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:

```
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:        n steps
            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.

   Let $n \in \mathbb{N}$ and let nums1 and nums2 be arbitrary lists of ints of length n.

The for loop :

- iterates at most n times
- the loop body takes at most n steps (to evaluate x in nums2)

The return statement takes at most 1 step.

$\therefore$ the running time is at most $n \cdot n + 1 = n^2 + 1$

so $WC_{are\text{-}disjoint\_lists} \in O(n^2)$

2. Prove a matching lower bound on the worst-case running time of `are_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 $n \in \mathbb{N}$ be arbitrary and let nums1 be the list of numbers from $-n$ to -1

and nums2 be the list of numbers from 1 to n.

[there are many possible inputs that prove the same point]

Then the for loop :

- iterates n times
- takes n steps per iteration

The return statement takes 1 step.

The running time is (at least) $n^2 + 1$ steps.

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

so $WC_{a\text{-}d\text{-}l} \in \Omega(n^2)$

Since we know $WC_{a\text{-}d\text{-}l} \in O(n^2)$ and $WC_{a\text{-}d\text{-}l} \in \Omega(n^2)$, we have $WC_{a\text{-}d\text{-}l} \in \Theta(n^2)$.

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 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 now evaluation of loop body is $\Theta(1)$
   - can conclude $WC_{a\text{-}d\text{-}s} \in O(n)$ and $WC_{a\text{-}d\text{-}s} \in \Omega(n)$

   so $WC_{a\text{-}d\text{-}s} \in \Theta(n)$

2. Now let's consider what happens if we remove the precondition that `nums1` and `nums2` have differen 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 thi case, in terms of n_1 and/or n_2?

   $\Theta(n_1 \cdot n_2)$

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

$\Theta(n_1)$

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?

$\Theta(n_2)$

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

Yes

Use an if-statement where the condition is len(nums1) <= len(nums2).

[constant time]

Use selection to iterate over the smaller set.