Tutorial 9

The goal of this tutorial is to reinforce the following material:

- Efficiency

Efficiency

- When looking at a function, it is often useful to understand its running time.
- To do this, we compute the running time as a function of the input size, and use Big O notation to simplify the computation.

In this course, you will see the following running times:

\[ O(1) \quad O(\log n) \quad O(n) \quad O(n \log n) \quad O(n^2) \quad O(n^3) \quad O(2^n) \]

Recursive Functions

Recall the steps for a recursive function:
1. Identify the order of the function excluding any recursion
2. Determine the size of the input for the next recursive call(s)
3. Write the full recurrence relation (combine step 1 & 2)
4. Look up the closed-form solution in a table
Recurrence Relations

\[ T(n) = O(1) + T(n - k_1) = O(n) \]
\[ T(n) = O(n) + T(n - k_1) = O(n^2) \]
\[ T(n) = O(n^2) + T(n - k_1) = O(n^3) \]
\[ T(n) = O(1) + T\left(\frac{n}{k_2}\right) = O(\log n) \]
\[ T(n) = O(1) + k_2 \cdot T\left(\frac{n}{k_2}\right) = O(n) \]
\[ T(n) = O(n) + k_2 \cdot T\left(\frac{n}{k_2}\right) = O(n \log n) \]
\[ T(n) = O(1) + T(n - k_1) + T(n - k'_1) = O(2^n) \]

where \( k_1, k'_1 \geq 1 \) and \( k_2 > 1 \)

Iterative analysis

Recall the steps for an iterative function:
1. Work from the innermost loop to the outermost
2. Determine the number of iterations in the loop (in the worst case)
in relation to the size of the input \( (n) \) or an outer loop counter
3. Determine the running time per iteration
4. Write the summation(s) and simplify the expression

Common Summations

\[ \sum_{i=1}^{\log n} O(1) = O(\log n) \]
\[ \sum_{i=1}^{n} O(1) = O(n) \]
\[ \sum_{i=1}^{n} O(n) = O(n^2) \]
\[ \sum_{i=1}^{n} O(i) = O(n^2) \]
\[ \sum_{i=1}^{n} O(i^2) = O(n^3) \]