Learning Objectives

By the end of this worksheet, you will:

- Analyse the worst-case running time of an algorithm.
- Find, with proof, an input family for a given algorithm that has a specified asymptotic running time.

---

1. **Substring matching**. Here is an algorithm which takes two strings and determines whether the first string is a substring of the second.[1]

```python
def substring(s1: str, s2: str) -> bool:
    for i in range(len(s2) - len(s1)):              # Loop 1
        # Check whether s1 == s2[i..i+len(s1)-1]
        match = True
        for j in range(len(s1)):                    # Loop 2
            # If the current corresponding characters don't match, stop the inner loop.
            if s1[j] != s2[i + j]:
                match = False
                break

        # If a match has been found, stop and return True.
        if match:
            return True

    return False
```

(a) Assume that both strings are non-empty, and that the length of the second string is equal to the square of the length of the first string.[2]

Let n represent the length of `s1` (and so the length of `s2` is n^2). Find a good asymptotic upper bound on the worst-case running time of `substring` in terms of n.

Solution

For a fixed iteration of Loop 1: Loop 2 takes *at most* n iterations ($j = 0, 1, \ldots, n-1$), and each iteration takes constant time. So the total number of steps for Loop 2 is n.

Loop 1 runs for *at most* $n^2 - n$ iterations ($i = 0, 1, \ldots, n^2 - 1$), and each iteration takes *at most* $n+1$ steps (counting 1 for the constant-time operations in Loop 1's body). So then the total cost of Loop 1 is *at most* $(n^2 - n)(n + 1)$.

So then since the last statement, `return False`, takes *at most* 1 step (it may or may not execute), the total running time is *at most* $(n^2 - n)(n + 1) + 1$ steps, which is $\mathcal{O}(n^3)$.

[1] In Python, this would correspond to the `in` operation, e.g., `'oof' in 'proofs are fun'`).

[2] The algorithm certainly works even if the input string lengths don't satisfy this requirement, we add it here to simplify some of the analysis.

(b) Find, with proof, an input family whose running time matches the upper bound you found in part (a).
Hint: you can pick `s1` to be a string of length n that just repeats the same character n times.

Solution

This input family is rather tricky to describe and analyse properly. Let $n \in \mathbb{Z}^+$, and let `s1` be the string of length n that only contains the character 'a', and let `s2` be the string of length n^2 defined as:

$$s2[i] = \begin{cases} b, & \text{if } n \mid i+1 \\ a, & \text{otherwise} \end{cases}$$

For example, when $n = 4$, we have

$$s1 = aaaa \text{ and } s2 = aaabaaabaaabaaab$$

Intuitively, since $s1$ and $s2$ are so similar, the inner loop has to run for many iterations until it finds a mismatch.

We leave the analysis of the running time of `substring` on this input family as an exercise, with one hint: the outer loop will run $n^2 - n$ times in total; rather than trying to sum up over all of these iterations, break it up into n groups of n consecutive iterations. You should find that the running time of the first n iterations (from $i = 0$ to $n - 1$) is more straightforward to analyse, and each subsequent group of n iterations has the same cost.