Analysis of Algorithms

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.]

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

Recursive

```c
uint64_t fib_rec(uint16_t n) {
    if (n < 2) return n;
    return fib_rec(n-1) + fib_rec(n-2);
}
```

or just ...

```c
uint64_t fib_rec(uint16_t n) {
    return (n < 2) ? n : fib_rec(n-1) + fib_rec(n-2);
}
```

Iterative

```c
uint64_t fib_iter(uint16_t n) {
    uint64_t sum;
    uint64_t prev[] = {0, 1};
    if (n < 2) {
        return n;
    }
    for (uint16_t i = 2 ; i <= n ; i++ ) {
        sum = prev[0] + prev[1];
        prev[0] = prev[1];
        prev[1] = sum;
    }
    return sum;
}
```
Timing …

```c
void time_func(uint16_t n, const char *name) {
    uint64_t val;
    Clock::time_point tic, toc;
    if (! strcmp(name, "Iter")) {
        tic = Clock::now();
        val = fib_iter(n);
        toc = Clock::now();
    }
    if (! strcmp(name, "Rec")) {
        tic = Clock::now();
        val = fib_rec(n);
        toc = Clock::now();
    }
    std::cout << name << " fib(" << n << ")": " << std::fixed << std::setprecision(4) << Seconds(toc-tic).count() << " sec. Output: " << val << std::endl;
}
int main(int argc, char **argv) {
    uint16_t n = (uint16_t) atoi(argv[1]);
    time_func(n, argv[2]);
}
```

Time Complexity
Empirical Analysis vs Mathematical Model

Analyzing running time

- **Empirical Analysis**
  - Run algorithm
  - Measure actual time

- **Mathematical Model**
  - Analyze algorithm
  - Develop Model

Empirical Analysis

- Implement algorithm
- Run on different input sizes
- Record actual running times
- Calculate hypothesis
- Predict and validate
Limitations of Empirical Analysis

- Requires implementing the algorithm
  - may be difficult
  - implementation details also play a role
- Variations in HW, SW, and OS affect analysis

Analyzing running time

Empirical Analysis
- Run algorithm
- Measure actual time

Mathematical Model
- Analyze algorithm
- Develop Model
Mathematical Model

- High-level analysis — no need to implement
- Independent of HW/SW
- Based on counts of **elementary** operations
  - additions, multiplications, comparisons, etc
  - exact definition not important but **must be relevant** to the problem

Recursion Tree

- Visualize call structure, for example: fib_rec(5)

Problem?

- Same computation multiple times
- Let \( c_n \) be the number of calls to fib_rec()

\[
\begin{align*}
  c_0 &= 1 \\
  c_1 &= 1 \\
  c_2 &= c_1 + c_0 + 1 = 3 \\
  c_3 &= c_2 + c_1 + 1 = 5 \\
  c_4 &= c_3 + c_2 + 1 = 9 \\
  c_5 &= c_4 + c_3 + 1 = 15 \\
  c_6 &= c_5 + c_4 + 1 = 25 \\
  c_7 &= c_6 + c_5 + 1 = 41 \\
\end{align*}
\]

\( c_n \) at least doubles *every other time*, i.e., \( c_n > 2^{n/2} \)

Math Review

- Summations
- Powers
- Logarithms
- Proof Techniques
- Basic Probability and Combinatorics