csce350 — Data Structures and Algorithms
Fall 2019 — Lecture Notes: Algorithm Analysis

This document contains slides from the lecture, formatted to be suitable for printing or individual reading, and with occasional supplemental explanations added. It is intended as a supplement to, rather than a replacement for, the lectures themselves — you should not expect the notes to be self-contained or complete on their own.

1 Introduction

The basic question in **algorithm analysis** is: 2.1

How "good" is my algorithm?

This can be measured across many different dimensions.

- Time efficiency
- Space efficiency

2 Example

Problem: Array search

- **Input:** An array $A[0, \ldots, n-1]$ and a search key K.
- **Output:** An index i such that $A[i] = K$, or -1 if there is no such index.

Algorithm:

```
SEQUENTIALSEARCH(A[0, ..., n − 1], K)
  i ← 0
  while i ≤ n and A[i] ≠ K do
    i ← i + 1
  end while
  if i < n then
    return i
  else
    return −1
  end if
```

3 Step 1

Step 1: Identify the **input size**.

4 Step 2

Step 2: Identify the **basic operation**.

Why focus on basic operations?

$$T(n) \approx c_{\text{op}} C(n)$$

5 *Step 3*

Step 3: **Count** the basic operations.

Three choices:

- $C_{\text{worst}}(n)$: maximum over all inputs of size n
- $C_{\text{avg}}(n)$: average over all inputs of size n
- $C_{\text{best}}(n)$: minimum over all inputs of size n

6 *Back to* SEQUENTIALSEARCH

Identify input size: Length of array. n (...or $n+1$ if you count K.)

Identify basic operation: Comparison between two array elements

Count of basic operations.

$$C_{\text{worst}}(n) =$$

$$C_{\text{best}}(n) =$$

$$C_{\text{avg}}(n) =$$

* …if K is in A and all positions are equally likely.

7 *Step 4*

Step 4: **Classify** the order of growth.

Thought experiment: Suppose $C_{\text{worst}} = an^2$.
What happens if we double the input size?

$$T(2n) =$$

8 *So what?*

Conclusion: "Ignore" multiplicative constants. 2.2

Result: "Efficiency classes" a.k.a. "orders of growth"

Basic tool: Asymptotic notations: big-O big-Ω big-Θ