Algorithms and Data Structures

Summer Semester 2025

For discussion on Wednesday, May 21, 2025

1. (GTG Exercise R-5.3) Modify the experiment from

   `https://github.com/mjwestcott/Goodrich/blob/master/ch05/experiment_list_size.py`

   in order to demonstrate that Python's list class occasionally shrinks the size of its underlying array when elements are popped from a list.

2. (GTG Exercise R-5.7) Let A be an array of size $n \geq 2$ containing integers from 1 to $n-1$, inclusive, with exactly one repeated. Describe a fast algorithm for finding the integer in A that is repeated.

3. (GTG Exercise C-5.15) Consider an implementation of a dynamic array, but instead of copying the elements into an array of double the size (that is, from N to $2N$) when its capacity is reached, we copy the elements into an array with $\lceil N/4 \rceil$ additional cells, going from capacity N to capacity $N + \lceil N/4 \rceil$. Prove that performing a sequence of n append operations still runs in $O(n)$ time in this case.

4. (From GTG Exercise C-5.19) Consider a shrink strategy in which an array of capacity N is resized to capacity precisely that of twice the number of elements any time the number of elements in the array goes strictly below $N/4$. Give a formal proof that any sequence of n append or pop operations on an initially empty dynamic array takes $O(n)$ time.

5. (GTG Exercise C-5.20) Consider a variant shrink strategy in which a dynamic array of capacity N is resized to capacity precisely that of the number of elements, any time the number of elements in the array goes strictly below $N/2$. Show that there exists a sequence of n operations that requires $\Omega(n^2)$ time to execute.