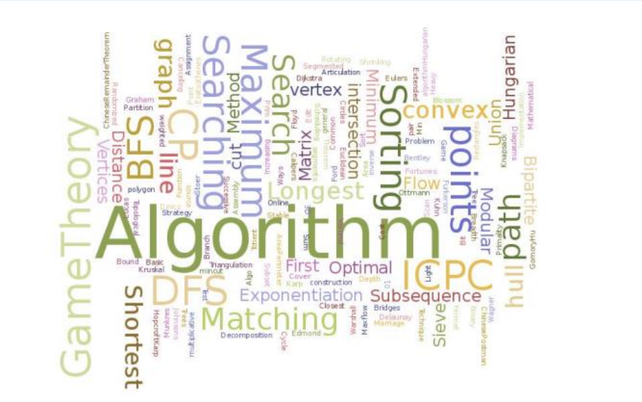
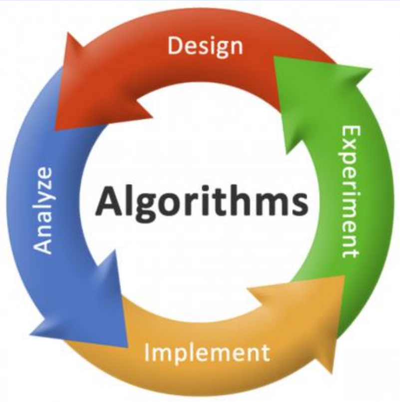
Analyzing Algorithms

COP 3503
Fall 2021
Department of Computer Science
University of Central Florida
Dr. Steinberg





Remember The Sorting Problem?
Lets look at one of the solutions.
Insertion Sort

What is the running time of Insertion-Sort in the Best Case scenario?

- Array input is sorted already. $t_j = 1$

$$T(n) = c_1 n * c_2(n-1) + 0 * (n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1)$$

$$T(n) = c_1 n * c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1) + c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1)$$

$$T(n) = (c_1 + c_2 + c_4 + c_8) * n - (c_2 + c_4 + c_8) + c_5 \sum_{j=2}^{n} t_j + (c_6 + c_7) \sum_{j=2}^{n} (t_j - 1)$$

$$\sum_{j=2}^{n} t_j = \sum_{j=2}^{n} 1 = \underbrace{1 + 1 + 1 + \cdots + 1}_{\text{(n-1) times}} = n - 1$$

$$\sum_{j=2}^{n} (t_j - 1) = \sum_{j=2}^{n} 0 = 0$$

$$T(n) = (c_1 + c_2 + c_4 + c_8)n - (c_2 + c_4 + c_8) + c_5(n-1)$$
$$T(n) = (c_1 + c_2 + c_4 + c_5 + c_8)n - (c_2 + c_4 + c_5 + c_8)$$
$$T(n) = an + b$$ Linear Function

What is the running time of Insertion-Sort in the Worst Case scenario?

- Array input is reverse order. $t_j = j$

$$\sum_{j=2}^{n} t_j = \sum_{j=2}^{n} j = 2 + 3 + 4 + \cdots + n = \frac{n(n+1)}{2} - 1 = \frac{n^2 + n - 2}{2}$$

Woah we have an Arithmetic Series: $\boxed{1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}}$

$$\sum_{j=2}^{n} t_j - 1 = \sum_{j=2}^{n} j - 1 = 1 + 2 + 3 + \cdots + (n-1) = \frac{(n-1)n}{2} = \frac{n^2 - n}{2}$$

$$T(n) = (c_1 + c_2 + c_4 + c_8) * n - (c_2 + c_4 + c_8) + c_5 \frac{n^2 + n - 2}{2} + (c_6 + c_7) \frac{n^2 - n}{2}$$

$$T(n) = \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right) n^2 + \left(c_1 + c_2 + c_4 + c_8 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2}\right) n - (c_2 + c_4 + c_8 + c_5)$$

$T(n) = an^2 + bn + c$ Quadratic Function

$$T(n) = \Theta(n^2)$$

Important! An algorithm is considered more efficient than another algorithm if its worst-case running time has a smaller order of growth.

Order of Growth

- Determining how long an algorithm runs in terms of some input.
- Drop the lower-order terms
- Ignore the constant in the leading term

Let's look at some examples of Order of Growth