Exercise 4 - Solutions

1. See code in separate file

2. Case 1: Array is sorted. Then, if integer j is repeated, the list reads

$$A[0] = 1, \ldots, A[j-1] = A[j] = j, \ldots, A[n-1] = n-1$$

Thus, we can find the repeated index by a variant of binary search in $O(\log n)$ time.

Case 2: Repeated number is known to be a neighboring pair:

Iterate through the list and check which neighboring pair matches. Takes $O(n)$ time.

Case 3: Repeated pair can occur in any position:

Keep Boolean array of length $n-1$, initially set to FALSE, to check off which integers have already been encountered. If an integer has already been checked off and is seen again, return this integer. Takes $O(n)$ time and $O(n)$ additional memory. If the indices of both members of the pair are required, use an array that stores the index of each integer that has been encountered.

3. When capacity is reached, we need to copy N elements, so we are N time units "in debt". But we can perform the next $\lceil \frac{N}{4} \rceil$ appends without new allocations, so if we "charge" 4 extra time units per append, we pay back the debt by the time we fill up the newly created capacity. Thus, each append has $O(1)$ cost.

Alternative, algebraic proof: if $\left(\frac{5}{4}\right)^{i-1} < N \leq \left(\frac{5}{4}\right)^{i}$, then we reach the required capacity in at most $i-1$ copy operations. The j-th copy operation involves at most

$$\left(\ldots\left(\left(\frac{5}{4}+1\right)\cdot\frac{5}{4}+1\right)\ldots\right)\frac{5}{4}+1 = \left(\frac{5}{4}\right)^0 + \left(\frac{5}{4}\right)^1 + \ldots + \left(\frac{5}{4}\right)^j = \frac{\left(\frac{5}{4}\right)^{j+1}-1}{\frac{5}{4}-1}$$

$$\leq 5\left(\frac{5}{4}\right)^j \quad \text{elements,}$$

For a total of

$$5\left(\left(\frac{5}{4}\right)^0 + \cdots + \left(\frac{5}{4}\right)^{i-1}\right) = 5\,\frac{\left(\frac{5}{4}\right)^i - 1}{\frac{5}{4} - 1} \leq 25\left(\frac{5}{4}\right)^{i-1} \leq 25N.$$

Note: The factor 25 is a huge over-estimate, to find a simple upper bound for the ceiling function that is applied at each step. A sharper bound appears easily possible given the earlier amortization argument.

4. It is assumed that the doubling strategie for appends is used.

When an array is resized to twice its current capacity N, it incurs a debt of N time units. The next resize with the minimal number of appends is at $2N$, so an extra charge of 1 time unit per append will pay the debt. The next resize with the minimum number of pops occurs at $\lfloor \frac{N}{2} \rfloor$, so after $\lceil \frac{N}{2} \rceil$ pops. So if each pop is charged with two time units per pop will pay off the debt. Any other sequence of appends and pops will be longer than these extremal cases, thus generate more "income".

A similar argument can be made to pay off the debt of copying when down-sizing, at the same stated charges.

5. Suppose $4 \cdot 2^j + 1 = n$.

Do $2^j + 1$ appends, so that the first half of the array is full and the second half has one element. Then take $3 \cdot 2^j$ further step following the pattern p-p-a.

The cost of copying is $\left(2^j - 1\right) \cdot 2 + \left(2^j - 2\right) \cdot 2 + \cdots + \left(2^j - 2^j\right) \cdot 2$

$$= 2\left(2^j \cdot 2^j - \frac{2^j\left(2^{j+1}+1\right)}{2}\right) = \Omega\left(2^{2j}\right) = \Omega\left(n^2\right).$$