Unit #2: Complexity Theory and Asymptotic Analysis

CPSC 221: Algorithms and Data Structures

Lars Kotthoff[1]
larsko@cs.ubc.ca

[1]With material from Will Evans, Steve Wolfman, Alan Hu, Ed Knorr, and Kim Voll.

Runtime example #6: Fibonacci

Recursive Fibonacci:

```
int fib(n)
  if(n == 0 or n == 1) return n
  return fib(n-1) + fib(n-2)
```

Recurrence relation: (lower bound)

$$
\begin{aligned}
T(0) &\geq b \\
T(1) &\geq b \\
T(n) &\geq T(n-1) + T(n-2) + c \qquad \text{if } n > 1
\end{aligned}
$$

Runtime example #6: Fibonacci

Recursive Fibonacci:

```
int fib(n)
  if(n == 0 or n == 1) return n
  return fib(n-1) + fib(n-2)
```

Recurrence relation: (lower bound)

$$
\begin{aligned}
T(0) &\geq b \\
T(1) &\geq b \\
T(n) &\geq T(n-1) + T(n-2) + c \qquad \text{if } n > 1
\end{aligned}
$$

Claim:

$$T(n) \geq b\varphi^{n-1}$$

where $\varphi = (1 + \sqrt{5})/2$. Remember the formula for computing Fibonacci numbers?
Note: $\varphi^2 = \varphi + 1$.

Runtime example #6: Fibonacci

Claim:

$$T(n) \geq b\varphi^{n-1}$$

Proof: (by induction on n)
Base case: $T(0) \geq b > b\varphi^{-1}$ and $T(1) \geq b = b\varphi^0$.
Inductive hypothesis: Assume $T(n) \geq b\varphi^{n-1}$ for all $n \leq k$.
Inductive step: Show true for $n = k + 1$.

$$\begin{aligned} T(n) &\geq T(n-1) + T(n-2) + c \\ &\geq b\varphi^{n-2} + b\varphi^{n-3} + c && \text{(by inductive hypothesis)} \\ &= b\varphi^{n-3}(\varphi + 1) + c \\ &= b\varphi^{n-3}\varphi^2 + c \\ &\geq b\varphi^{n-1} \end{aligned}$$

$T(n) \in$
Why? Same recursive call is made numerous times.

Example #7: Learning from analysis

To avoid recursive calls

- store base case values in a table
- before calculating the value for n
  - check if the value for n is in the table
  - if so, return it
  - if not, calculate it and store it in the table

This strategy is called *memoization* and is closely related to *dynamic programming*.

Example #7: Learning from analysis

```
int fib(int n) {
  int F[n+1];

  F[0]=0; F[1]=1; F[2]=1;
  for(int i=3; i<=n; ++i) {
    F[i] = F[i-1] + F[i-2];
  }
  return F[n];
}
```

How much time does this version take?

Runtime Example #8: Longest Common Subsequence

Problem: Given two strings (A and B), find the longest sequence of characters that appears, in order, in both strings.

Example:

$A =$ `search me` $B =$ `insane method`

A longest common subsequence is "`same`" (so is "`seme`")

Applications:
DNA sequencing, revision control systems, `diff`, ...