# CSC110 Lecture 25: Worst-Case Running Time Analysis

Print this handout

## Exercise 1: Worst-case running time analysis practice

Consider the following function, which has an early return:

```python
def are_disjoint_lists(nums1: list[int], nums2: list[int]) -> bool:
    """Return whether nums1 and nums2 are disjoint lists of numbers.

    Preconditions:
        - len(nums1) == len(nums2)
    """
    for x in nums1:
        if x in nums2:
            return False

    return True
```

**Note**: For your analysis in this exercise, assume both input lists have the same length $n$.

1. Find a tight upper bound (Big-O) on the worst-case running time of `are_disjoint_lists`. By "tight" we mean it should be possible to prove the same lower bound (Omega), but we're not asking you to do it until the next question.

   Use phrases like "*at most*" to indicate inequalities in your analysis.

2. Prove a matching lower bound on the worst-case running time of `are_disjoint_lists`. Remember that this means finding an input family whose asymptotic running time is the same as the upper bound you found in Question 1.

3. Using Questions 1 and 2, conclude a tight Theta bound on the worst-case running time of `are_disjoint_lists`.

## Exercise 2: Lists vs. sets

Now consider the following function, which is the same as the previous one, but operates on sets instead of lists:

```python
def are_disjoint_sets(nums1: set[int], nums2: set[int]) -> bool:
    """Return whether nums1 and nums2 are disjoint sets of numbers.

    Preconditions:
        - len(nums1) == len(nums2)
    """
    for x in nums1:
        if x in nums2:
            return False

    return True
```

*Note*: all parts of this question explores a few variations of the analysis you did in Exercise 1. To save time, don't rewrite your full analysis. Just describe the parts that would change, and the final bound that you get.

1. Analyse the worst-case running time of `are_disjoint_sets`, still assuming that the two input sets have the same length.

2. Now let's consider what happens if we remove the precondition that `nums1` and `nums2` have different lengths. For this question, let $n_1$ be the length of `nums1` and $n_2$ be the length of `nums2`.

   a. What would the worst-case running time of `are_disjoint_lists` (from Exercise 1) be in this case, in terms of $n_1$ and/or $n_2$?

   b. What would the worst-case running time of `are_disjoint_sets` be, in terms of $n_1$ and/or $n_2$?

   c. What would the worst-case running time of `are_disjoint_sets` be, in terms of $n_1$ and/or $n_2$, if we switched the `nums1` and `nums2` in the function body?

   d. Can you write an implementation of `are_disjoint_sets` whose worst-case running time is $\Theta(\min(n_1, n_2))$?