<u>Week 1 Tuesday Lecture Notes</u>

Prep for Class: read DPV Chapter 0. You can find the prep as well as Lecture notes on Canvas under Pages.

Pictures: at the end of these notes are pictures of key proofs and concepts covered in class.

Lecture 1: Introduction, Asymptotic run time, Big-Oh Notations:

Information about Instructor:

Prof. Xiaoli Fern (Office: Kelley 3073)

Email: xfern@eecs.oregonstate.edu

Office Hours: Tuesday/Thursday after class (3:20-4:00pm)

4 TA's for the class: Juneki, Hamed, Souti, Juan.

Grade Breakdown:

5 Quizzes – 25%

3 group (up to three people per group) implementation Assignments – 25%

1 Midterm (Thursday of 6^{th} week) – 25%

1 Final – 25%

Information about the course:

- Very large class size: 150+ people. Stay focused and ask questions if you need help.
- In Class Quizzes will be at the end of classes on Thursday. From 3:05-3:20.
- You can switch to different group members between assignments.
- Communication is key with group members.
- **Problem sets practice questions:** Will be provided as written homework. However, they **will NOT be graded**. Exam questions are based on problem sets practice questions. These practice questions will be handed out a week in advance before the quiz.
- There will be recitations with TA's the Wednesday before the quiz. (times not yet
- ***(See *Lecture 1: Page 4 for complete overview list):***

Algorithm: a term coined to Honor Al Khwarizmi

- He Introduced a general way of solving problems.

Why Study Algorithms?

- They are important for all branches of CS. For example:
- Bioinformatics: comparing two gene sequences and how similar/different they are.
- Algorithms are very satisfying to accomplish.
- Knowledge of Algorithm Design are great for job interviews.

Two Fundamental questions of algorithm:

1: Is it Correct

2: Is it Efficient

Run time of algorithm depends allot on external factors: Processor speed, language, implementation... ect.

- We will abstract away from these fine details.
- Analytical, the run time at the Asymptotic Level.

In Class Example: Largest Power of n Example:

- n is a Number
- c is a Constant

Example: c + nc + n(n+1)/2 * 2c.

Conclusion: We don’t care about constants, summation and multiplication of n. All that matters is the largest power of n. The largest power will dominate for large numbers of N and thus creates an estimate for computation time.

Insertion Sort Example: *(**See Insertion Sort Picture**)*

- Give array a with n numbers. The insertion sort builds the assorted array one element at a time. The runtime of **Insertion Sort** is **not deterministic** (every run can have different runtime speeds)

- **The best case scenario**: the array is already sorted: takes 1 run through. Big-Oh run time: cn. It only runs n times
- **The Worst Case Scenario:** The list is completely out of order. Takes n * n times through. Big-Oh time: n^2.

Insertion Sort Example: Sort this array in order from lowest to highest:

- Fastest: Best case example: [1, 2, 3, 4, 5]
- Slowest: Worst case example: [5, 4, 3, 2, 1]

Runtime Depends on input: Use Worst-Case to determine runtime.

We focus on the "**Worst-Case Scenario".**

- Why not the **"Average-case"**? It is much harder to compute, requires probabilistic analysis, out of the scope of this class.
- Why not the **"Best-case"?** Rarely encountered and so is not relevant to runtime speed for real life simulations.
- Constants and lower order terms **don't matter.**
- Constant factors: 2n vs 3n. **n** is what matters.
- In contrast **n vs n^2** is a big difference**.**
- You can simplify constant and lower order terms:

For example:

$2n^2 + 2n$ vs $3n^2$. Becomes **n^2**.

2n + 5n + 4. Becomes **n.**

Why don’t we care? Because we focus on very large n values. (*Big-Oh Notation*)

Asymptotic Growth Example*: (**See Significant n Picture**)*

Shows that on small numbers the constants have an effect, but the larger the number gets the less important the constants become. This is a main reason we care about Big-Oh Notation.

Big-Oh Notation:

Intuitive meaning: Even though f(n) starts above g(n) (** See Significant n Picture**) if the two functions cross that point n(0) . The upper bound is what is important, since g(n) grows faster than f(n).

Simplified: Even though one function may start at a higher value than another it is what happens at very high values and how functions grow that determines significance.

Proof by Substitution Example: *(**See Proof 1 Picture**)*

Claim: $T(n) = 0(n^k)$

Solution: Explains why the lower terms can be ignored; the upper bounded power is all that matters when determining Big-Oh Complexity.

Proof by contradiction example. *(**See Proof 2 Picture**)*

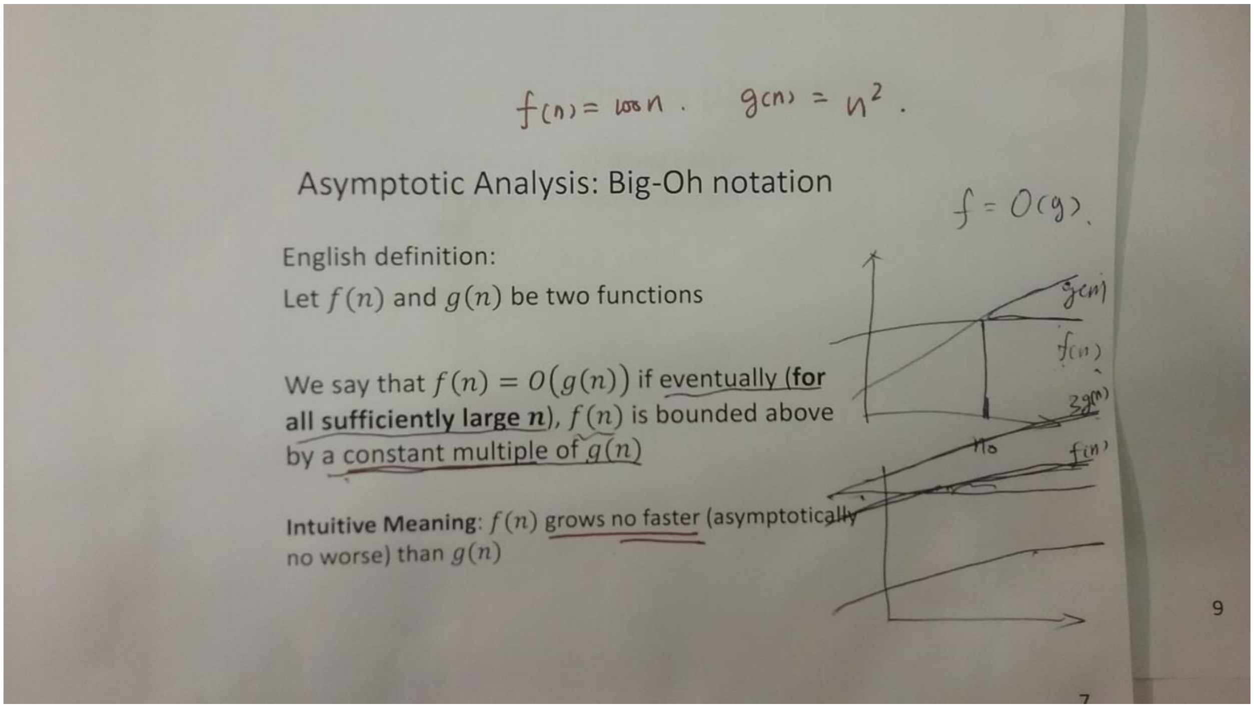
If this is true than $n^k \leq C * n^{k-1}$ for $n \geq n0$.

$n \leq c$. which is a contradiction to $n \geq n0$.

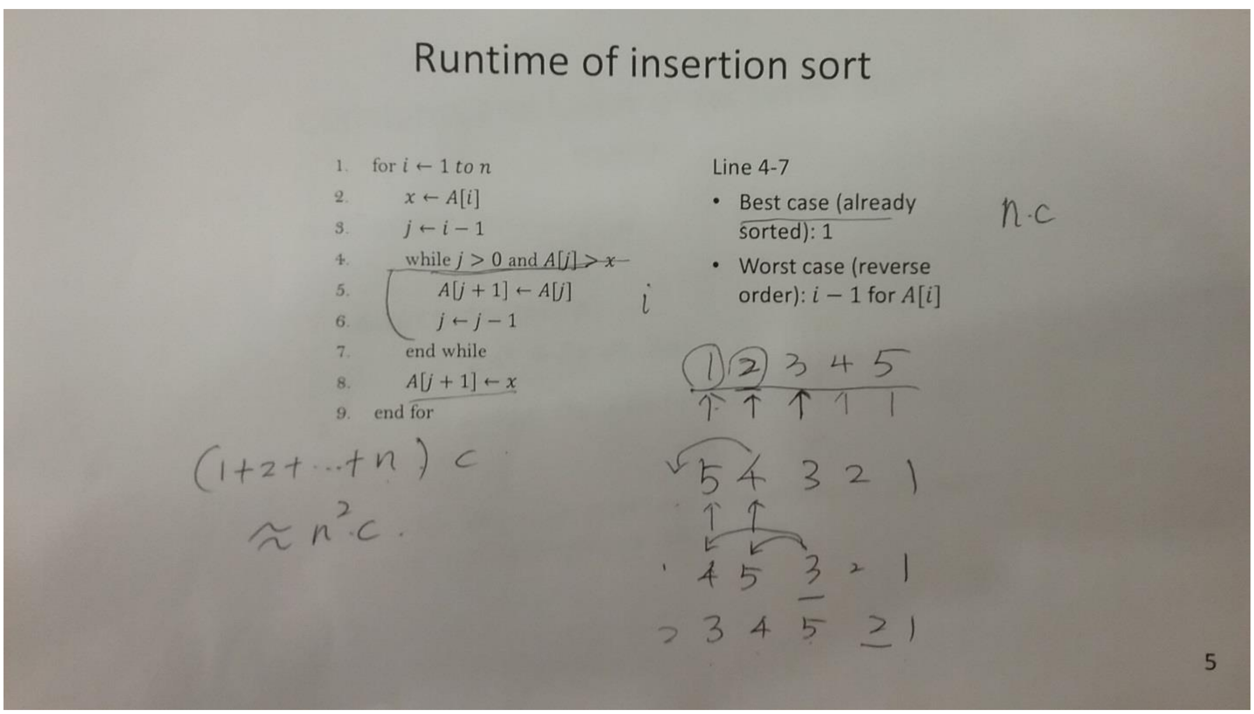
(covered pages 1-16 of Lecture 1 Powerpoint)

Pictures from Lecture 1:

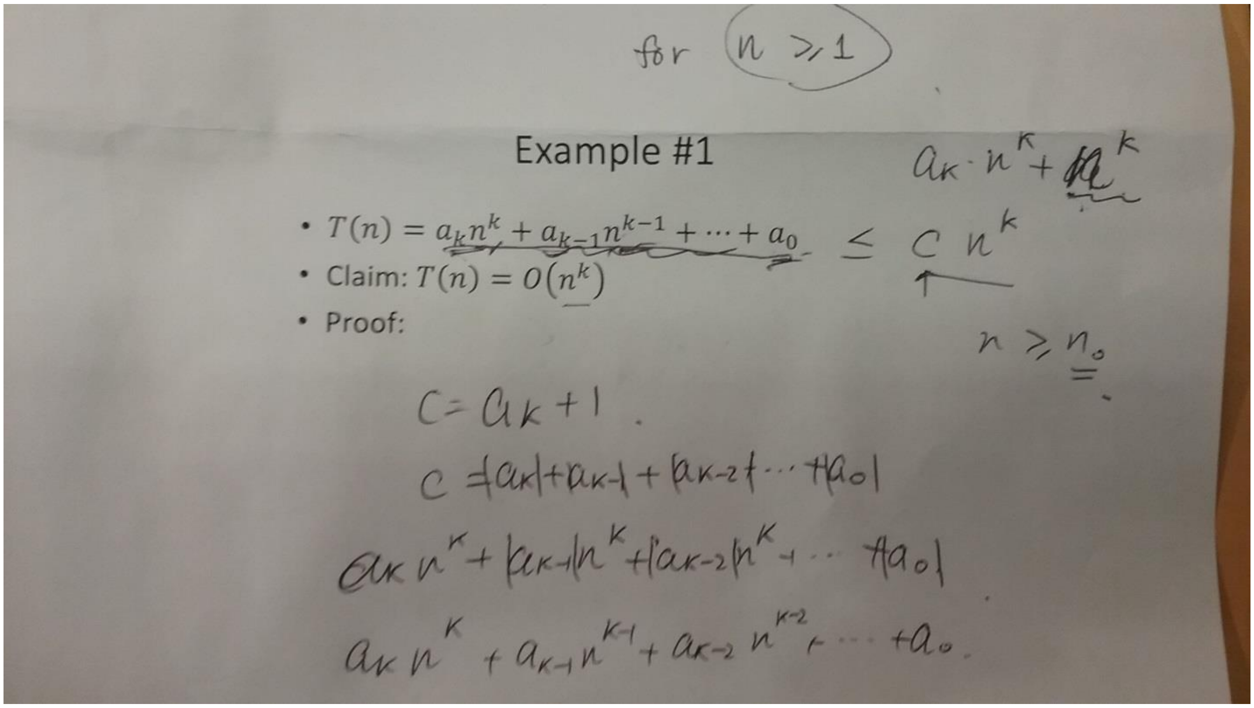
- Significant n.jpg



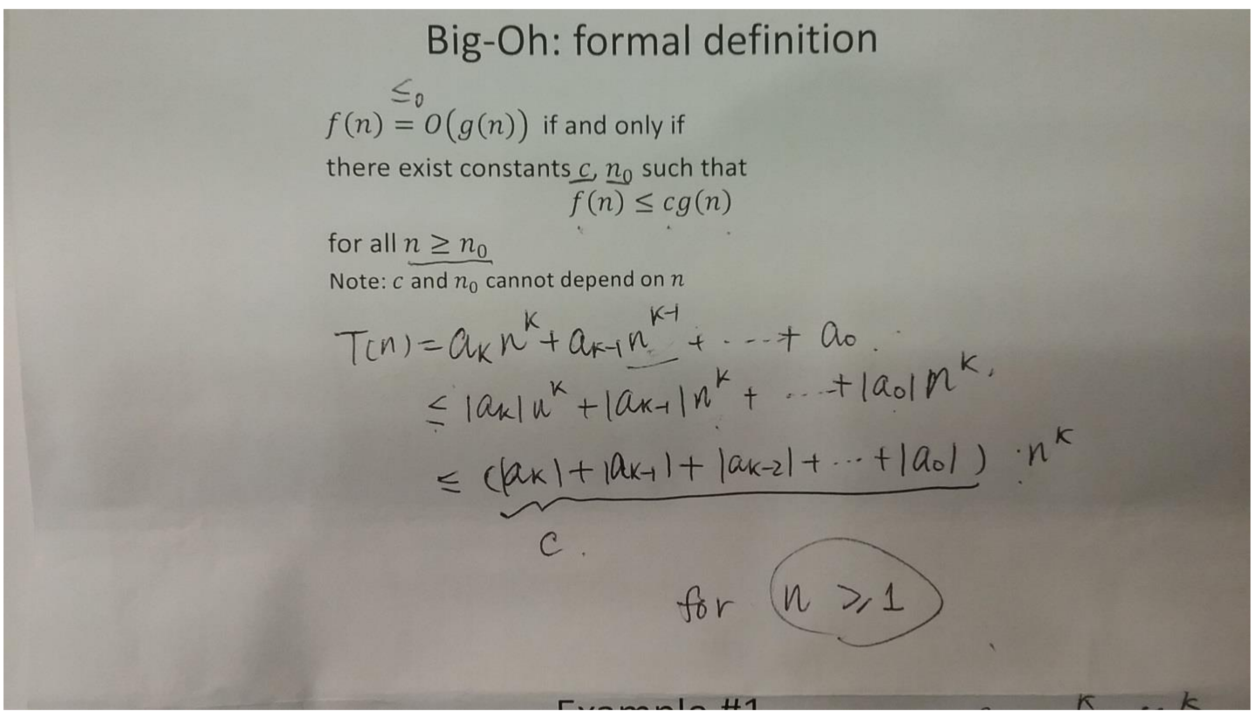
- Insertion Sort.jpg



Proof 1.jpg



Proof 2.jpg



<u>End of Week 1 Tuesday Lecture</u>

~Information composed by Notetaker Scott Russell for CS 325 **DAS** students