CMPT231

Data Structures and Algorithms

Dr. Sean Ho

Psalm 127:1-2 (NIV)

Unless the Lord **builds** the house,
the builders **labor** in vain.
Unless the Lord **watches** over the city,
the guards **stand watch** in vain.

In vain you **rise early** and **stay up** late,
toiling for food to eat –
for he **grants sleep** to those he loves.

CMPT231

Data Structures and Algorithms

Schedule

#	Date	Topic	Text	HW Due
1	Sep 13	Analysis of Algorithms, Insertion Sort, Math Review, Asymptotic Notation	ch 1-3	–
2	Sep 20	Divide and Conquer, Solving Recurrences, Randomized Algorithms	ch 4-5	HW1: 9/22
3	Sep 27	Heaps/Queues, Quicksort	ch 6-7	HW2: 9/29
4	Oct 4	Linear-time Sort and Hash Tables	ch 8, 11	HW3: 10/6
5	Oct 11	Pointers: Linked Lists, Binary Search Trees	ch 10, 12	HW4: 10/13
	Oct			HW5:

What you need to succeed in 231

- **Explorer's** heart (self-motivated)
- Discrete **math** (e.g., MATH150)
  - Logic, **proofs**
- Comfortable **coding** environment
  - Python, C++, Java, etc.
  - but not until **later** in semester

Outline for today

- **Algorithmic** analysis
    - **Insertion** sort
- Discrete **math** review
    - **Logic** and proofs
    - Monotonicity, limits, iterated functions
    - **Fibonacci** sequence and golden ratio
- **Asymptotic** notation: V , O, l , o, ω
    - **Proving** asymptotic bounds

What is an algorithm?

- Precise **process** for solving a problem:
  - Input → Compute → Output
- Various languages for **expression**:
  - English, pseudocode, UML diagrams, etc.
- Programming languages for **implementation**:
  - Python, C, Java, etc.
- Focus: not **toolkits** but **problem solving**

Algorithmic complexity

- How many machine **instr** to **execute**
  - As function of input **size**
  - Ignoring **constant** factors
- Depends on machine **architecture**
  - CPUs generally **sequential**
  - GPUs are massively **parallel**
- **Running time** is more complex than this
  - Cache / **memory** very important

Basic machine model

- Our simple CPU **instruction set**:
  - **Arith**: +, -, *, /, <, >, ≠
  - **Data**: `load` (read), `store` (write), copy
  - **Control**: `if`/`else`, `for`/`while`, functions
  - **Types**: char, int, float
  - Data **Structures**: pointers, fixed-length arrays
    - but **not** Python list / STL vec!
- Assume each takes **constant** time

Problem definition: Sorting

- Input: **array** of key-value pairs
  - wlog, assume **keys** are `1` … `n`
  - **values** (payload) can be any data
- Output: array **sorted** by key
  - **in-place**: modify original array
  - **out-of-place**: return a copy
- In standard **libraries**:
  - Python: `sort()`, `sorted()`
  - C++/Java: `sort()`
  - **How** do they do it?

Outline for today

- Algorithmic analysis
  - **Insertion sort**
- Discrete math review
  - Logic and proofs
  - Monotonicity, limits, iterated functions
  - Fibonacci sequence and golden ratio
- Asymptotic notation: Θ, O, Ω, o, ω
  - Proving asymptotic bounds

Insertion sort: hand of cards

```
insertion_sort(A, n):
  for j = 2 to n:
    key = A[j]
    i = j - 1
    while i > 0 and A[i] > key:
      A[i+1] = A[i]
      i = i - 1
    A[i+1] = key
```

In:	E	B	D	F	A	C
j=3	B	E	D	F	A	C
j=4	B	D	E	F	A	C
j=5	B	D	E	F	A	C
j=6	A	B	D	E	F	C
Out:	A	B	C	D	E	F

Proof of correctness

- Loop **invariant**:
- Property that is true **before**, **during**, and **after** loop
- For insertion sort: at each iteration of loop,
  - part of array `A[1 .. j-1]` is in **sorted** order

```
insertion_sort(A, n):
  for j = 2 to n:
    key = A[j]
    i = j - 1
    while i > 0 and A[i] > key:
      A[i+1] = A[i]
      i = i - 1
    A[i+1] = key
```

Complexity analysis

Let t_j be the number of times the loop condition is checked in the inner “while” loop:

```
insertion_sort(A, n):
  for j = 2 to n: # c0 * n
    key = A[j] # c1 * (n-1)
    i = j - 1 # c2 * (n-1)
    while i > 0 and A[i] > key: # c3 * sum (t_j)
      A[i+1] = A[i] # c4 * sum (t_j - 1)
      i = i - 1 # c5 * sum (t_j - 1)
    A[i+1] = key # c6 * (n-1)
```

Summation notation: $\sum_{2}^{n} t_j = t_2 + t_3 + \ldots + t_n$

Best vs. worst case

- **Best** case is if input is **pre-sorted**:
  - Still need to **scan** to verify sorted,
  - But inner **while** loop only has 1 iteration: $t_j = 1$
  - Total complexity: T(n) = a n + b, for some a, b
  - **Linear** in n
- **Worst** case: input is in **reverse** order!
  - Inner “while” loop always max iterations: $t_j = j$
  - Calculate **total** complexity T(n):
  - Pick a line in inner loop, e.g., **line 5**: `A[i+1] = A[i]`
  - Complexity of other lines **similar**

Worst case complexity

- **Total** complexity for line 5, worst-case:

$$c_4 \sum_2^n (t_j - 1) = c_4 \sum_2^n (j - 1)$$
$$= c_4(n-1)\frac{n}{2} = \left(\frac{c_4}{2}\right)n^2 - \left(\frac{c_4}{2}\right)n$$

  - **Quadratic** in n
- **Average** case: input is random, $t_j = \frac{j}{2}$ on average
  - Still quadratic (only changes by a **constant** factor)

Theta (Θ) notation

- Insertion sort, line 5: $\left(\frac{c_4}{2}\right)n^2 - \left(\frac{c_4}{2}\right)n$
- **Constants** $c_1, c_2, \ldots$ may vary for different computers
  - As n gets **big**, constants become **irrelevant**
  - Even the n term is dominated by the n^2 term
- Complexity of insertion sort is on **order** of n^2
  - Notation: $T(n) = \Theta(n^2)$ (“theta”)
- $\Theta(1)$ means an algorithm runs in **constant time**
  - i.e., does not depend on **size** of input
- We’ll define V more **precisely** later today

Outline for today

- Algorithmic analysis
  - Insertion sort
- **Discrete math review**
  - **Logic and proofs**
  - Monotonicity, limits, iterated functions
  - Fibonacci sequence and golden ratio
- Asymptotic notation: V , O, l , o, ω
  - Proving asymptotic bounds

Logic notation

- $\neg A$ (or !A): "**not** A"
  - e.g., let A = "it is Tue": then $\neg A$ = "it is not Tue"
- A $\Rightarrow$ B: "**implies**", "if A, then B"
  - e.g., let B = "meatloaf":
  - then A $\Rightarrow$ B = "if Tue, then meatloaf"
- A $\Leftrightarrow$ B: if and only if ("**iff**"):
  - **equivalence**: (A $\Rightarrow$ B) and (B $\Rightarrow$ A)
- **John 14:15**: "If you love me, keep my commands"
- **v21**: "Whoever keeps my commands loves me"
- **v24**: "He who does not love me will not obey my teaching"

Logic notation: ∀ and ∃

- ∀: “**for all**“,
  - e.g., “∀ day: meal(day) = meatloaf”
  - “For all days, the meal on that day is meatloaf“
- ∃: “there **exists**“ (not necessarily unique)
  - e.g., “∃ day: meal(day) = meatloaf”
  - “There exists a day on which the meal is meatloaf“

Logic: contrapos and converse

- **Contrapositive** of "A $\Rightarrow$ B" is $\neg B \Rightarrow \neg A$
  - **Equivalent** to original statement
  - "If Tue, then meatloaf" $\Leftrightarrow$ "if not meatloaf, then not Tue"
- **Converse** of "A $\Rightarrow$ B" is "$\neg A \Rightarrow \neg B$"
  - **Not** equivalent to original statement!
  - "if not Tue, then not meatloaf"

Outline for today

- Algorithmic analysis
  - Insertion sort
- Discrete math review
  - Logic and proofs
  - **Monotonicity, limits, iterated functions**
  - Fibonacci sequence and golden ratio
- Asymptotic notation: V , O, l , o, ω
  - Proving asymptotic bounds

Monotonicity

- f(x) is **monotone increasing** iff: $x < y \Rightarrow f(x) \leq f(y)$
  - Also called “non-decreasing”
  - Can be **flat**
- f(x) is **strictly increasing** iff: $x < y \Rightarrow f(x) < f(y)$
  - note inequality is **strict**
- “a mod n“ is the **remainder** of a when divided by n
  - e.g., 17 mod 5 = 2 (in Python: `17 % 5`)

Limits

- Formal **definitions** of limits involve ∀ and ∃
- $\lim_{x \to a} f(x) = b$: "**limit** of f(x) as x goes to a"
  - ∀ z > 0, ∃ y > 0: |x-a| < y ⇒ |f(x)-b| < z
  - When x is "close" to a, then f(x) is "close" to b
- $\lim_{n \to \infty} f(n) = b$: "**limit** of f(n) as n goes to infinity":
  - ∀ z > 0, ∃ n0: n > n0 ⇒ |f(n)-b| < z
  - When n is "big", f(n) is "close" to b

Iterated functions (recursion)

- $f^{(i)}(x)$: function f, **applied** i times to x: f(f(f(… f(x) …)))
  - **Not** the same as $f^i(x) = (f(x))^i$
- e.g., $\log^{(2)}(1000)$ = log(log(1000)) = log(3) ≈ 0.477
  - But $\log^2(1000) = (\log(1000))^2 = 3^2$ = 9
- By convention, $f^{(0)}(x) = x$ (apply f **zero** times)

Iterated log: log*(n)

- $\log^*(n) = \min\left(i \geq 0 : \log^{(i)}(n) \leq 1\right)$
- # **times** log needs to be applied to n until the result is ≤1
- e.g., let lg = $\log_2$:
  - then $\lg^*(16) = 3$, because
  - lg(lg(lg(16))) = lg(lg(4)) = lg(2) = 1

Outline for today

- Algorithmic analysis
  - Insertion sort
- Discrete math review
  - Logic and proofs
  - Monotonicity, limits, iterated functions
  - **Fibonacci sequence and golden ratio**
- Asymptotic notation: V , O, l , o, ω
  - Proving asymptotic bounds

Fibonacci sequence

- The n-th **Fibonacci number** is $F_n = F_{n-1} + F_{n-2}$
  - Start with $F_0 = 0, F_1 = 1$:
  - F_n = 0, 1, 1, 2, 3, 5, 8, 13, 21, …
  - (Lucas numbers start with $F_0 = 2$)
- Shows up all over **nature**
  - Num of **spirals** on sunflowers, pinecones, etc.

  - Vi Hart video: Doodling in Math

The Golden ratio ϕ

- ϕ is the **solution** to the equation $x^2 = x + 1$
  - $\phi = \dfrac{1 \pm \sqrt{5}}{2}$
  - Actually, two solutions: ϕ and its **conjugate**, $\overline{\phi}$
  - ϕ ≈ 1.61803, and $\overline{\phi}$ ≈ -0.61803
- Also shows up all over **nature**
  - Dimensions of **Nautilus** seashells, spiral **galaxies**, etc.
  - **Aspect ratio** in architecture, e.g., Parthenon

Fibonacci + golden ratio

- Can **prove** (#3.2-7) that $F_n = \frac{\phi^n - (\overline{\phi})^n}{\sqrt{5}}$
- Second term is **fractional**: $\frac{|(\overline{\phi})^n|}{\sqrt{5}} < \frac{1}{2}$
- Thus: $F_n = \left\lfloor \frac{\phi^n}{\sqrt{5}} + \frac{1}{2} \right\rfloor = \mathrm{round}\left(\frac{\phi^n}{\sqrt{5}}\right)$
- i.e., Fibonacci grows **exponentially**!

Outline for today

- Algorithmic analysis
  - Insertion sort
- Discrete math review
  - Logic and proofs
  - Monotonicity, limits, iterated functions
  - Fibonacci sequence and golden ratio
- **Asymptotic notation: Θ, O, Ω, o, ω**
  - Proving asymptotic bounds

Asymptotic growth: Θ, O, Ω

- Behaviour "**in the limit**" (big n)
- Define V as **class** of functions: f(n) $\in$ V(g(n)) iff
  - $\exists$ c1, c2, n0: $\forall$ n > n0, 0 $\leq$ c1 g(n) $\leq$ f(n) $\leq$ c2 g(n)
  - f is "**sandwiched**" between two multiples of g:
    - c1 g(n) and c2 g(n)
- "**Big O**": specify only **upper** bound: f(n) $\in$ O(g(n)) iff
  - $\exists$ c2, n0: $\forall$ n > n0, 0 $\leq$ f(n) $\leq$ c2 g(n)
  - e.g., $\Theta(n^2) \subset O(n^2) \subset O(n^3)$
- "**Big Omega**": l (g(n)) specifies only the **lower** bound
  - Think of other examples?

Proving asymptotic growth

- **(p.52 #3.1-2)** ∀ a, b > 0, prove: $(n+a)^b \in \Theta(n^b)$
- From definition: we need to **find** n_0, c_1, c_2 such that $\forall n > n_0 : c_1 n^b \le (n+a)^b \le c_2 n^b$
- i.e., find constants so we can **sandwich** $(n+a)^b$ in between two multiples of n^b

Prove: (n+a)^b ∈ Θ(n^b)

- **Observe** that n+a ≥ n/2, as long as n > 2|a|
  - Also, n+a ≤ 2n, as long as n > |a|
  - Hence, n+a is **sandwiched** by n/2 and 2n (if n > 2|a|):
    - n/2 ≤ n+a ≤ 2n
- **Raise** to the b power (x^b is **monotone** if x > 1, b > 0)
  - Thus, $\left(\frac{n}{2}\right)^b \leq (n+a)^b \leq (2n)^b$ (for n > 2|a|)
- So we select $n_0 = 2|a|$, $c_1 = 2^{-b}$, $c_2 = 2^b$
  - This **proves** the Theta bound.

Asymptotic shorthand

- V (g) is a **class** of functions
  - But for convenience, some **short-hand** notation:
- When V (et al) are on the **right** side of =:
  - It means “there **exists**“ $f \in \Theta(g)$
  - e.g., $2n^2 + 3n = \Theta\left(n^2\right)$
- When V (et al) are on the **left** side of =:
  - It means “**for all**“ $f \in \Theta(g)$
  - e.g., $4n^2 + \Theta(n \log(n)) = \Theta\left(n^2\right)$
  - True for **any** function in $\Theta(n \log(n))$

Asymptotic domination: o, ω

- "**Little o**": like a strict **less than** inequality: $f \in o(g)$ iff
  - $\forall\, c > 0\ \exists\, n0: \forall\, n > n0,\ 0 \le f(n) < c\, g(n)$
  - i.e., the **limit** of $f(n)/g(n) \to 0$ as $n \to \infty$
- "**Little omega**": like a strict **greater than**: $f \in \omega(g)$ iff
  - $\forall\, c > 0\ \exists\, n0: \forall\, n > n0,\ 0 \le c\, g(n) < f(n)$
  - i.e., the **limit** of $f(n)/g(n) \to \infty$ as $n \to \infty$

Examples of o and ω

- **Little o**: $n^{1.999} \in o(n^2)$, and $\dfrac{n^2}{\log(n)} \in o(n^2)$
  - but $\dfrac{n^2}{10000} \notin o(n^2)$
- **Little omega**: $n^{2.0001} \in \omega(n^2)$ and $n^2 \log(n) = \omega(n^2)$

Useful math identities

- All logs are the **same** up to a constant factor:
  - $\log_a(n) = \left(\frac{1}{\log_b(a)}\right)\log_b(n)$
  - So we often use lg = $\log_2$ for convenience
- $\Theta(1) \subset o(\log(n)) \subset o(n) \subset o(n^p) \subset o(p^n)$
  - For any constant p > 1
- In fact, ∀ a>1, b: $\lim_{n\to\infty} \frac{n^b}{a^n} = 0$
  - Hence, $n^b \in o(a^n)$
  - *"Exponentials dominate polynomials"*

Stirling's approximation

- **Factorial**: n! = n(n-1)(n-2)…(2)(1)
  - Number of **permutations** of n distinct objects
- **Stirling's approx**: $n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1+\Theta\left(\frac{1}{n}\right)\right)$
- Hence, $\log(n!) \in \Theta(n\log(n))$

Outline for today

- Algorithmic analysis
  - Insertion sort
- Discrete math review
  - Logic and proofs
  - Monotonicity, limits, iterated functions
  - Fibonacci sequence and golden ratio
- Asymptotic notation: V , O, l , o, ω
  - **Proving asymptotic bounds**

Example asymptotic proof

- (p.62 #3-3): **Prove**: $(\log n)! \in \omega(n^3)$
- **Approach**: take log of both sides (log is monotone)
- **Left side**: use Stirling: $n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1+\Theta\left(\frac{1}{n}\right)\right)$
  - So log(n!) ∈ V (n log(n))
  - Now **substitute** log(n) for n, using monotonicity of log:
    - So log((log n)!) ∈ V ((log n) log(log n))

Prove: (log n)! = ω(n^3)

- … so: log((log n)!) ∈ V ((log n) log(log n))
- **Right side**: $\log\left(n^3\right) = 3\log n$
  - This is **close** to the left side, with 3 instead of log(log n)
  - But we only need an **ω** bound, and log(log n) ∈ ω(3)
- **Combining**: $\log((\log n)!) \in \Theta((\log n)\log(\log n))$
  - $= \omega((\log n)3) = \omega\left(\log\left(n^3\right)\right)$
- So by **monotonicity**, $(\log n)! \in \omega\left(n^3\right)$

Outline for today

- **Algorithmic** analysis
  - **Insertion** sort
- Discrete **math** review
  - **Logic** and proofs
  - Monotonicity, limits, iterated functions
  - **Fibonacci** sequence and golden ratio
- **Asymptotic** notation: V , O, l , o, ω
  - **Proving** asymptotic bounds