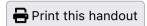
## CSC110 Lecture 25: Worst-Case Running Time Analysis



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

Consider the following function, which has an early return:

```
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:
    ix 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.



The for loop · iterates at most n times · the loop body takes at most n Steps (to evaluate x in nums 2) The return statement takes at most 1 step. .. therunning time is at most n. n+1 = n2+1 2. Prove a matching lower bound on the worst-case running time of a re\_disjoint\_lists. Rememb that this means finding an input family whose asymptotic running time is the same as the upper bound you found in Question 1. Let n EM be arbitrary and let nums! be the lat of numbers from and nums? be the list of number from [ there are many possible in puts that poone the same point?] Then the for 1000:
iterates in times
takes in steps per iteration
The neturn statement takes 1 step. 3. Using Questions 1 and 2, conclude a tight Theta bound on the worst-case runni so we add & J Z (n2) are\_disjoint\_lists. Since we know wo , Go(n2) and wc, esc(n2), we have wc, 1 e to (n2).

Exercise 2: Lists vs. sets

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

```
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 ge

1. Analyse the worst-case running time of are\_disjoint\_sets, still assuming that the two input sets have the same length.

as above but mor evaluation of long  
body is 
$$\Theta(i)$$
  
· can conclude  $WC_{a-d-s} \in O(n)$   
and  $WC_{a-d-s} \in SC(n)$   
So  $WC_{a-d-s} \in \Theta(n)$ 

- 2. Now let's consider what happens if we remove the precondition that nums 1 and nums 2 have different lengths. For this question, let  $n_1$  be the length of nums 1 and  $n_2$  be the length of nums 2.
  - a. What would the worst-case running time of are\_disjoint\_lists (from Exercise 1) be in thi 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  $\Theta(\min(n_1, n_2))$ ?

Use an if-statement where the Condition is len(nunsi) <= lon(nuns 2). [ constant time ]

> use selection to iterate one the smaller sets.