

Analysis of Algorithms

- Elementary operations
 - $O(1)$
 - addition, subtraction, multiplication, division of limited-size numbers
 - branches, gotos, logical operations
 - 5 elementary operations: $5 O(1) = O(1)$
- Sequence of operations
 - $O(\text{firstOperation}) + O(\text{secondOperation})$
 - use maximum rule!
- If statement
 - $O(\text{then-branch}) + O(\text{else-branch})$
- Procedures
 - add up for every call
 - unless recursive

© 2005. 324-1-41

Analysis of Algorithms: Loops

- If number of repetitions does not depend on input:
 - add them all up
- If it is based on input:
 - find function $f(n)$ that gives number of repetitions
 - $O(f(n))$
- How to find that function?
 - find a variable that decreases in each repetition
 - and that is 0 when loop ends
 - express start value as a function of n (n is the number of input objects)
- Example: iterative Fibonacci algorithm
 - variable: $n-k$, goes from $n-1$ to 0
 - $O(n)$
- What about while-loops?

© 2005. 324-1-42

Barometers

- An elementary operation that is executed at least as often as any other
- Then:
 - order of algorithm = order of times the barometer is executed
- Useful for nested loops

© 2005. 324-1-43

Example: Selection Sort

```
sort( Comparable[] list ) {
    for( int i = 0; i < list.length; i++ ) {
        int minimum = i;
        for( j = i+1; j <= list.length; j++ ) {
            if( list[ j ] < minimum ) {
                minimum = j;
            }
        }
        swap( list, i, minimum );
    }
}
```

- if() is barometer
- $O(\sum \sum 1) = O(n(n-1)/2) = O(n^2)$

© 2005. 324-1-44

Example: Euclid's Algorithm

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

- if $m \geq n$: $m \% n < m/2$
- How often is gcd called?
 - initial: m, n
 - 1st call: $n, m\%n$
 - 2nd call: $m\%n, n\%(m\%n)$
- So after 2 calls: $m \rightarrow m\%n$
 - hence: maximum number of calls = $2 \lg m$
 - $O(\log n)$

© 2005. 324-1-45

Solving Recurrences

- Recurrence
 - a function defined by referring to itself
 - example:
 - $\text{fibonacci}(n) = 0$ if $n=0$
 - $= 1$ if $n=1$
 - $= \text{fibonacci}(n-1) + \text{fibonacci}(n-2)$ if $n > 1$
- Equivalent to recursive functions
- How to calculate the order?
 - need to convert to non-recursive equation!

© 2005. 324-1-46

Characteristic Equation

- For solving homogenous recurrences
 - of the form:

$$a_0f(n) + a_1f(n-1) + \dots + a_kf(n-k) = 0$$
 - example: fibonacci(n) = fibonacci(n-1) + fibonacci(n-2)

$$\text{fibonacci}(n) - \text{fibonacci}(n-1) - \text{fibonacci}(n-2) = 0$$
 - homogenous!
- Homogenous recurrence has
 - infinitely many solutions
 - unless: there are k known values
- For Fibonacci:
 - we need 2 known values
 - got 'em! (0 and 1)

© 2005. 324-1-47

Characteristic Equation

- Characteristic equation of a homogenous recurrence:
 - $a_0x^k + a_1x^{k-1} + \dots + a_k = 0$
 - r_i are the k solutions of this equation
- Theorem of the Characteristic Equation
 - All the solutions of the homogeneous recurrence are:

$$f(n) = \sum_{i=1}^k c_i r_i^n$$
- Fibonacci
 - homogenous rec.: $f(n) - f(n-1) - f(n-2) = 0$
 - char. equation: $x^2 - x - 1 = 0$
 - roots: $(1+\sqrt{5})/2$ and $(1-\sqrt{5})/2$
 - solution: $f(n) = c_1 (1+\sqrt{5})/2^n + c_2 ((1-\sqrt{5})/2)^n$
 - solve linear equations with two known values
 - result: $f(n) = (1/\sqrt{5})(((1+\sqrt{5})/2)^n - ((1-\sqrt{5})/2)^n)$

© 2005. 324-1-48