CS 457, Fall 2016

Drexel University, Department of Computer Science

Lecture 1

Today’s Lecture

- The structure of the course
- Why study algorithms?
- How to measure the efficiency of an algorithm
- Asymptotic notation

Why study algorithms?

- What is an algorithm?
  - A well-defined **computational procedure** that takes some value(s) as **input** and produces some value(s) as **output**.
  - A **sequence of computational steps** that transform the **input** into the **output**.
- Goal of an algorithm: solve a **computational problem**
  - Sorting problem:
    - Input: (32, 20, 25, 10, 18, 1, 9)
    - Output: (1, 9, 10, 18, 20, 25, 32)
  - Shortest path, string matching, travelling salesman, knapsack, max flow …
- Algorithm **design and analysis techniques**
  - divide & conquer, recursion, randomization, dynamic programming…

Why study algorithms?

- Given a computational problem, e.g., the sorting problem
  - A specific input, e.g., (32, 20, 25, 10, 18, 1, 9) is a **problem instance**
- When is an algorithm **correct**?
  - When it computes the desired output on ***every*** *problem* instance
- When is an algorithm **efficient**?
  - Time
  - Space
  - Parallelism
  - Bandwidth
  - Simple to code…

How to measure the (time) efficiency

- The actual run-time of an algorithm depends on:
  - The **specs of the machine** being used
  - The **problem instance** at hand
    - Input size, e.g., number of values in a sorting instance
    - Even for a fixed input size, the run-time may vary by a lot! (e.g., sorting problem)

- How can we **compare** two algorithms?
  - Code and run experiments
  - Analyze the run-time as a function of the **input size**
    - Worst-case analysis
    - Best-case analysis
    - Average-case analysis

Insertion Sort

```
INSERTION_SORT ($A$)

for j =2 to A.length
    key = $A[j]$
    // Insert $A[j]$ into the sorted sequence $A[1 .. j - 1]$.
    $i = j - 1$
    while $i > 0$ and $A[i] > key$
        $A[i+1] = A[i]$
        $i = i - 1$
    $A[i + 1] = key$
```

Execution:

7 3 5 8 1 2
3 7
3 **5** 7
3 5 7 **8**
1 3 5 7 8
1 **2** 3 5 7 8

Asymptotic Notation

- Worst-case running time as a function of input size n is a **function f(n)**

- How does f(n) grow **as a function of n**?
  - $f(n) = n + \log n$
  - $f(n) = n^2 + 3n$
  - $f(n) = 2^n + n$

- Comparing algorithms for sorting:
  - Insertion sort is roughly $f(n) = c_1 n^2$. We will say that f(n) is $O(n^2)$
  - Merge sort is roughly $f(n) = c_2 n \log n$. We will say that f(n) is $O(n \log n)$