PDS Lab

Section 16

Autumn-2018

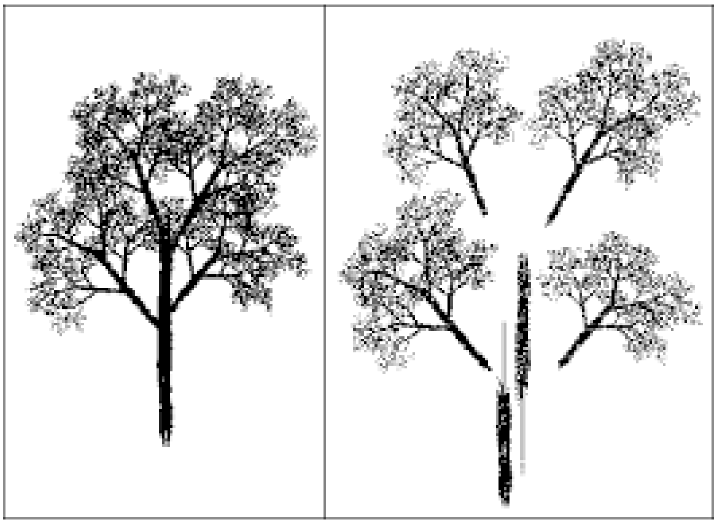
Tutorial 6

Recursions

What is Recursion?

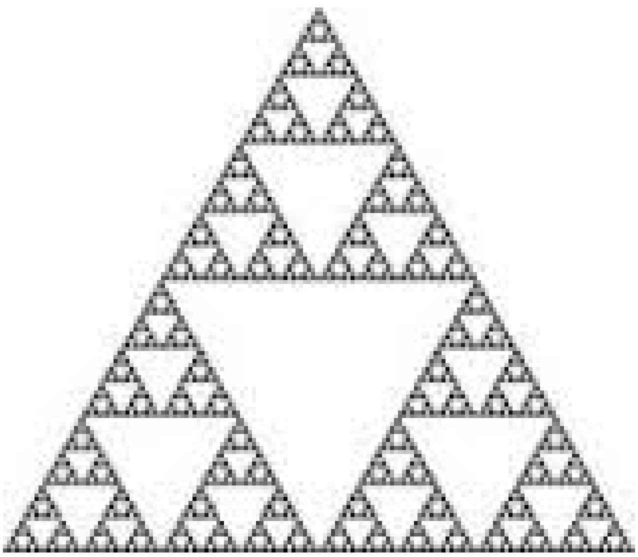
A ting by which it defines itself.

Example 1



Tree

Example 2



Object in Fractal Geometry

Recursive Function in C

- A process by which a function calls itself repeatedly.

Example 3

Calculation of n!

n! = n×(n-1)×(n-2) ×…×3×2×1

= n×(n-1)!

```
#include <stdio.h>
int fact(int n)
{
        if (n == 0)
            return 1;
        else
            return (n * fact(n-1));
}

void main()
{
        int x;
        scanf("%d", &x);
        printf ("Factorial of %d is:
        %d", x, fact(x));
}
```

fact(0) = 1 // Termination condition

fact(n) = n × fact(n-1), if n > 0

Direct Recursion

- When a function f(…) calls f(…).

Cyclically in a chain recursion

- f1(…) calls f2(…) , f2(…) calls f3(…) . . . fi(…) calls f1(…)

Example 4

Calculation of Recurrence Relation

`T(n) = n+2T(n-1)` given that `T(1) = 0`

What is the value of T(100)?

Example 5

Calculation of Greatest Common Divisor (GCD)

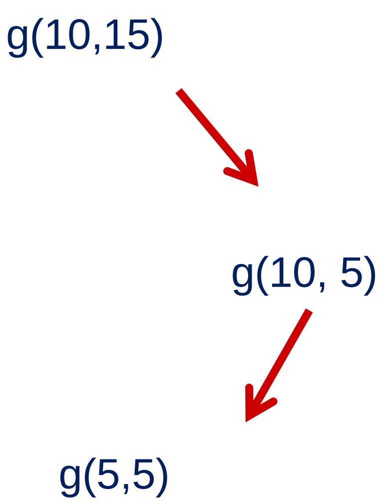
```
gcd(10, 15) = 5     gcd(11, 13) = 1
```

Recursive definition

```
gcd(m,m) = m;
gcd(m,n) = gcd(m-n, n)  if m > n
         = gcd(m, n-m)  else
```

```
#include <stdio.h>
int gcd(int m, int n)
{
        if (m == n)
            return m;
        else
            if (m > n) return (gcd(m-n, n);
                 else return gcd(m, n-m);
}

void main()
{
     int x, y;
     scanf(“x = %d, y = %d”, &x, &y);
     printf (“GCD of %d and %d is %d”, x,
y, gcd(x, y));
}
```



Some More Examples

Example 6

```
Sum = 1 + 2 + 3 + …+ (n-1) + n
    = n + (n-1) + … + 3 + 2 +1

sum(1) = 1
sum(n) = N + sum(n-1)
```

Example 7

What the following function does?
Check the function for the following.

a) gcd(12, 16)
b) gcd(17, 11)
c) gcd(2, 2)
d) gcd(0, 5)

```
int gcd(int m, int n)
{
   if ((m == n)|| (n == 0))
      return m;
   else
      if(n > m) return (gcd(n,m);
      else return gcd(m, n%m);
}
```

```
gcd(m,n) = m, if (m = n) or n = 0;
         = gcd(n,m) if n>m
         = gcd(m%n,m)
```

Example 8

Following are the series called Fibanacci secuence, `n`-th number is the sum of the `(n-1)`-th and `(n-2)`-th numbers.

O, 1, 1, 2, 3, 5, 8, 13, 21, 34, ….

Recursively it can be defined as follows.

```
f(0)  =  0
f(1)  =  1
f(n)  = f(n-1)+f(n-2),if n>1
```

The corresponding recursive function is given by

```
int f(int n)
{
    if (n < 2)
        return (n);
    else
        return (f(n-1) + f(n-2));
}
```

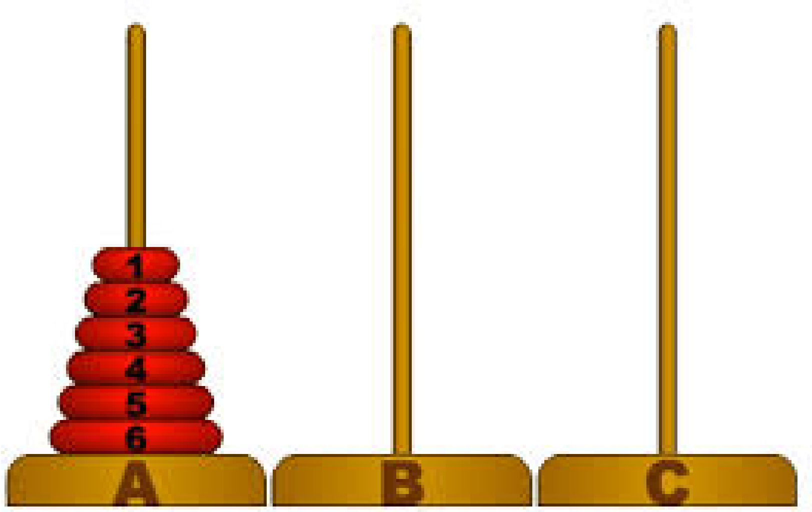
Example 9

Expand the following recurrence relation and express it in terms of n. Assume that $n = 2^k$ for some $k \geq 0$ and $T(1) = 1$.

$$T(n) = 1 + T\left(\frac{n}{2}\right)$$

Example 10

Tower of Hanoi problem



Tower of Hanoi is a mathematical puzzle where we have three pegs A, B and C and n disks all of are of unequal sizes. The objective of the puzzle is to move the entire stack from one peg to another peg with a minimum number of disk movement and obeying the following rules:

a) Initially all the disks are stacked on the peg A.
b) Required to transfer all the disks to the peg C.
c) Only one disk can be moved at a time.
d) A larger disk cannot be placed on a smaller disk.
e) C peg is used for temporary storage of disks.

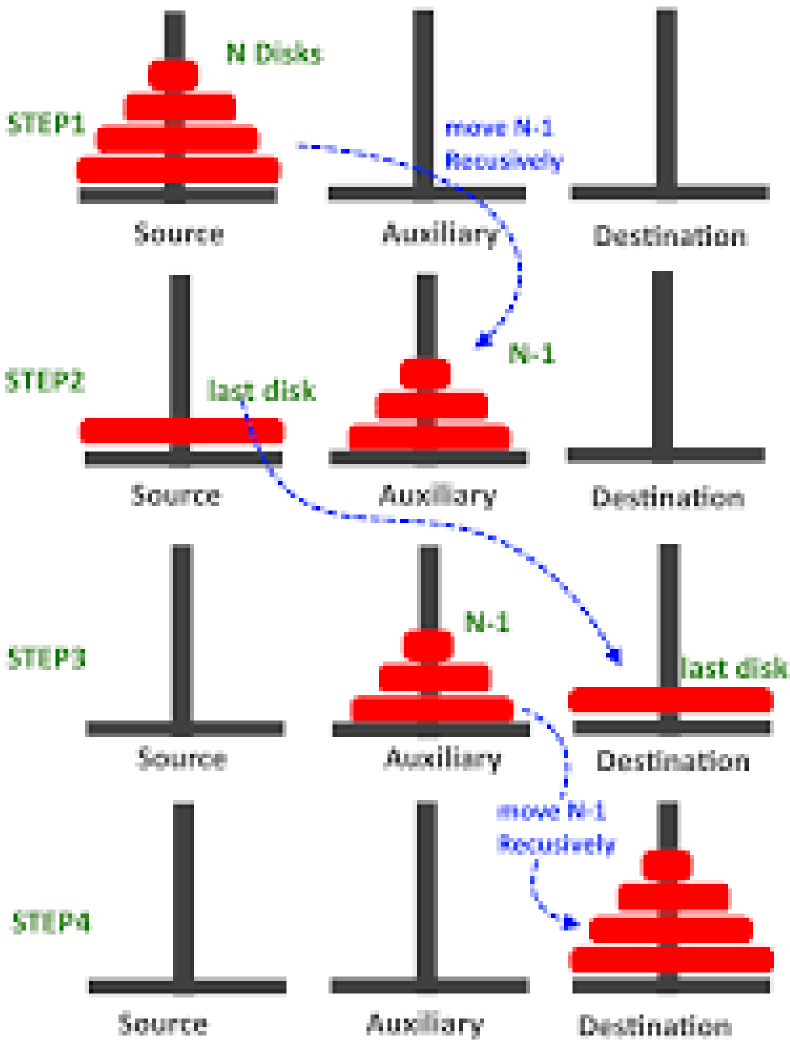
There are some sample solutions.

3
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C

4
Move disk 1 from A to B
Move disk 2 from A to C
Move disk 1 from B to R
Move disk 3 from A to B
Move disk 1 from C to L
Move disk 2 from C to B
Move disk 1 from A to B
Move disk 4 from A to C
Move disk 1 from B to C
Move disk 2 from B to A
Move disk 1 from C to A
Move disk 3 from B to C
Move disk 1 from A to B
Move disk 2 from A to C
Move disk 1 from B to C

Recursive statement of the general problem of n disks

- **Step 1:**
  - Move the top (n-1) disks from A to B
- **Step 2:**
  - Move the largest disk from A to C.
- **Step 3:**
  - Move the (n-1) disks from B to C.



How many number of moves for n disks is required?

```c
#include <stdio.h>
void move(int n, char A, char B, char C);
int main()
{
    int n;   /* Number of disks */
    scanf (“%d”, &n);
    move (n, ‘A’, ‘B’, ‘C’);
    return 0;
}
void move (int n, char A, char B, char C)
{
    if (n > 0) {
                 move (n-1, A, C, B);
                 printf (“Move disk %d from %c to %c \n”, n,
                 A, C);
                 move (n-1, B, C, A);
                 }
return;
}
```

Can you write the recurrence relation for the number of movements required for the Tower of Hanoi problem with n disks?

Tutorial Problems

Problem 1

Formulate each of the following algebraic formulas in recursive forms.

a) `sum = a[0] + a[1] + a[2] + … + a[size]`, where `a[size]` is an array of integer with size `size`.

b) $y = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x4}{4!} - \cdots (-1)^n \frac{x^n}{n!}$

c) $y = x^n$, where x is any floating point number and n is a positive integer.

Problem 2

What does the following foomatic programs return?

a. ---

```
int foo5 ( unsigned int n )
{
   if (n == 0) return 0;
   return 3*n*(n-1) + foo5(n-1) + 1;
}
```

b. ---

```
int foo6 ( char A[] , unsigned int n )
 {
    int t;

    if (n == 0) return 0;
    t = foo6(A,n-1);
    if ( ((A[n-1]>='a') && (A[n-1]<='z')) ||
         ((A[n-1]>='A') && (A[n-1]<='Z')) ||
         ((A[n-1]>='0') && (A[n-1]<='9')) )
       ++t;
```

```
    return t;
}
```

c. ---

```
int foo7 ( unsigned int a , unsigned int b )
 {
    if ((a == 0) || (b == 0)) return 0;
    return a * b / bar7(a,b);
 }

 int bar7 ( unsigned int a , unsigned int b )
 {
    if (b == 0) return a;
    return bar7(b,a%b);
 }
```

d. ---

```
int foo8 ( unsigned int n )
 {
    if (n == 0) return 0;
    if (n & 1) return -1;
    return 1 + bar8(n-1);
 }

 int bar8 ( int n )
 {
    if (!(n & 1)) return -2;
    return 2 + foo8(n-1);
 }
```

Problem 3

Write a function to recursively compute the sum of digits of a positive integer. The function has to be recursive?

Problem 4

Write a Function to recursively compute the harmonic mean of an array of numbers. In the main function, create an array of size 10. Input integers from the user till a negative number is given as input or the 10 elements have been filled up. Find the harmonic mean of the elements of this array?

Hint: It is the reciprocal of the arithmetic mean of the reciprocals of the given set of observations. That is

$$H = \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}$$

Problem 5

Find the product of *n* floating point numbers. The numbers should be read from the keyboard. You should not use any looping construct.

Hint: use recursion and decide a suitable sentinel for termination of recursion.

Problem 6

All possible combinations of size *r* from a set of *n* ($n \le r$) distinct objects is a known combinatorics problem. Number of such combinations is given by ${}^{n}C_{r} = \frac{n!}{r!(n-r)!}$

For example, suppose

n-objects: a b c

Then ${}^{3}C_{2} = 3$ combinations are: ab bc ca

The following function calculates the all possible combinations from *n* “distinct” objects taking *r* objects at a time.

```
#include <stdio.h>
int count;
int objects[10];
int tempComb[10];

void nCrFind(int n, int r, int *object)
{
    count = 0;
     // Read value of n
    // Read value of r
    // Read n elements in the array “objects”

     fill(0,0, n, r);   //Call the recursive function
}
```

The following routine places the objects to get various combinations

```
void fill(int i, int j, int n, int r)
{
    int k, m;
    if(i < r) {
        for(k = j; k <= n-r;+j; k++) {
            if (k < n) {
                tempComb[i] = objects[k];
                fill(i+1; k+1, n, r);
            }
        }
        else {
            printf(“\n Combination %d:”, ++count);
            for(m=0; m<r; m++)
                printf(“%d”tempComb[m]);
            printf(“\n”);
        }
            return;
```

```
}
```

Problem 7

Permutation of *n* "distinct" objects is the all possible arrangements (*n!*) of the objects. For example for 3 objects `a b c`, 3! = 6 arrangements are

```
    abc     acb     bca     bac     cab     cba
```

The following function attempts to print all such arrangements given *n* "distinct" objects stored in an array *A*.

```
void permute(int n, int *A)
{
    int B[];
    if (n == 0) return;
    B = (int *) malloc((n-1)*sizeof(int));
   for (i=0; i<n; i++) {
        // Print the array A
        // Copy element A[0] to A[n-1] except
            A[i] into the array B
       permute(n-1, B);
    }
}
```

Important links:

http://cse.iitkgp.ac.in/~dsamanta/courses/pds/index.html