Mathematical Software Programming (02635)

Lecture 11 — November 22, 2018

Instructor: Martin S. Andersen

Fall 2018

Announcements

Course evaluation

Survey open from November 20 until November 30 (both days included).

Your feedback is valued

- What activities/exercises helped you learn the material?
- Which concepts/exercises/lectures/assignments/... were difficult? What can be improved?
- Is there anything that you expected to learn in this course but did not?
- Do you feel that your programming skills have improved throughout the course?
- Did the CodeJudge exercises help you learn?

This week

Topic

- Recursion

Learning objectives

- Compare iterative and recursive solutions for simple problems
- Analyze the runtime behavior and the time and space complexity of simple programs

Recursive functions

Definition A recursive function is a function that calls itself during its execution

Example 1: Factorial (single recursion)

Base case: $f_0 = 1$
Recursive case:

$$f_n = n \cdot f_{n-1}, \quad n \geq 1$$

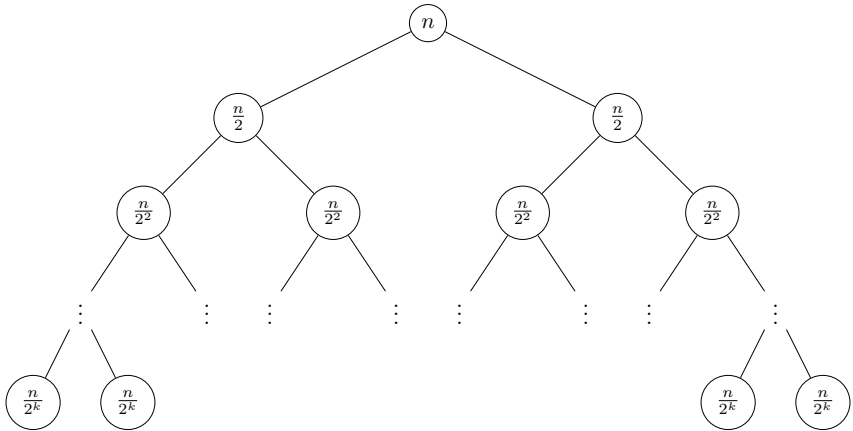
Example 2: Fibonacci numbers (multiple recursion)

Base cases: $f_0 = 0$ and $f_1 = 1$
Recursive cases:

$$f_n = f_{n-1} + f_{n-2}, \quad n \geq 2$$

Divide and conquer

Break problem into subproblems and combine answers



Example 1: power function

The function x^n (with $n > 0$ and integer) can be expressed as

$$x^n = \begin{cases} x^{n/2} \cdot x^{n/2} & n \text{ even} \\ x \cdot x^{(n-1)/2} \cdot x^{(n-1)/2} & n \text{ odd} \end{cases}$$

Example 1: power function (cont.)

Non-recursive implementation of power function x^n ($n \geq 0$ integer)

```
double power_v1(double x, unsigned int n) {
   double val = 1.0;
   for (int i=0; i<n; i++) val *= x;
   return val;
}
```

What is the space/time complexity?

Example 1: power function (cont.)

Recursive implementation of power function x^n ($n \geq 0$ integer)

```
double power_v2(double x, unsigned int n) {
    double val;
    if (n == 0)
       return 1.0;
    val = power_v2(x, n/2);
    if (n%2 == 0)  // n is even
       return val*val;
    else           // n is odd
       return x*val*val;
}
```

What is the space/time complexity?

Example 1: power function (cont.)

Non-recursive implementation of power function x^n ($n \geq 0$ integer)

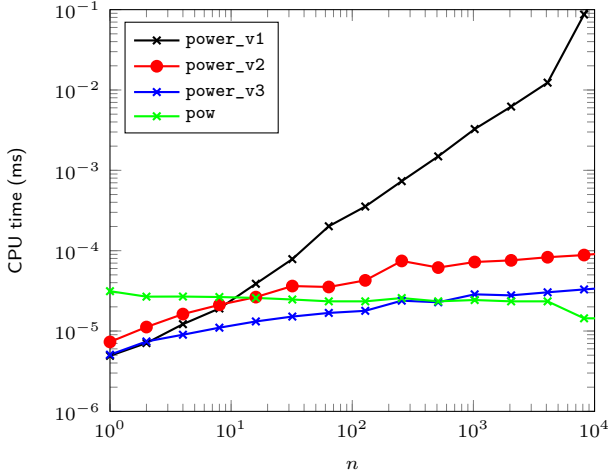
```
double power_v3(double x, unsigned int n) {
   double val = 1.0;
   while(n != 0){
      if(n%2 == 0) {    // n is even
         x = x*x;
         n = n/2;
      }
      else {            // n is odd
         val = val*x;
         n = n-1;
      }
   }
   return val;
}
```

What is the space/time complexity?

Complexity

Function	Space complexity	Time complexity
`power_v1`	$O(1)$	$O(n)$
`power_v2`	$O(\log n)$	$O(\log n)$
`power_v3`	$O(1)$	$O(\log n)$

Experiment


power_v1
power_v2
power_v3
pow
CPU time (ms)
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{0}
10^{1}
10^{2}
10^{3}
10^{4}
n

Example 2: summation

Recursively divide summation into two partial sums

$$\sum_{i=1}^{n} x_i = \sum_{i=1}^{\lfloor n/2 \rfloor} x_i + \sum_{i=\lfloor n/2 \rfloor + 1}^{n} x_i$$

```
#define Nbase 128
double recursive_sum(int n, const double * x) {
    double psum = 0.0;
    if (n > Nbase)
        return recursive_sum(n/2,x)+recursive_sum(n-(n/2),x+n/2);
    for (int k=0;k<n;k++) psum += x[k];
    return psum;
}
```

- also known as *pairwise summation* or *cascade summation*
- can be parallelized
- worst-case error bound is *better* than for sequential summation ($O(\log n)$ vs. $O(n)$)

Quiz 3

1. Go to socrative.com on your laptop or mobile device
2. Enter "room number" **02635**
3. Answer ten quick question (the quiz is anonymous)