Exercise Session n. 7 (31 March 2023)

Algorithms and Data Structures

Tests are available here: **Tests and Solutions**

1. New Exercise

Consider the following algorithm that takes an array `A` of integers:

```
Algo-X(A):
  s=0
  n=0
  for i = 1 to A.length:
    s = s + A[i]
    n = n + 1
  return Algo-Y(A, s, n)

Algo-Y(A, p, q):
  m = p/q
  s = 0
  for j = 1 to A.length:
    if A[j] < m:
      s=s+1
  return s
```

Explain what `Algo-X` does and analyze the complexity of `Algo-X`.

2. Exercise 178

Consider the following algorithm that takes an array `A` of integers:

```
  Algo-X(A):
    i=1
    j = A.length + 1
    while i < j:
      if A[i] mod 2 = 0: // A[i] is even
        j = j-1
```

```
        v = A[i]
        Algo-Y(A, i, j)
        A[j] = v
      else i = i + 1
    return j

Algo-Y(A, p, q):
  while p < q:
    A[p] = A[ p + 1 ]
    p = p + 1
```

Question 1 Briefly explain what `Algo-X` does and analyze the complexity of `Algo-X`.

Question 2 Write an algorithm `Better-Algo-X` that is functionally identical to `Algo-X` but with a strictly better complexity. Also briefly analyze the complexity of `Better-Algo-X`.

---

3. Exercise 254

Question 1: Explain what `Algo-X` does. Do not simply paraphrase the code. Instead, explain the high-level semantics, independent of the code.

Question 2: Analyze the complexity of `Algo-X`. Is there a difference between the best and worst case complexity? If so, describe a best-case and a worst-case input of size `n`, as well as the behavior of the algorithm in each case.

Question 3: Write an algorithm called `Better-Algo-X` that does exactly the same thing as `Algo-X`, but with a strictly better complexity in the worst case. Analyze the complexity of `Better-Algo-X`, showing a best-case and a worst-case input. Notice that if `Algo-X` modifies the content of the input array `A`, then `Better-Algo-X` must do the same. Otherwise, if `Algo-X` does not modify `A`, then `Better-Algo-X` must not modify `A`.

Examples

```
>>> count_vertical( [1,2,1,3] )
1
>>> count_horizontal( [1,1,3,1] )
1
>>> intersect( [1,1,3,1,2,0,2,4] )
True
>>> intersect( [1,2,1,3,2,1,2,2] )
False
```