CS 218 : Design and Analysis of Algorithms

Lecture 3: Analysis of Algorithms

Lecturer: *Sundar Vishwanathan*
Computer Science & Engineering — Indian Institute of Technology, Bombay

1 The Model of Computation

There are two parameters of interest for any algorithm. One is the time it takes to yield an output and the other is the amount of memory it uses. Our focus will be on the time taken. By analysis of an algorithm, loosely, we mean getting an upper bound on the time taken by the algorithm for various inputs.

The computing machines on which our algorithms are analyzed are assumed to have a CPU and adequate memory (RAM). We also assume that all the usual primitives like add, subtract, compare, transfer of control, indirect addressing, etc, are available. Primitive operations are assumed to have unit cost per operation. There is an alternative log cost model in which the cost of say adding or comparing two b bit numbers is $O(b)$.

In most cases, when our algorithm deals with n numbers, we will assume that each is $O(\log n)$ bits long and that the cost of each operation on such numbers is $O(1)$. However, if we run into algorithms where the numbers grow, then we will invoke the log cost model. Our focus is on making algorithms faster and not on conserving memory. In this course we will focus only on running times of algorithms. However, note that if the memory requirement is large, it is very likely that the algorithm is taking a lot of time since reading/writing to those locations will take time.

2 Comparing Algorithms

Throughout this course, by analysis, we always mean worst case analysis. What is worst case analysis? Here is a short discussion.

Consider two algorithms A_1 and A_2 for a problem Π which we wish to compare. You can think of Π as sorting. One way to compare them is to get test cases, run both the algorithms on these test cases and pick the one which performs better on average. This analysis is empirical and is often used in practice. However, this is not satisfactory, since there is no *guarantee* about their behavior on all inputs. Also it seems preferable to attach a figure of merit to each algorithm so that one can compare many algorithms together. One problem with comparing algorithms is that on different inputs different algorithms may perform better. One way to deal with this is to consider the average behavior over all inputs. This is called average case analysis. The other way, which we will follow, is to consider the maximum time taken over all inputs. This is called the *worst case analysis.* There is another problem one has to deal with. The time taken by any non-trivial algorithm increases with the input size. For each algorithm, for each input size, a figure of merit based on our discussion so far could be the maximum time taken by the algorithm over all inputs of that size.

But this entails one figure of merit for each input size which is an infinite set. Note that this figure of merit will grow with input size. This is one problem. The other problem is that calculation of this value for each value of input size is clearly impossible. Ideally however one would like one figure of merit. The solution that deals simultaneously with all these problems is to give the best upper bound one can for the running time as a function of the input size.

Let A be an algorithm. Let $A(I)$ be the time taken by the algorithm on input I. Find the best (slowest growing) function f which can be easily described which satisfies the following for all n.

$$A(I) \leq f(n) : |I| = n$$

This function f is the figure of merit we will attach to the algorithm.