Notes for ECE 36800 - Data Structures and Algorithms

Shubham Saluja Kumar Agarwal

January 15, 2024

These are lecture notes for spring 2024 ECE 36800 at Purdue. Modify, use, and distribute as you please.

Contents

Course Introduction 1
Introduction to Data Structures & Algorithms 2
Asymptotic Notation 2

Course Introduction

Provides insight into the use of data structures. Topics include stacks, queues and lists, trees, graphs, sorting, searching, and hashing. The learning outcomes are:

- Advanced programming ideas, in practice and in theory
- Data structures and their abstractions: Stacks, lists, trees, and graphs
- Fundamentals of algorithms and their complexities: Sorting, searching, hashing, and graph algorithms
- Problem Solving

Introduction to Data Structures & Algorithms

Data Structures are methods of organizing information for ease of manipulation. Examples:

1. Dictionary
2. Check-out line or queues
3. Spring-loaded plate dispenser or stacked
4. Organizational Chart or tree

These are associated with methods known as algorithms to be manipulated
Algorithms are methods of doing something. Examples:

1. Multiplying two numbers
2. Making a sandwich
3. Getting dressed

The topics of interest within them are:

- Correctness
- Efficiency in time and space

Asymptotic Notation

The questions to be asked about an algorithm are the following:

- Is it correct?
- Is it as fast as possible?
- How many machine instructions (in terms of n) does it take?

Let us take the following algorithm to add the numbers form 1 to n:

```
total = 0;
for (i=1:n)
    total = total + i;
return total
```

The cost will be:

Cost	Frequency	Function
C_1	1	Assign initial value
C_2	n+1	For loop iterations and exit
C_3	n	Number additions
C_4	1	Return value

The total is then:

$$C_1 * 1 + C_2(n+1) + C_3(n) + C_4(1) = (C_2 + C_3)n + (C_1 + C_2) + C_4$$

However the $O(n)$ will only be n, as the constants and coefficients of these will be deprecated, as we will come to understand in more detail as this topic continues.
Let us take another example of some code that has a

$$\begin{aligned} T(n) &= n^2 + 10^7 n + 10^{10} \\ T(10^{11}) &= 10^{22} + 10^{18} + 10^{10} \\ T(2*10^{11}) &= 4*10^{22} + 2*10^{18} + 10^{10} \\ \implies \frac{T(2*10^{11})}{T(10^{11})} &\approx 4 = \left(\frac{2*10^{11}}{10^{11}}\right)^2 \end{aligned}$$

This goes to show that this algorithm has an $O(n) = n^2$, and all coefficients and lower order terms that are a part of the complexity are largely irrelevant for large n values. This is why this is called **asymptotic notation**.
Another example of a simple algorithm is

```
total = 0;
for (i=1:n):
    if (((i*i%3)==0)||((i*i%7)==0)):
        total = total+i*i;
return total;
```

Which has a cost table that looks like the following:

Cost	Frequency	Function
C_1	1	Assign initial value
C_2	n+1	For loop iterations and exit
C_3	n	Number of $i\%3$ comparisons
C_4	$n - \lfloor \frac{n}{3} \rfloor$	Number of $i\%7$ comparisons
C_5	$\lfloor \frac{n}{3} \rfloor + \lfloor \frac{n}{7} \rfloor - \lfloor \frac{n}{21} \rfloor$	Number of additions
C_6	1	Returning value

It can be noted that $O(n) = n$ for this function, despite all the other complexities in the algorithm. However, it is important to know how to calculate $T(n)$ as well.
Now, let us look at something more complicated, matrix multiplication of two lower triangular matrices.

```
for (i=1:n):
    for (j=1:i):
        C_ij = 0;
        for (k=j:i):
            C_ij = C_ij+A_ik*B_kj
return C
```

This has a cost table that looks like the following:

Cost	Frequency	Function
C_1	n+1	First loop
C_2	$\sum_{i=1}^{n}(i+1)$	Second loop
C_3	$\sum_{i=1}^{n}\sum_{j=1}^{i} 1$	Number of assigns
C_4	$\sum_{i=1}^{n}\sum_{j=1}^{i}(i-j+2)$	Third loop
C_5	$\sum_{i=1}^{n}\sum_{j=1}^{i}\sum_{k=j}^{i} 1$	Number of assigns to matrix
C_6	1	Returning value