CS310: ALGORITHMS AND DATA STRUCTURES

Homework

- Questions?

Computational Time of Insertion Sort

- In general: $K_1(n-1) + K_2 \sum_{j=2 \ldots n} (j - 1)$

- Three notions of computational time
  - Worst case: Maximum time on any input of size n
  - Average case: Expected time over all inputs of size n
  - Best case: Minimum time on any input of size n

- Computational time equation for each case?

Computational Time Equations

- Worst Case: $K_1(n-1) + K_2\sum_{j=2 \dots n}(j - 1)$
- Average Case: $K_1(n-1) + K_2\sum_{j=2 \dots n}(j - 1)/2$
- Best Case: $K_1(n-1)$

- Not worry about the value of the constants but only about the most dominant term as $n \to \infty$

Simplifying Computational Time

- Constants: machine dependent
- We will be concerned about values of running times only for large inputs, i.e. n -> ∞
- As such large values of n, one of the terms in the expression for the running time will usually dominate over other (lower-order) terms, so we will ignore the other terms

Asymptotic Notation

- We will thus use a machine-independent notation to express the running time of an algorithm as n -> ∞

This is the **asymptotic notation**

Worst Case: $K_1(n-1) + K_2 n(n-1)/2$ $\Theta(n^2)$

Average Case: $K_1(n-1) + K_2 n(n-1)/4$ $\Theta(n^2)$

Best Case: $K_1(n-1)$ $\Theta(n)$

Why Asymptotic Notation?

- As n -> ∞, a $\Theta(n^2)$ algorithm will beat a $\Theta(n^3)$ algorithm

From now on, not worry about constructing the actual computational time equation but only look at its asymptotic notation

Selection Sort

- An iterative algorithm that sorts in place
- Find the smallest element in the collection and place it in the beginning of the collection
- At each iteration, has a sorted and an unsorted part
- After n-1 iteration, the collection is sorted