CS203 Programming with Data Structures

Recursion

Chengyu Sun
California State University, Los Angeles

Method

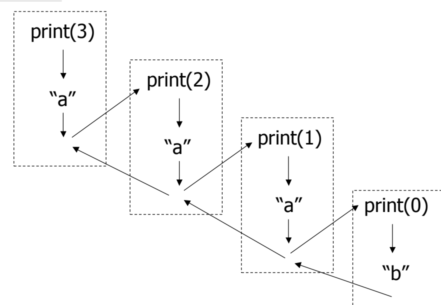
- Header
  - Access modifier
  - `static`
  - Return type
  - Name
  - Parameter list
- Body
- Signature

Recursion

- A method calls itself

```
void print( int n )
{
   if( n <= 0 ) System.out.println("b");
   else
   {
      System.out.print("a");
      print(n-1);
   }
}
```

Recursive Process



Ending Condition

- When the recursion should stop
- To avoid infinite recursion, make sure the ending condition
  - Exists
  - Reachable
  - Comes before the recursive call

Simple Recursion Examples

- Factorial
- Search

String Permutation

- Output all the permutations of n characters
  - E.g. "abc"
    - abc, acb
    - bac, bca
    - cab, cba
- How do we reduce the problem of n characters to the problem of n-1 characters??

Fibonacci Series

- 0, 1, 1, 2, 3, 5, 8, 13, 21, …
- Definition
  - fibonacci(0) = 0
  - fibonacci(1) = 1
  - fibonacci(n) = fibonacci(n-1)+fibonacci(n-2)

Recursive Fibonacci

```
int fibonacci( int n )
{
    if( n == 0 ) return 0;
    else if( n==1 ) return 1;
    else
        return fibonacci(n-1) + fibonacci(n-2);
}
```

Non-recursive Fibonacci

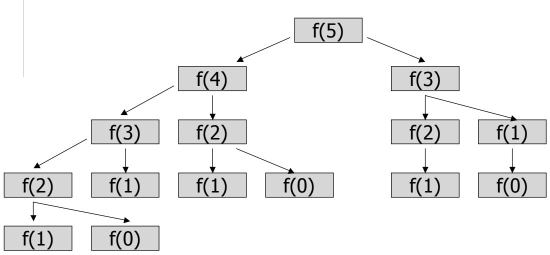
```
int fibonacci( int n )
{
    if( n == 0 || n==1 ) return n;

    int last1 = 1, last2 = 0, fibo;
    for( int i=2 ; ?? ; ++i )
    {
        fibo = last1+last2;
        ??
    }

    return fibo;
}
```

Recursion vs. Non-recursion

- Less code != more efficient



Timing

- The best way to appreciate a good algorithm is to see how fast it runs
- And time it
- `System.currentTimeMillis()`
- `System.nanoTime()`

When Can We Use Recursion?

- A problem itself is recursively defined
  - Fibonacci   f(n) = f(n-1) + f(n-2)
  - Tree
    - A tree has a root
    - Each child of the root is also a tree
- A problem of size n can be reduced to a problem of size less than n
  - Factorial: n   n-1
  - Sort: n   n-1
  - Binary search: n   n/2

When Should We Use Recursion?

- When the homework problem says so
- When speed of code development takes precedence over code efficiency
- When the problem is naturally recursive
  - Fibonacci Series
- When the non-recursive solution is much harder
  - Hanoi tower
  - Solving maze