Bioinformatics Group
Prof. Dr. Rolf Backofen
Florian Eggenhofer
Michael Uhl
Rick Gelhausen

Algorithms and Data Structures
WS 2018 / 2019

`http://www.bioinf.uni-freiburg.de/Lehre/`

Exercise sheet 7

Deadline: Tuesday, 04.12.2018 12:00 AM

Expand the class *DynamicIntArray* (template on the website) with the following new functionalities:

Exercise 1 (5 points)
Implement the method *remove* which removes the last element from the list analogous to the *append* method from the lecture. Scale down the underlying fixed-size array each time it is occupied less than one third of its size. A sequence of n append / remove operations should have an amortized linear runtime. The template makes use of a fixed-size integer array taken from Python's *numPy* package. Thus you first have to install *numPy* for Python 3 (if you haven't done so yet). You can do this e.g. directly on the command line by typing:

```
sudo apt-get install python3-numpy
```

Exercise 2 (7 points)
Expand your implementation such that different tests can be selected by using a command line argument to set which test to execute. You can e.g. use Python's *argparse* module for that. Make the program produce an output which represents the accumulated runtime for a sequence of n operations. You can plot your results e.g. with *gnuplot* (see description below).

Implement the following tests:

Test 1: A sequence of 10 million *append* operations starting with an empty array.

Test 2: A sequence of 10 million *remove* operations starting with an array containing 10 million elements.

Test 3: A sequence of 10 million operations starting with an array containing 1 million elements. The operations should start with *append* and then alternate after each reallocation between *append* and *remove*.

Test 4: Same as Test 3, but this time starting with *remove*.

Generate runtime plots for all tests and commit them inside a subdirectory *non-code* as part of your solution into the SVN directory. You can e.g. use *gnuplot* for plotting the runtime. Make your program generate a text file (e.g. "runtime.txt") or just redirect the standard output of your program into the text file on the command line:

```
python3 dynamic_int_array.py --test 1 > runtime.txt
```

The text file should have the following format:
<input size 1> <TAB> <runtime 1>
<input size 2> <TAB> <runtime 2>
...
<input size n> <TAB> <runtime n>

Based on this file you can generate a plot with the following command:
```
gnuplot -e "plot 'runtime.txt'; pause -1;"
```

A window will subsequently pop up, containing the desired plot. You can then store the plot e.g. in JPEG format. For details on the used parameters please check gnuplot's man pages by executing "`man gnuplot`" or have a look here: `http://www.gnuplot.info/docs_4.6/gnuplot.pdf`

Exercise 3 (3 points)
Find out which strategies for up- and down-scaling are used with: `java.util.ArrayList` (Java), `std::vector` (C++) and Python's list data type which you already encountered in the course.

Exercise 4 (7 points) **!BONUS EXERCISE!**
The programmer in your neighboring office has asked you for a formula to calculate the increase factor r needed for increasing the array size such that on average (amortized) an `append()` operation does not exceed the costs $k \cdot A$. We have shown in the lecture that the costs for the increase factor $r = \frac{3}{2}$ are $4 \cdot A$ (i.e. $k = 4$).

Commit
Commit your code into the SVN in a new subdirectory **uebungsblatt_07** and the PDFs with the solutions of the theoretical tasks in the sub folder *non-code*. Commit your feedback in a text file *erfahrungen.txt* and specify: The length of time needed for the exercise. Which tasks have been difficult for you and where did you have problems? How much time did you spend to solve the problems?