This assignment is **due on March 25**. All submitted work must be *done individually* without consulting someone else's solutions in accordance with the University's "Academic Dishonesty and Plagiarism" policies.

As a first step go to the last page and read the section: Advice on how to do the home work.

**Problem 1.** (10 points)
We have a stack that is modified in the following way: every time we call the push operation with an integer $i$, we first pop all integers that are larger than $i$ and then push $i$ onto the stack. The pop operation is unchanged.

Argue what the worst-case running time and the amortized running time of the operations of this stack are.

**Problem 2.** (25 points)
We want to design a list-like data structure that supports the following operations on integers.

- addToFront($e$): Adds a new element $e$ at the front of the list.
- addToBack($e$): Adds a new element $e$ at the end of the list.
- removeFromFront(): Removes the first element from the list and returns it.
- removeFromBack(): Removes the last element from the list and returns it.
- setElement($i$, $e$): Sets the element of the $i$-th element of the list to $e$.

Each operation should run in $O(1)$ time. You can assume that we know that the list will never contain more than $k$ elements ($k$ is **not** a constant and should **not** show up in your running time).

Example execution:

| | |
|---|---|
| addToBack(0) | [0] |
| addToFront(-1) | [-1,0] |
| addToFront(2) | [2,-1,0] |
| setElement(1,6) | [2,6,0] |
| removeFromBack() | [2,6], returns 0 |
| removeFromFront() | [6], returns 2 |

a) Design a data structure that supports the above operations in the required time.

b) Briefly argue the correctness of your data structure and operations.

c) Analyse the running time of your operations.

**Problem 3.** (25 points)
We are given $n \geq 2$ wireless sensors modelled as points on a 1D line and every point has a distinct location modelled as a positive $x$-coordinate. These sensors are given as a *sorted* array $A$. Each wireless sensor has the same broadcast radius $r$. Two wireless sensors can send messages to each other if their locations are at most distance $r$ apart.

Your task is to design an algorithm that returns all pairs of sensors that can communicate with each other, possibly by having their messages forwarded via some of the intermediate sensors. For full marks your algorithm needs to run in $O(n+k)$ time, where $k$ is the number of pairs of sensors that we need to report.

Example:


$A : [1,4,5,8,9,10,12]$, $r = 2$: return $\{\{1,2\},\{3,4\},\{3,5\},\{3,6\},\{4,5\},\{4,6\},\{5,6\}\}$. Note that $A[3]$ and $A[6]$ communicate via $A[5]$.

a) Design an algorithm that solves the problem.

b) Briefly argue the correctness of your algorithm.

c) Analyse the running time of your algorithm.

# Advice on how to do the home work

- Assignments should be typed and submitted as pdf (no handwriting)
- When designing an algorithm or data structure, it might help you (and us) if you briefly describe your general idea, and after that you might want to develop and elaborate on details. If we don't see/understand your general idea, we cannot give you points for it.
- Be careful with giving multiple or alternative answers. If you give multiple answers, then we will give you marks only for "your worst answer", as this indicates how well you understood the question.
- Some of the questions are very easy (with the help of the lecture notes or book). You can use the material presented in the lecture or book without proving it. You do not need to write more than necessary (see comment above).
- When giving answers to questions, always prove/explain/motivate your answers.
- When giving an algorithm as an answer, the algorithm does not have to be given as (pseudo-)code.
- If you do give (pseudo-)code, then you still have to explain your code and your ideas in plain English.
- Unless otherwise stated, we always ask about worst-case analysis, worst case running times etc.
- As done in the lecture, and as it is typical for an algorithms course, we are interested in the most efficient algorithms and data structures.
- If you use further resources (books, scientific papers, the internet,...) to formulate your answers, then add references to your sources.
- If you refer to a result in a scientific paper or on the web you need to explain the results to show that you understand the results, and how it was proven.