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]

```
1  def substring(s1: str, s2: str) -> bool:
2      for i in range(len(s2) - len(s1)):              # Loop 1
3          # Check whether s1 == s2[i..i+len(s1)-1]
4          match = True
5          for j in range(len(s1)):                    # Loop 2
6              # If the current corresponding characters don't match, stop the inner loop.
7              if s1[j] != s2[i + j]:
8                  match = False
9                  break
10
11         # If a match has been found, stop and return True.
12         if match:
13             return True
14
15     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.

[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.