CS 240

Data Structures and Algorithms I

Alex Vondrak

ajvondrak@csupomona.edu

November 16, 2011

Iterative Algorithms

Iteration:

- Set up initial state
- Repeatedly perform a process until it reaches a desired goal

Example (Length of a Linked List)

```
public int length() {
   Node<E> current = head;
   int count = 0;
   while(current != null) {
      count++;
      current = current.link;
   }
   return count;
}
```

Recursive Algorithms

Recursion:

- Start with base cases
- Use recursive cases to simplify input down to a base case

Example (Length of a Linked List)

```
public int length() { return length(head); }

private int length(Node<E> current) {
   if (current == null)
      return 0;
   return 1 + length(current.link);
}
```

A Closer Look At Recursion

Internally, recursive methods are handled by stacks of call frames (or activation records):

- Every time a method is invoked, we allocate space to store
  - Input parameters
  - The return address
  - Local variables
- Upon allocating the frame, we push it to the call stack
- When we finish executing the method we
  - Restore certain portions of memory
  - Pop the activation record
  - Jump to the code at the frame's return address

Patterns of Recursion

Definition (Tail Recursion)

A recursive call is in the tail position if it is the return value of the method. If all calls are in the tail position, a method is said to be tail-recursive.

Example (Length)

Our recursive version of `length` was not tail-recursive.

```
public int length() { return length(head); }

private int length(Node<E> current) {
    if (current == null)
        return 0;
    return 1 + length(current.link);
}
```

Patterns of Recursion

Definition (Tail Recursion)

A recursive call is in the tail position if it is the return value of the method. If all calls are in the tail position, a method is said to be tail-recursive.

Example (Tail-Recursive Length)

```
public int length() { return length(head, 0); }

private int length(Node<E> current, int total) {
   if (current == null)
      return total;
   return length(current.link, total + 1);
}
```

Patterns of Recursion

Definition (Fold)

A fold is a recursive way to replace the “structural” components of a data structure with desired functions and values. Also known as *reduce*, *accumulate*, *compress*, or *inject*.
Folds may either be left-associative or right-associative.

Example (Right Fold)

The linked list `( 1 2 3 )` can be built up by

```
new Node<Integer>(1,
  new Node<Integer>(2,
    new Node<Integer>(3, null)))
```

We can think of a right fold as replacing the **`new`** `Node<Integer>`s with a specific function, and **`null`** with a specific value.