

Computer Science I

Professor Tom Ellman
Lecture 11

When is the Accumulator Method Useful?

- When the order of members in a list need to be reversed. (REVERSE and DIGITS).
- When more than one value is being returned. (SUMS-BELOW-ABOVE).
- When ordinary recursion leads to multiple calls of a procedure with the same parameters. (FIBONACCI).

Finding Two Sums at Once

```
(define sums-below-above (lambda (lst bound) ...?...))
```

```
Welcome to DrScheme.  
> (sums-below-above '(8 1 4 2 9) 5))  
(7 17)  
> (sums-below-above '(8 1 4 2 9) 0))  
(0 24)  
> (sums-below-above '(8 1 4 2 9) 9))  
(24 0)  
> (sums-below-above '() 7))  
(0 0)
```

SUMS-BELOW-ABOVE (Flat Recursion)

```
(define sums-below-above  
  (lambda (lst bound)  
    (cond ((null? lst) (list 0 0))  
          ((<= (car lst) bound)  
           (list (+ (car lst)  
                   (car (sums-below-above (cdr lst) bound)))  
                 (cadr (sums-below-above (cdr lst) bound))))  
          (else (list (car (sums-below-above (cdr lst) bound))  
                      (+ (car lst)  
                          (cadr (sums-below-above (cdr lst) bound))))))))
```

SUMS-BELOW-ABOVE (Accumulator Method)

```
(define sums-below-above  
  (lambda (lst bound) (sba-helper lst bound 0 0)))  
  
(define sba-helper  
  (lambda (lst bound sum-below sum-above)  
    (cond ((null? lst) (list sum-below sum-above))  
          ((<= (car lst) bound)  
           (sba-helper (cdr lst)  
                        bound  
                        (+ (car lst) sum-below)  
                        sum-above))  
          (else (sba-helper (cdr lst)  
                             bound  
                             sum-below  
                             (+ (car lst) sum-above))))))
```

Rabbit Breeder's Problem

- Start with a pair of newborn rabbits. (1 Pair)
- At one month, they are too young to breed. (1 Pair.)
- At two months, they produce one pair of offspring. (2 Pairs).
- At three months, they produce another pair of offspring. (3 Pairs).
- At four months, each pair alive at two months produces a new pair of offspring. (5 pairs.)
- At N months, each pair alive at N-2 months produces a new pair of offspring.

Rabbit Breeding Records

Month	Pairs	Total
0	xx	1
1	xx	1
2	xx xx	2
3	xx xx xx	3
4	xx xx xx xx xx	5
5	xx xx xx xx xx xx xx xx	8

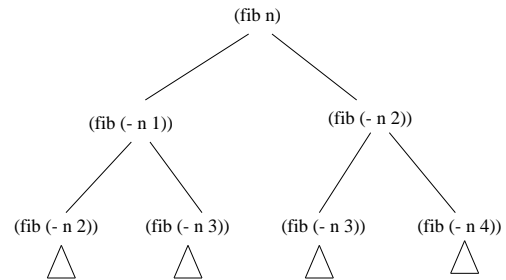
A Recursive Formula for the Fibonacci Numbers

$$\text{Fibonacci}(n) = \begin{cases} 1 & \text{If } n = 0. \\ 1 & \text{If } n = 1. \\ \text{Fibonacci}(n-1) + \text{Fibonacci}(n-2) & \end{cases}$$

Recursive Procedure for Computing Fibonacci(N)

```
(define fib
  (lambda (n)
    (cond ((= n 0) 1)
          ((= n 1) 1)
          (else (+ (fib (- n 1)) (fib (- n 2)))))))
```

Fib Call Tree



A Cause for Concern

- An evaluation of (fib n) leads to two evaluations of (fib (- n 3)).
- ...And many evaluations of (fib i) for smaller values of i.

Counting the Number of Calls to FIB in Computing (FIB N)

```
(define fib-calls
  (lambda (n) (cdr (fib-calls-helper n))))

(define fib-calls-helper
  (lambda (n)
    (cond ((= n 0) (cons 1 1))
          ((= n 1) (cons 1 1))
          (else (fib-calls-packager (fib-calls-helper (- n 1))
                                    (fib-calls-helper (- n 2)))))))

(define fib-calls-packager
  (lambda (f1 f2)
    (cons (+ (car f1) (car f2)) (+ 1 (cdr f1) (cdr f2)))))
```

Costs of Evaluating (FIB N)

N	(FIB N)	Calls
0	1	1
1	1	3
2	2	5
4	5	15
9	55	177
24	75025	242785

The FIB procedure with Two Accumulation Parameters

```
(define fib
  (lambda (n)
    (if (= n 0)
        1
        (fib-helper (- n 1) 1 1))))

(define fib-helper
  (lambda (remaining i j)
    (if (= remaining 0)
        j
        (fib-helper (- remaining 1) j (+ i j)))))
```

Behavior of FIB Procedure Using the Accumulator Method

```
(fib 4)
|
(fib 3 1 1)
|
(fib 2 1 2)
|
(fib 1 2 3)
|
(fib 0 3 5)
```

Counting the Number of Calls to FIB in Computing (FIB N)

```
(define fib-calls
  (lambda (n)
    (if (= n 0)
        1
        (fib-calls-helper (- n 1) 1 1 1))))

(define fib-calls-helper
  (lambda (remaining i j calls)
    (if (= remaining 0)
        (+ 1 calls)
        (fib-calls-helper (- remaining 1) j (+ i j) (+ calls 1)))))
```

Costs of Evaluating (FIB N)

N	(FIB N)	Calls
0	1	1
1	1	2
2	2	3
4	5	5
9	55	10
24	75025	25

Comparing Costs of Evaluating FIB with and without Accumulator

N	Calls without Accumulator	Calls with Accumulator
0	1	1
1	3	2
2	5	3
4	15	5
9	177	10
24	242785	25