Lecture 09 Notes

Derek Harter

CSci 515 Spring 2015 *<2015-03-31 Tue>*

1 First Session (11 - 11:40)

1.1 Introduction to Analysis of Algorithms

Analysis of algorithms is a branch of computer science that studies the performance of algorithms, especially their run time and space requirements. Most classes tend to emphasize the run time complexity, but space complexity can be just as important, e.g if your program can't fit into the size of primary memory for a supercomputing application, if often will go from being feasible to compute, to impossible.

- The practical goal if analysis of algorithms is to predict the perormance of different algorithms, as a function of the input size, in order to guide design decisions.
- The goal is to make meaningful comparisons between algorithms, but some problems:
  - The relative performance might depend on characteristics of hardware. Alg 1 faster machine A, Alg 2 faster machine B. Specify a machine model.
  - Relative performance might depend on details of dataset. Thus usually look at best, worst and average case behavior.
  - Relative importance depends on the size of the input. A sorting algorithm will be slower for a small list than for a very large one.
- Usual solution for last: specify run time (or number of operations) as a function of problem size. Compare functions **asymptotically** as the problem size increases.

1.2 Order of Growth

Suppose you have analyzed two algorithms and expressed their run times in terms of the size of the input: Algorithm A takes $100n + 1$ steps to solve a problem with size n; Algorithm B takes $n^2 + n + 1$ steps.

Which algorithm is "better" in terms of time complexity (number of steps)?

The following table shows run time of algorithms for different problem sizes:

Input size	Run time of Algorithm A	Run time of Algorithm B
10	1 001	111
100	10 001	10 101
1 000	100 001	1 001 001
10 000	1 000 001	$> 10^{10}$

- Big O notation
- Time and Space
- Relative growth as a function of the size N of input.

2 Second Session (11:45 - 12:30)

2.1 Sorting Arrays

- Bubble Sort
- Insertion Sort
- Merge Sort

3 Third Session (12:40 - 1:40)

Algorithm	**Best case**	**Expected**	**Worst case**
Bubble sort	$O(N^2)$	$O(N^2)$	$O(N^2)$
Insertion sort	$O(N^2)$	$O(N^2)$	$O(N^2)$
Merge Sort	$O(NlogN)$	$O(NlogN)$	$O(NlogN)$
Linear search	$O(1)$	$O(N)$	$O(N)
Binary search	$O(1)$	$O(logN)$	$O(logN)$