24th March 2023 Session 6

Rotate array

We will revisit the solution of `rotate_array` presented yesterday in class. We will also go over an alternative solution.

Exercise 139 (m14) - algo X

Consider the following algorithm that takes an array of numbers:
Algo-X(A)

```
01      i  =  1
02      while  i < A.length
03          if  A[i] > A[i  +  1]
04              swap  A[i]  ↔  A[i  +  1]
05          p  =  i
06          q  =  i  +  1
07          for  j  =  i  +  2  to  A.length
08              if  A[j] < A[p]
09                  p  =  j
10              else if  A[j] > A[q]
11                  q  =  j
12          swap  A[i]  ↔  A[p]
13          swap  A[i  +  1]  ↔  A[q]
14          i  =  i  +  2
```

Question 1: Explain what `Algo-X` does and analyze its complexity.
Question 2: Write an algorithm `Better-Algo-X` that is functionally equivalent to `Algo-X` but with
a strictly better time complexity.

Exercise 29 (m07) - partition odd even

Develop an efficient in-place algorithm called `Partition-Even-Odd(A)` that partitions an array `A` in even and odd numbers. The algorithm must terminate with `A` containing all
its even elements preceding all its odd elements. For example, with

```
A = ⟨7, 17, 74, 21, 7, 9, 26, 10⟩
the result might be
A = ⟨74, 10, 26, 17, 7, 21, 9, 7⟩
```

`Partition-Even-Odd` must be an in-place algorithm, which means that it may use only a constant memory space in addition to `A`. In practice, this means that you may not use another temporary array.

Exercise 156 (m15) - partition primes

Write an algorithm `Partition-Primes-Composites(A)` that takes an array `A` of `n` integers such that $1 < A[i] \leq m$ for all `i`, and partitions `A` in-place so that all primes precede all composites in `A`. Analyze the complexity of your solution as a function of `n` and `m`. Recall that an integer greater than `1` is *prime* if it is divisible by only two positive integers (itself and 1) or
otherwise it is *composite*.

Exercise 97 (m11) - modulo partition

Write an in-place partition algorithm called `Modulo-Partition(A)` that takes an array `A` of `n` numbers and changes `A` in such a way that

1. the final content of A is a permutation of the initial content of A, and
2. all the values that are equivalent to 0 mod 10 precede all the values equivalent to 1 mod 10, which precede all the values equivalent to 2 mod 10, etc.

Being an in-place algorithm, `Modulo-Partition` must not allocate more than a constant amount
of memory. For example,

```
for an input array
A = ⟨7, 62, 5, 57, 12, 39, 5, 8, 16, 48⟩
```

```
a correct result would be
A = ⟨12, 62, 5, 5, 16, 57, 7, 8, 48, 39⟩
```

Analyze the complexity of `Modulo-Partition`

Exercise 175 (m16) - three way partition

Write an algorithm `Three-Way-Partition(A, v)` that takes an array `A` of `n` numbers, and partitions `A` *in-place* in three parts, some of which might be empty, so that the left
part `A[1 . . . p − 1]` contains all the elements less than `v`, the middle part `A[p . . . q − 1]` contains all the elements equal to `v`, and the right part `A[q . . . n]` contains all the elements greater than `v`.
Three-Way-Partition must return the positions `p` and `q` and must run in time `O(n)`.

Exercise 176 (m16) - algo-X

The following algorithm `Sum(A, s)` takes an array `A` of `n` numbers and a number `s`. Describe what `Sum(A, s)` does at a high level and analyze its complexity in the best and worst cases. Justify your answer by clearly describing the best- and worst-case input, as well as the behavior of the algorithm in each case.

```
Sum(A, s)
1  return  Sum-R(A, s,  1, A.length)

Sum-R(A, s, b, e)
1   if b > e and s = = 0
2     return True
3   elseif b ≤ e and Sum-R(A, s, b + 1, e)
4     return True
5   elseif b ≤ e and Sum-R(A, s − A[b], b + 1, e)
6     return True
7   else return False
```

Exercise 136 (r13) - odd in odd, even in even

Consider the following sorting problem: you must reorder the elements of an array of numbers in-place so that odd numbers are in odd positions while even numbers are in even positions. If there are more even elements than odd ones in A (or vice-versa) then those additional elements will be grouped at the end of the array.

For example,

```
with an initial sequence
A = ⟨50, 47, 92, 78, 76, 7, 60, 36, 59, 30, 50, 43⟩

the result could be this
A = ⟨47, 50, 7, 78, 59, 76, 43, 92, 36, 60, 30, 50⟩
```

Question 1: Write an algorithm called `Alternate-Even-Odd(A)` that sorts `A` in place as explained
above. Also, analyze the complexity of `Alternate-Even-Odd` . (You might want to consider ques-
tion 2 before you start solving this problem.)
Question 2: If you have not done so already, write a variant of `Alternate-Even-Odd` that runs in
`O(n)` steps for an array `A` of `n` elements.