CSCI 310 : Advanced Algorithms

Analysis of Algorithm: Fundamentals

Organizational

- Teaching Assistant: Sophia Frankel
- Online Office Hours: Monday 12:00-1:30pm and Wednesday, 10:00-11:30am.
- Please email her if you are going to meet her during the time.
- Email: frankelsf@g.cofc.edu

Recollection from Last Lecture

An *algorithm* is a sequence of unambiguous instructions for solving a problem. i.e for obtaining a required output for any legitimate input in a finite amount of time.

Proposed Improvement: Replace legitimate input with **finite input**.

Recollection from Last Lecture

An *algorithm* is a sequence of unambiguous instructions for solving a problem. i.e for obtaining a required output for any legitimate input in a finite amount of time.
Proposed Improvement: Replace legitimate input with **finite input**.

Reasons to study Algorithms

- How to design algorithms?
- How to analyze algorithm efficiency?

Analysis of Algorithms

- Problems
  - Correctness
  - Time Efficiency
  - Space Efficiency
  - Optimality
- Approaches - Theoretical and Empirical Analysis.

Correctness

There are two main ways to verify if an algorithm solves a given problem:

- Experimental (by testing): the algorithm is executed for a several instances of the input data
- Formal (by proving): it is proved that the algorithm produces the right answer for any input data
- In practice: testing, informed by formal methods

Efficiency- Measures

- Time Complexity - Clock time, Processor cycle, Operations Count
- Space Complexity - Byte of Memory

Example : Sequential Search

- Input: a list of items and a key.
  Output: the item matching the specified key
  Think: Names and grade records
- Algorithm: Check each name in order through the entire list of names. Return either the matching record or not present
- What impacts the speed most?

Input Size

- Does measuring strictly input size tell us everything?
  - How does this map to speed?
  - How does this compare across algorithms?
- Input size impacts speed, doesn't directly determine speed.

Basic Operations

- Basic operation count determines speed
  - Exact formula e.g., $C(n) = n(n-1)/2$
- Formula indicating order of growth with specific multiplicative constant:
  - e.g., $C(n) \approx 0 : 5n^2$
- Formula indicating order of growth with unknown multiplicative constant:
  - e.g., $C(n) \approx cn^2$

Orders of growth

- The relation of basic operations to size of input determines how effcient an algorithm performs as n grows
- Basically: does increasing the input size increase the operation count a LOT or a little?

Orders of growth- $n, logn\ldots$

Basic Analysis Framework

- Effciency based on input size and basic operation count
- This framework applies to both speed and storage analysis
- One assumption we are missing

Case-wise efficiency

- Best case: what is the absolute fastest effciency for a good input size n?
- Average case: out of several executions, what is the average efficiency?
- Worst case: what is the absolute slowest efficiency for a bad input of size n?
- Amortized effciency: does repeating the algorithm many times reduce the per execution speed?

Example : Sequential Search

- Best care: 1.
- Average case: $\frac{n}{2}$?
- Worst case: n

Summary

- Time and storage are most critical
- We measure efficiency in basic operation counts with respect to the input size
- Always consider: how fast does the speed/storage requirement grow (i.e. order of growth)?
- Different qualities of inputs affect the efficiency.

Next time

- Levitin Chapter 2.2
- Homework:
  Chapter 2.1: 5,6, 7 and 9 Remember the hints in the back of the book.
- Due Date not set. After we meet next lecture, we will discuss.