

Discrete Mathematical Structures

CS 3233 Lecture 29

Prof. William Winsborough

November 14, 2005

Recursive Programming

- Recursive Modular Exponentiation
 - Compute $b^n \bmod m$, where b , n , and m are integers with $m \geq 2$, $n \geq 0$, and $1 \leq b < m$

procedure mpower(b, n, m : integers as above)

if $n = 0$ then

 mpower(b, n, m) = 1

else if n is even then

 mpower(b, n, m) = mpower($b, n/2, m$)² mod m

else

 mpower(b, n, m) =

 ((mpower($b, \lfloor n/2 \rfloor, m$)² mod m)($b \bmod m$)) mod m

Correctness of mpower

- Use strong induction to show that $\text{mpower}(b,n,m) = b^n \bmod m$ for all $n \in \mathbb{N}$
- Basis: When $n = 0$, $\text{mpower}(b,n,m) = 1$, which is correct
- Step: assume $\text{mpower}(b,j,m) = b^j \bmod m$ for all integers $0 \leq j < k$ and show for k
 - Even k : $\text{mpower}(b,k,m) = \text{mpower}(b,k/2,m)^2 \bmod m = (b^{k/2} \bmod m)^2 \bmod m = b^k \bmod m$
 - Odd k : $\text{mpower}(b,k,m) = ((\text{mpower}(b, \lfloor k/2 \rfloor, m)^2 \bmod m \bmod m)(b \bmod m)) \bmod m = b^{2\lfloor k/2 \rfloor + 1} \bmod m = b^k \bmod m$

Recursive Binary Search

```
procedure binary search(x, i, j)
  m :=  $\lfloor (i+j)/2 \rfloor$ 
  if  $x = a_m$  then
    location := m
  else if ( $x < a_m$  and  $i < m$ ) then
    binary search (x, i, m - 1)
  else ( $x > a_m$  and  $j > m$ ) then
    binary search (x, m + 1, j)
  else location := 0
```

Merge Sort

procedure mergesort($L = a_1, a_2, \dots, a_n$)

If $n > 1$ then

$m := \lfloor n/2 \rfloor$

$L_1 = a_1, a_2, \dots, a_m$

$L_2 = a_{m+1}, a_{m+2}, \dots, a_n$

$L = \text{merge}(\text{mergesort}(L_1), \text{mergesort}(L_2))$

procedure merge(L_1, L_2 : lists)

$L := \text{emptylist}$

while L_1 and L_2 are both nonempty

remove the first element of either L_1 or L_2 , whichever element is smaller and append to L

if this makes one list empty, append the other list to L

Complexity of Merge Sort

- A sorted list of length m and a sorted list of length n can be merged using at most $m+n-1$ comparisons
- Assume the length of L , n , is a power of two, $n = 2^m$
- There are 2^{k-1} lists at level $k-1$ each of length 2^{m-k+1} . These are split into 2^k lists at level k , each with 2^{m-k} elements
 - At the end of the splitting, we have 2^m lists of length 1 at level m
 - Working backwards through the levels, we then merge 2^k lists of length 2^{m-k} into 2^{k-1} lists at a cost of $2^{k-1}(2^{m-k+1}-1)$ comparisons
 - Summing these over m levels yields $n \log n - n + 1$ comparisons for merge sort