Algorithms and Data Structures — Graded Assignment n. 1
A. Carzaniga — April 1, 2021

Due date: *Monday, April 12, 2021 at 22:00*

Instructions

- This is an individual assignment. You must write your code and documentation on your own. *Always acknowledge any and all sources you might use.*
- Write and submit source files with the exact names specified in each exercise. Do not submit any file, folder, or archive, other than what is required.
- You may only use the following, limited subset of the Python 3 language and libraries.
  - You may only use the built-in numeric types (e.g., `int`) and sequence types (e.g., arrays).
  - With arrays or other sequence types, you may only use the following operations:
    - direct access to an element by index, as in `return A[7]` or `A[i+1] = A[i]`
    - append an element, as in `A.append(10)`
    - delete the last element, as in `A.pop()` or `del A[len(A)-1]`
    - read the length, as in `n = len(A)`
  - You may use the `range` function, typically in a for-loop, as in `for i in range(10)`
  - You may not use any library or external function other than the ones listed above.

►**Exercise 1.** In a source file `ex1.py` write a Python function `first_unique(A)` that takes an array A of values (numbers, strings, whatever) and returns the first unique value in the sequence, meaning the left-most value that does not appear anywhere else in the sequence. As a source-code comment, analyze the complexity of `first_unique(A)` by also describing a worst-case input. *(20)*

►**Exercise 2.** Consider the following algorithm ALGO-X(A, x) that takes an array A of numbers and another number x.

```
ALGO-X(A, x)
 1  B = [0]   // an array containing one value: 0
 2  for i = 1 to A.length
 3       ℓ = B.length
 4       for j = 1 to ℓ
 5            s = B[j] + A[i]
 6            if ALGO-Y(B, s)
 7                 B = B ∘ s   // append s to B
 8  for i = 1 to B.length
 9       if x ≥ B[i]
10            return TRUE
11  return FALSE
```

```
ALGO-Y(A, x)
1  for i = 1 to A.length
2       if x == A[i]
3            return FALSE
4  return TRUE
```

Answer the following questions in a PDF document called `ex2.pdf`:

Question 1: Explain what ALGO-X does. Do not simply paraphrase the code. Instead, explain the high level semantics, independent of the code. *(10)*

Question 2: Analyze the complexity of ALGO-X in the best and worst case. Justify your answer by clearly describing a best- and worst-case input of size n, as well as the behavior of the algorithm in each case. *(10)*

Question 3: Write an algorithm called BETTER-ALGO-X that does exactly the same thing as ALGO-X, but with a strictly better complexity (worst-case). Analyze the complexity of BETTER-ALGO-X. *(10)*

►**Exercise 3.** Given a sequence $A = a_1, a_2, \ldots, a_n$ of positive numbers, you must tell whether A contains two distinct but possibly overlapping sub-sequences of contiguous elements, $a_i, a_{i+1}, \ldots, a_j$ and $a_k, a_{k+1}, \ldots, a_l$, such that $a_i + a_{i+1} + \cdots + a_j = a_k + a_{k+1} + \cdots + a_l$.

Question 1: In a source file `ex3_1.py` write a Python function `equal_sum_seq(A)` that solves this problem (returning `True` or `False`) with complexity $O(n^3)$. In a code comment in the same source file, analyze the complexity of `equal_sum_seq(A)`. *(20)*

Question 2: In a source file `ex3_2.py` write a Python function `equal_sum_seq2(A)` that solves this problem with complexity $O(n^2 \log n)$. In a code comment in the same source file, analyze the complexity of `equal_sum_seq2(A)`. *(30)*