CS 381 – Spring 2021

Week 1, Lecture 2
Part 2

What are the essentials parts of a program?

- Input/output
- Variables
- Operations on variables
- Conditionals
- Repetition in the form of loops

Understanding the asymptotic performance of nested loops is part of the analysis of algorithms.

How many times is F called?

```
while n > 1 do
    for i = 1 to n do
        F(i,n)
    n = n/4
```

How many times is F called?

$$
\begin{array}{l}
\textbf{while} \;\; n > 1 \;\textbf{do} \\
\quad\quad \textbf{for } i = 1 \textbf{ to } \mathrm{n} \textbf{ do} \\
\quad\quad\quad\quad \mathrm{F(i,n)} \\
\quad\quad n = n/4
\end{array}
$$

- How many times is the while loop executed?
- How many times is F(i,n) called for each n?

How many times is F called?

```
while  n > 1 do
    for i = 1 to n do
        F(i,n)
    n = n/4
```

executed $k = \log_4 n$ times

called n times

Assume n is a power of 4 ($n=4^k$)

Total number of times F is called is…?

How many times is F called?

```
while n > 1 do
    for i = 1 to n do
        F(i,n)
    n = n/4
```

executed $k=\log_4 n$ times

called n times

Assume n is a power of 4 ($n=4^k$)

Total number of times F is called is:

$n + n/4 + n/16 + n/64 + \ldots + 4 + 1$

Review: Geometric Series

Suppose $0 < x < 1$. Then $\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$

Proof: $\sum_{i=0}^{\infty} x^i = \frac{1-x}{1-x} \sum_{i=0}^{\infty} x^i$

$$= \frac{1}{1-x}\left(\sum_{i=0}^{\infty} x^i - \sum_{i=0}^{\infty} x^{i+1}\right)$$

$$= \frac{1}{1-x}\left(\sum_{i=0}^{\infty} x^i - \sum_{i=1}^{\infty} x^i\right) = \frac{1}{1-x}$$

For $x = \frac{1}{4}$, we have $\sum_{i=0}^{\infty} \left(\frac{1}{4}\right)^{-i} = \frac{1}{1-\frac{1}{4}} = \frac{4}{3}$

How many times is F called?

```
while n > 1 do
    for i = 1 to n do
        F(i,n)
    n = n/4
```

Total number of times F is called is:

$$n + n/4 + n/16 + n/64 + \ \ldots + 4 + 1 = n \cdot \sum_{i=0}^{log_4 n} \left(\frac{1}{4^i}\right) \ < \ \frac{4n}{3}$$

How many times is F called?

```
while n > 1 do
    for i = 1 to n do
        F(i,n)
    n = n/4
```

$O(n \log n)$	$\Theta(n \log n)$
$O(n^2)$	$\Theta(n^2)$
$O(n)$	$\Theta(n)$
$O(\log n)$	$\Theta(\log n)$

How many times is F called?

```
while n > 1 do
    for i = 1 to n do
        F(i,n)
    n = n/4
```

$O(n \log n)$	~~$\Theta(n \log n)$~~
$O(n^2)$	~~$\Theta(n^2)$~~
$O(n)$	$\Theta(n)$
~~$O(\log n)$~~	~~$\Theta(\log n)$~~

Exercises

Is $\sqrt{n}\,(\log n)^2 = O(n/\log n)$?

Is $(\log n)^2 = O(\sqrt{n}/\log n)$?

What is the relationship between $(\log n)^3$ and $n^{1/2}$?

Using definition and working with inequalities works in many situations, but not in all.

$(\log n)^2$ is typically written as $\log^2 n$
$\log n^2$ is $\log (n^2)$

Review L'Hopital's rule (if needed)

- Suppose we are trying to analyze the behavior of a function such as

$$F(x) = \frac{\ln x}{x-1}$$

- Although F is not defined when $x = 1$, we need to know how F behaves near 1.

- In particular, we would like to know the value of the limit $\lim_{x \to 1} \frac{\ln x}{x-1}$

- In computing this limit, we can't apply the usual law of limits because the limit of the denominator is 0.

Review L'Hopital's rule

- In general, if we have a limit of the form $\lim_{x \to a} \frac{f(x)}{g(x)}$

where both $f(x) \longrightarrow 0$ and $g(x) \longrightarrow 0$ as $x \longrightarrow a$, then this limit may or may not exist.

- It is called an indeterminate form of type $\frac{0}{0}$.

Review L'Hopital's rule

- Another situation in which a limit is not obvious occurs when we look for a horizontal asymptote of F and need to evaluate the limit

$$\lim_{x\to\infty}\frac{\ln x}{x-1}$$

- It isn't obvious how to evaluate this limit because both numerator and denominator become large as $x \longrightarrow \infty$.

- There is a struggle between the two.
  - If the numerator wins, the limit will be ∞.
  - If the denominator wins, the answer will be 0.
  - Alternatively, there may be some compromise— the answer may be some finite positive number.

Review L'Hopital's rule

- In general, if we have a limit of the form $\lim\limits_{x \to a} \frac{f(x)}{g(x)}$

where both $f(x) \longrightarrow \infty$ (or $-\infty$) and $g(x) \longrightarrow \infty$ (or $-\infty$), then the limit may or may not exist.

- It is called an indeterminate form of type ∞/∞.

L'Hopital's rule

- Suppose f and g are differentiable and $g'(x) \neq 0$ on an open interval I that contains a (except possibly at a).

- Suppose $\lim_{x \to a} f(x) = 0$ and $\lim_{x \to a} g(x) = 0$

or that

$$\lim_{x \to a} f(x) = \pm\infty \quad \text{and} \quad \lim_{x \to a} g(x) = \pm\infty$$

In other words, we have an indeterminate form of type $\frac{0}{0}$ or ∞/∞.

- Then,

$$\lim_{x \to a} \frac{f(x)}{g(x)} = \lim_{x \to a} \frac{f'(x)}{g'(x)}$$

if the limit on the right exists (or is ∞ or $-\infty$).

Back to our exercises

- **Is $\sqrt{n}\,(\log n)^2 = O(n/\log n)$?**
- **Is $(\log n)^2 = O(\sqrt{n}/\log n)$?**

What is the relationship between $(\log n)^3$ and $n^{1/2}$?

Take limits and use L'Hopital's rule

What the limit tells us:

- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0,$ then f(n) = O(g(n)); g(n) grows faster than f(n)

  and f(n) = Θ(g(n)) does not hold

- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c,$ for a constant c>0, then f(n) = Θ(g(n))

- $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty,$ then f(n) is of a higher order than g(n)

Finding the limits can be easier than working with the definitions!

Claim: $\sqrt{}$n (log n)^2 = O(n/log n)

$$\lim_{n\to\infty} \frac{\sqrt{n}\ \ (\log n)^2}{\frac{n}{\log n}} = \lim_{n\to\infty} \frac{((\log n)^3)'}{(\sqrt{n})'} =$$

$$\frac{3\,(\log n)^2\,(\log n)'}{\frac{1}{2\sqrt{n}}} = 6\ \log e\ \log n^2 \sqrt{n}\,\frac{1}{n} = \frac{c(\log n)^2}{\sqrt{n}} = \ldots = 0$$

Note: $\log_2$ n = ln n * $\log_2$ e and (ln n)' = 1/n

Conclusion:

- n/log n grows faster and the claim follows
- $\sqrt{n}$ grows faster than log^{3}n

Common complexity classes

$O(1)$ – constant

$O(\log n)$ – logarithmic (any base; base 2 if no base indicated)

$O(\log^k n)$ – poly log

$O(n)$ – linear

$O(n \log n)$

$O(n^2)$ – quadratic; $O(n^3)$ – cubic

$O(n^k)$ – polynomial, k is a positive constant

$O(c^n)$ – exponential, c is a constant > 1

- $O(2^n)$ is not $\Theta(3^n)$

$O(n!)$ – factorial

$O(n^n)$, $O(n^{2n})$, …

Table 2.1 The running times (rounded up) of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second. In cases where the running time exceeds 10^{25} years, we simply record the algorithm as taking a very long time.

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	4 sec
$n = 30$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	< 1 sec	18 min	10^{25} years
$n = 50$	< 1 sec	< 1 sec	< 1 sec	< 1 sec	11 min	36 years	very long
$n = 100$	< 1 sec	< 1 sec	< 1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	< 1 sec	< 1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	< 1 sec	< 1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	< 1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long