Exercise Session n. 4 (17 March 2023)

Algorithms and Data Structures

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

Remove Higher Than

Given a list of positive numbers `A` and a count `n`, write a function `rht` that returns a copy of the list `A` but without all elements which occur more than `n` times. The function `rht` has to preserve the original order of `A`

Examples

```
>>> rht([1,2,3,3], 1
[1,2]
>>> rht([1,2,3],0)
[]
>>> rht([1,2,3,4,4,4,4,4,5,5,5], 2)
[1,2,3]
>>> rht([1,2,3,4,4,4,4,4,5,5,5], 3)
[1, 2, 3, 5, 5, 5]
```

Sort a List

Write a function `sort` which takes a list `A` and an integer direction. The function sort has to sort the list `A` in-place, using the following orders:

- ascending, if n = 1
- descending, if n = 0

Examples

```
>>> A = [2, 1, 3, 4, 5]
>>> sort( A, 1 )
>>> print( A )
[1, 2, 3, 4, 5]
>>> sort( A, 0 )
```

```
1
>>> print( A )
[5, 4, 3, 2, 1]
```

Sort Odds and Evens

Write a function called `sortevensodds` that takes in a list of positive numbers as a parameter and sorts it in-place such that all even numbers appear before odd numbers.

Examples

```
>>> A = [1,2,3,4,5,6]
>>> sort_evens_odds( A )
>>> print( A )
[2,4,6,1,3,5]
>>> A = [1,1,1,2,2,2]
>>> sort_evens_odds( A )
>>> print( A )
[2,2,2,1,1,1]
```

Sort using fastest basic sorting algorithm

Given an almost fully sorted list, e.g. `A = [1,10,45,100,5,293]`, choose and implement one of the basic sorting algorithms seen in class that would sort it the fastest: bubble sort, selection sort or insertion sort. Assume the input list to be large.