- These notes are handwritten, unedited, and incomplete.
- If you find any bugs, ambiguities or incorrect interpretation of facts, please let me know. I'll be grateful to have your comments.
- Please note that I will provide notes for a few lectures, not all. Therefore, it is your responsibility to come to the class, take notes regularly, and ask me or the TAs if you have questions.

- Our first Algorithm: Computing with Fibonacci Numbers

- Asymptotic notation: $O$, $\Omega$, $\Theta$, $o$, $\omega$, $\sim$
Our First Problem:

Problem: Suppose we put two glass prisms back-to-back. When the light ray enters the first prism, how many paths it may take after changing direction? n times due to reflection?

\[ R_0 = 1 \quad R_1 = 2 \quad R_2 = 3 \quad R_3 = 5 \]

Relation

For \( n > 2 \), the \( n \)-bounce rays have two possibilities:

1. They bounce at the middle surface, and then bounce back at the top surface and continue in \( R_{n-2} \) ways.

2. They bounce back at the bottom surface and continue in \( R_n \) ways.

\[ n = R_0 + R_1 + R_2 + \ldots + R_{n-1} \]

Initial values

\[ R_0 = 1 \quad R_1 = 2 \]

Observations:

1. \( n \) odd: ray goes through the same side it entered.
2. \( n \) even: ray \( \ldots \) opp. side it entered.

Recurrence: Induction

Initial/Boundary conditions: Bases.
The above sequence \( <F_i> \), \( i \in \mathbb{N} \) is called the Fibonacci sequence.

\[
\begin{align*}
F_0 &= 0, & F_1 &= 1, & F_n &= F_{n-1} + F_{n-2} \\
R_0 &= R_1 = R_2 = R_3 = R_4 = R_5 = \ldots
\end{align*}
\]

- **History:**
  - Pingala - 200 BC
  - Varahamihira - 700 AD
  - Gopala - 1135 AD
  - Hemachandra - 1150 AD
  - Leonardo of Pisa, aka. Fibonacci - 1202 AD in the book *Liber Abaci*

- Appears a lot in nature:
  - Pine cone
  - Pineapple scales
  - Our body parts: one nose, two eyes, 3 segments of limbs, 5 fingers on each hand.
  - Honey bee family tree

- Appears a lot in:
  - Art
  - Mathematics
  - Computer science: Fibonacci heap, Fibonacci tree
How do we compute the $n$-th Fibonacci Number?

**Algorithm 1:**

1. if $n = 0$ or $n = 1$
2. return $n$
3. return $FIB1(n-1) + FIB1(n-2)$

**Analysis:**

- **Obs.:**
  1. Lots of recomputations
  2. Looks like #n$^2$ or exponential in $n$

**Time:** $T(n) = T(n-1) + T(n-2) + C_1 \approx C_2 \cdot 1.618... \cdot n \approx 0.634n$

**Space:** What is the stack depth? $n-1$

**Note on Style:** Unless otherwise mentioned, $c, d, e,$ and their subscripted/superscripted versions are **CONSTANTS**, in our derivations.

Can we do better?
Algorithm 2: Idea: Why not memoize previously computed results?

Let $F[0...n]$ be our memo.

```
FIB2(n)  
1. if $n$ is in Memo $F$  
2. return $F[n]$  
3. else if $n=0$ or $n=1$  
4. return $n$  
5. $F[n] \leftarrow FIB2(n-1)+FIB2(n-2)$  
6. return $F[n]$
```

Iterative Version

```
FIB2ITER(n)  
1. $F[0] \leftarrow 0$  
2. $F[1] \leftarrow 1$  
3. for $i \leftarrow 2$ to $n$  
4. $F[i] \leftarrow F[i-1]+F[i-2]$  
5. return $F[n]$
```

Analysis:

**Time:** $T(n) = c_4 n + c_2$ i.e. linear time, big range from exp time.

**Space:** $S(n) = c_3 n$ i.e. linear space.

Note: the above algorithm is a dynamic programming algorithm.

Can we do better, again?

Idea: All we need to remember are $F_{n-1}$ and $F_{n-2}$ to compute $F_n$.

Algorithm 3:

```
FIB3(n)  
1. if $n=0$ or $n=1$  
2. return $n$  
3. prev, curr $\leftarrow 1$  
4. for $i \leftarrow 2$ to $n$  
5. curr $\leftarrow$ prev + curr  
6. prev $\leftarrow$ curr - prev  
7. return curr
```

Analysis:

**Time:** $T(n) = c_4 n + c_2$

**Space:** $S(n) = c_3$
Did we cheat a little bit? We were a bit sloppy in our analysis.

- Our bounds look good (are they really?) assuming that the size of integers are fixed.

- Are they fixed?
  - Remember, 1 needs about \( g(\log^{0.694} n) \approx 0.694 \log n \) steps for its representation.

- Also, our algorithm fib2 and fib3 do an addition.

- More accurate to say that fib2 and fib3 take \( cn^2 + cn + cn^2 \log n^2 \) time.

- Are we still missing a small piece?

Exercise:

1. **What is the size of the input to fib#?**

2. **Express the running time and space requirements of fib# in terms of the size of the input.**

Hint:
- Don't you encode your input as a binary string?
- What can you say about time and space w.r.t. the length of your input? (Which you encode as a binary string).
Asymptotic Notation: \[ \text{"Big O"} \]

**O-notation:** Asymptotic upper bounds

- \( f(n) = O(g(n)) \) written for \( f(n) \in O(g(n)) \) to mean:
  
  \( \exists c, n_0 > 0, \text{ s.t. } 0 \leq f(n) \leq c g(n), \forall n \geq n_0. \)

**Example 1:**

Let \( f(n) = an^2 + bn + c, \) \( a > 0 \) and \( a, b, c \) are constants.

Then \( f(n) = O(n^2), \) for \( a, b, c \geq \max(a, b, c) \), \( n \geq 2 = n_0. \)

**Example 2:**

\[
2n^2 = O(n^2) \\
2n^2 = O(n^2)
\]

**Note:**

- \( O(g(n)) \) represents a set of functions, i.e.,

\[
O(g(n)) = \{ f(n) \mid \exists c, n_0 > 0 \text{ s.t. } 0 \leq f(n) \leq c g(n), \forall n \geq n_0 \}
\]
**Ω-notation:** Asymptotic lower bound

\[ \Omega(g(n)) = \{ f(n) \mid \exists c > 0, n_0 \geq \text{s.t. } 0 \leq c g(n) \leq f(n), \forall n \geq n_0 \} \]

**Example:**

- \( 3n^3 = \Omega(n^3) \)
- \( n \log n = \Omega(n \log n) \)
- \( n^2 = \Omega(n^2) \)

**Θ-notation:** Asymptotically tight bound

\[ \Theta(g(n)) = \{ f(n) \mid \exists c_1, c_2, n_0 > 0 \text{ s.t. } c_1 g(n) \leq f(n) \leq c_2 g(n), \forall n \geq n_0 \} \]

**Example:**

- \( 4x^2 + 5n + 6 = \Theta(x^2) \)
  for \( c_1 = 4, c_2 = 5, n_0 = 5 \).
- \( 5n \log n + 6n + 7 \log n + 8 = \Theta(n \log n) \)
  for \( c_1 = 5, c_2 = 15, n_0 = 29 \).

**Note:**
1. So long as you are able to find \( c_1, c_2, n_0 \) for \( \Theta \), \( c \) and \( n_0 \) for \( \Theta \) and \( \Omega \), you are good.
2. Leading constants and low order terms don't matter. But, they do matter in practice if two algorithms have comparable complexities; e.g., 2-fold speed-up can be significant in real-world applications.
3. Not dealing with constants and low order terms, however, gives us a fairly general way of comparing algorithms.
**Theorem:** \( O(g(n)) = \Omega(g(n)) \cap \Theta(g(n)) \)

i.e. for any two functions \( f(n) \) and \( g(n) \), we have
\( f(n) = \Omega(g(n)) \iff f(n) = \Theta(g(n)) \text{ and } f(n) = \Omega(g(n)). \)

**Proof:** Exercise.

**Exercise:** Show that the running time of an algorithm is \( O(g(n)) \iff \) its worst case running time is \( O(g(n)) \) and best-case running time is \( \Omega(g(n)) \).

**\( O \)-notation:** Small-Oh

\[ O(g(n)) = \{ f(n) \mid \exists c, n_0 \geq 0 \text{ s.t. } 0 \leq f(n) < cg(n), \forall n \geq n_0 \} \]

**Examples:** \( 2^n = O(n^n) \) but \( 2^n \neq o(n^n) \).
\( n \log n = O(n \log n) \)

**\( \Omega \)-notation:** Small-Omega

\[ \Omega(g(n)) = \{ f(n) \mid \exists c, n_0 \geq 0 \text{ s.t. } 0 \leq cg(n) < f(n), \forall n \geq n_0 \} \]

**Examples:** \( 2^n = \Omega(n^n) \), but \( 2^n \neq \omega(n^n) \).

**Analogy:**

\[ O \subseteq \Omega \subseteq \Theta \subseteq o \subseteq \omega \]

\[ \leq \geq = < > \]
**Some Properties of Asymptotic Notation:**

**Limit Rule:** Let \( f \) and \( g : \mathbb{N} \to \mathbb{R}^+ \). Then,

1. \( \lim_{n \to \infty} \frac{f(n)}{g(n)} \in \mathbb{R}^+ \Rightarrow f(n) = \Theta(g(n)) \)

2. \( \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \Rightarrow f(n) = o(g(n)), \)
   or equivalently, \( f(n) = O(g(n)) \) but \( g(n) \neq \Theta(f(n)) \)

3. \( \lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty \Rightarrow f(n) = \omega(g(n)), \)
   or equivalently, \( f(n) \neq O(g(n)) \) but \( g(n) = \Theta(f(n)) \)

**Examples:**

1. Let \( f(n) = 5 \cdot n \cdot \log n \) and \( g(n) = n \cdot \log n + n \).

   Then, \( \lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{5 \cdot n \cdot \log n}{n \cdot \log n + n} = \lim_{n \to \infty} \frac{5 \cdot \log n}{1 + \frac{1}{n \cdot \log n}} = \frac{5}{1} = 5. \)

   \( \therefore f(n) = \Theta(g(n)). \)

2. Let \( f(n) = n \cdot \log n \) and \( g(n) = n^2. \)

   Then, \( \lim_{n \to \infty} \frac{f(n)}{g(n)} = \lim_{n \to \infty} \frac{n \cdot \log n}{n^2} = \lim_{n \to \infty} \frac{\log n}{n} \) (L'Hopital's rule)

   \( \therefore \lim_{n \to \infty} \frac{1}{n \cdot \log n} = 0 \)

   \( \therefore f(n) = o(g(n)). \)

3. Let \( f(n) = 2^n \) and \( g(n) = (1.67)^n \cdot \log n. \)

   \( \text{Exercise: Show that } \lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty. \) (Hint: Apply L'Hopital's rule)

   \( \therefore f(n) = \omega(g(n)). \)
Reflexivity:

- \( \Omega, \omega, \Theta, \theta, 0, \omega \) are reflexive, i.e.
  \( f(x) = \Omega(f(x));\ f(x) = \omega(f(x));\ f(x) = \Theta(f(x)) \).

Symmetry:

\[ f(x) = \Theta(g(x)) \iff g(x) = \Theta(f(x)). \]

Transitivity:

- \( \Omega, \omega, \Theta, \theta, 0, \omega \) are transitive, i.e.
  \[ f(x) = \Omega(\Theta(g(x))) \land g(x) = \Theta(\omega(h(x))) \Rightarrow f(x) = \Theta(\omega(h(x))) \]
  where \( \Omega \in \{ \Omega, \omega, \Theta, \theta, 0, \omega \} \).

Asymptotic Notation for Functions with Many Parameters:

- Straightforward to generalize for multivariate functions.

Example: Functions with \( \Omega \)-parameters.

We, for example, can define \( \Theta \) notation as follows:

\[ \Theta(g(n,m)) = \{ f(c_1,m,n) \mid f(c_1,m,n) \geq c_2 g(n,m), \exists n \geq n_0, m \geq m_0 \} \]

- Easy to generalize for \( p \)-variate functions, \( p \geq 1 \).
Asymptotic Equality (as. Equivalence):

- **Definition of Limit Revisited:**

  \[ \lim_{n \to \infty} f(n) = c \text{ if } \forall \varepsilon > 0, \exists n_0 \in \mathbb{N}, \forall n > n_0, |f(n) - c| < \varepsilon. \]

- Asymptotic equality (or asymptotic equivalence) formalizes the fact that the rate of growth of two functions is the same.

  - \( f(n) \) is asymptotically equal to \( g(n) \), denoted by \( f(n) \sim g(n) \), if \( \lim_{n \to \infty} \frac{f(n)}{g(n)} = 1 \).

- That is, using our above-defined limit notation

  \( f(n) \sim g(n) \) if \( \forall \varepsilon > 0, \exists n_0 \in \mathbb{N}, \forall n > n_0, |\frac{f(n)}{g(n)} - 1| < \varepsilon. \)

- **Observations:**

  1. \( f(n) \sim c, \text{ if } c \neq 0 \) is a constant, is equivalent to \( \lim_{n \to \infty} f(n) = c. \)

  2. \( f(n) = 0 \) for at least one \( n > n_0 \), for a suitably chosen \( n_0 \), then \( f(n) \) is not asymptotically equal to any other \( g(n) \), not even to itself.

- **Exercise:** Show that \( \sim \) is an equivalence relation.

  i.e. for \( f(n), g(n), h(n) \), show that

  1. \( f(n) \sim f(n) \), i.e. reflexivity

  2. \( f(n) \sim g(n) \Rightarrow g(n) \sim f(n) \), i.e. symmetry

  3. \( f(n) \sim g(n) \land g(n) \sim h(n) \Rightarrow f(n) \sim h(n) \), i.e. transitivity
Examples of Asymptotic Equality:

1. Stirling's formula
\[ n! \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n} \]

Exercise: Show that \( \log n! \sim n \log n \).

2. The Prime Number Theorem
Let \( \pi(x) \) be the number of primes \( \leq x \). Then,
\[ \pi(x) \sim \frac{x}{\log x} \]
\( \text{i.e.} \)
\[ \lim_{x \to \infty} \frac{\pi(x)}{\frac{x}{\log x}} = 1. \]

3. Example: Show that \( x^4 + 28x^2 + 47 \sim x^4 \).
   \[ \text{Proof:} \lim_{x \to \infty} \frac{x^4 + 28x^2 + 47}{x^4} = \lim_{x \to \infty} \left(1 + \frac{28}{x^2} + \frac{47}{x^4}\right) = 1 + 0 + 0 = 1. \text{ QED.} \]

4. Example: Show that \( x + \ln x \sim x \).
   \[ \text{Proof:} \lim_{x \to \infty} \frac{x + \ln x}{x} = \lim_{x \to \infty} \left(1 + \frac{\ln x}{x}\right) = 1 + 0 = 1. \text{ QED.} \]