Previously ...

Case Study: Union-Find
- quick-find algorithm
- quick-union algorithm
- weighted quick union
- path compression
- applications

Today ...

Analysis of Algorithms
- running times (empirical analysis)
- examples

Algorithm

“Any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output.”

[ Cormen et al., Introduction to Algorithms, 3rd Ed.]

Analysis of Algorithms

Analyze the amount of resources necessary to execute an algorithm
- time complexity (running time)
- space complexity (memory)

Resources typically depend on input size

Why Analysis of Algorithms?

Classify algorithms/problems
Predict performance/resources
Provide guarantees
Understand underlying principles

Timing Algorithms

\[
F_0 = 0 \\
F_1 = 1 \\
F_n = F_{n-1} + F_{n-2}
\]

0 1 1 2 3 5 8 13 21 34 . . .
typedef unsigned long int ul_int;

new type name

ul_int fib_rec(ul_int n) {
    if (n == 0 || n == 1) {
        return n;
    }
    return fib_rec(n-2) + fib_rec(n-1);
}

Recursive

ul_int fib_iter(ul_int n) {
    ul_int f;
    ul_int fib[] = {0, 1};
    if (n == 0 || n == 1) {
        return n;
    }
    for (int i = 2; i <= n; i++) {
        f = fib[0] + fib[1];
        fib[0] = fib[1];
        fib[1] = f;
    }
    return f;
}

Iterative

void time_func(ul_int (*f_ptr)(ul_int), ul_int n, char *name) {
    clock_t tic = clock();
    ul_int tt = (*f_ptr)(n);
    clock_t toc = clock();
    double elapsed = (double)(toc-tic) / CLOCKS_PER_SEC;
    printf("%s:	Output value: %ld	%f seconds\n", name, tt, elapsed);
}

int main(int argc, char **argv) {
    // get argument from command line
    ul_int n = (ul_int) atoi(argv[1]);
    // measure and print time for each call
    time_func(&fib_iter, n, "Iter");
    time_func(&fib_rec, n, "Rec");
    return 0;
}
Mathematical Model

High-level analysis — no need to implement running time $T(n)$ is a function of the input size $n$
Independent of HW/SW
Based on counts of elementary operations additions, multiplications, comparisons, etc.
exact definition not important
must be 'relevant' to the problem

What to Analyze?

We focus primarily on the worst-case