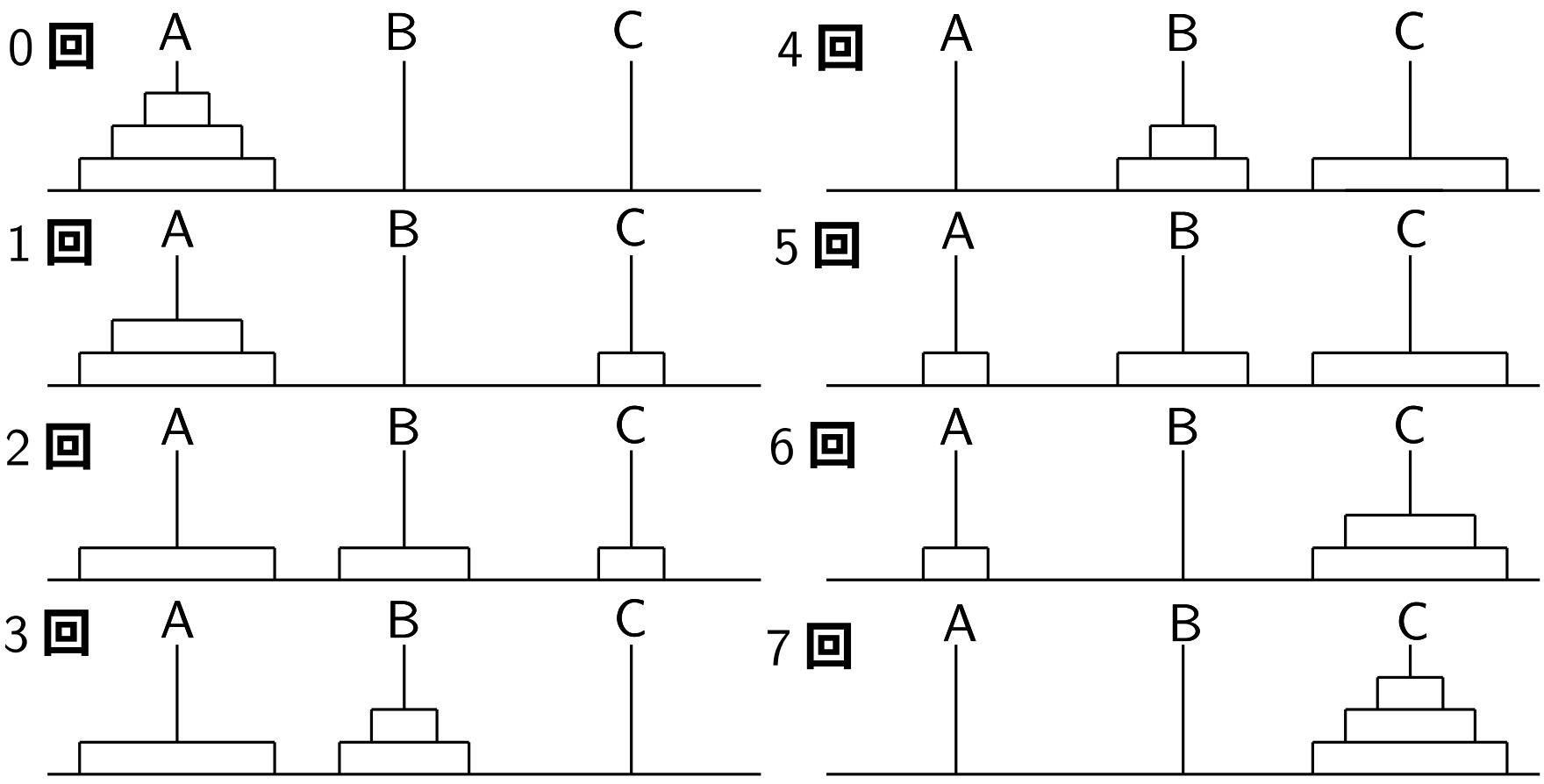
- The first move is taking the upper disk on the rod A and placing it on top of another rod, but...
- We cannot determine which rod (B or C) is better for moving this disk.
- The recursive procedure is effective for this game.

↓

Breaking the problem down into smaller problems.

Tower of Hanoi

- Moves for 3 disks



Tower of Hanoi

To move n disks from the rod A to the rod C

- Move $n-1$ disks from A to B. This leaves nth disk alone on A.
- Move the nth disk from A to C.
- Move $n-1$ disks from B to C so they sit on the nth disk.

For 3 disks,

- Move $3-1$ disks from A to B. (1〜3)
- Move the 3rd disk from A to C. (4)
- The number of disk is $3-1$.
- Move 2 disks from B to C. (5〜7)

The above is a recursive algorithm!

Tower of Hanoi

Recursive function

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

※ This function means that we move n disks from the second parameter "from" to the fourth parameter "dest" via the third parameter "work".

Tower of Hanoi

Call Hanoi function from the main function

- The total number of disks are N.
- Let "from" be A, "dest" C, and "work" B.

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

Move the n th disk from "from" to "dest"

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

Move $n-1$ disks from "from" to "work" by recursive call

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

Exercise 1

Exercise: hanoi.c

Write a program that solves Tower of Hanoi.

- Output of this program is as follows:

```
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.
```

Hint

```
#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;
}
```

Answer

```
#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;
}
```

Answer

```
#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;
}
```

Answer

```
#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;
}
```

Exercise 2

Exercise: vecsum.c

Let $\{x_i\}$ be a sequence of 64 numbers. Write a program that calculates the sum of the sequence by using pairwise summation.

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

- Declare a sequence by using the array and assign random real values from 0 to 1.
- Calculate the sum of the sequence by using pairwise summation.
- Output of this program is as follows:

```
Sum is ??
```

Hint

For example, the sum of a sequence of 8 numbers are given by

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

- Pairwise summation of the sequence works by recursively breaking the sequence into two halves.

Recursive function

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

※ This fuction means that we calculate the sum of the array x from the index n1 to n2.

- If n1=n2, return the value of x[n1].
- Otherwise, we add "the sum of the array x from the index n1 to (n1+n2)/2" to "the sum of the array x from the index (n1+n2)/2+1 to n2" .

Hint

For example, the sum of a sequence of 8 numbers are given by

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

- Pairwise summation of the sequence works by recursively breaking the sequence into two halves.

Recursive function

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

※ This fuction means that we calculate the sum of the array x from the index n1 to n2.

- If n1=n2, return the value of x[n1].
- Otherwise, we add "the sum of the array x from the index n1 to (n1+n2)/2" to "the sum of the array x from the index (n1+n2)/2+1 to n2" .

Hint

For example, the sum of a sequence of 8 numbers are given by

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

- Pairwise summation of the sequence works by recursively breaking the sequence into two halves.

Recursive function

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

※ This fuction means that we calculate the sum of the array x from the index n1 to n2.

- If n1=n2, return the value of x[n1].
- Otherwise, we add "the sum of the array x from the index n1 to (n1+n2)/2" to "the sum of the array x from the index (n1+n2)/2+1 to n2" .

Hint

For example, the sum of a sequence of 8 numbers are given by

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

- Pairwise summation of the sequence works by recursively breaking the sequence into two halves.

Recursive function

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

※ This fuction means that we calculate the sum of the array x from the index n1 to n2.

- If n1=n2, return the value of x[n1].
- Otherwise, we add "the sum of the array x from the index n1 to (n1+n2)/2" to "the sum of the array x from the index (n1+n2)/2+1 to n2" .

Hint

For example, the sum of a sequence of 8 numbers are given by

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

- Pairwise summation of the sequence works by recursively breaking the sequence into two halves.

Recursive function

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

※ This fuction means that we calculate the sum of the array x from the index n1 to n2.

- If n1=n2, return the value of x[n1].
- Otherwise, we add "the sum of the array x from the index n1 to (n1+n2)/2" to "the sum of the array x from the index (n1+n2)/2+1 to n2".

Answer

```
#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;
}
```

Summary

- Examples of the recursive call
  - Tower of Hanoi
  - Pairwise summation