General removal in functional arrays is slow

In this short note, we look at a question asked by one of your classmates (if she wants, I can name her). The question was whether it is possible to efficiently remove the n-th element from a functional array. Recall from the supervision exercises, that it is possible to remove the first element of the functional array in $\mathcal{O}(\log n)$ time/space and still maintain a functional array representation for the rest of the elements.
The answer is that we can only do this in $\Omega(n)$ time in the general case.[1] *This means that asymptotically it is equally fast to rebuild the entire array, with the value removed.*

1 Specific example

Consider a functional array with 15 elements and let's say that we want to remove the 7-th element.

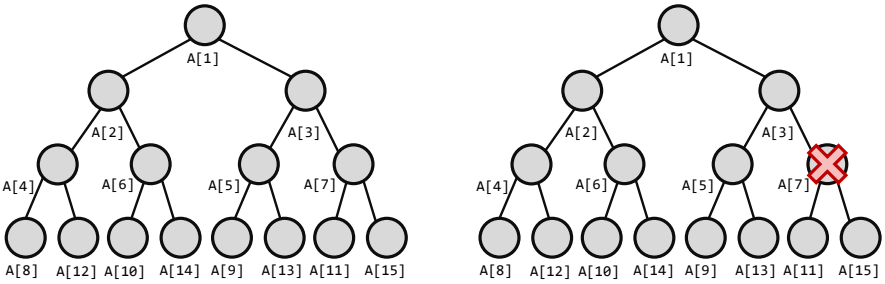

Figure 1: (**left**) The arrangement of functional array for 15 elements. (**right**) Requesting to remove element 7.

Then this means that we need to update the following entries in the functional array:

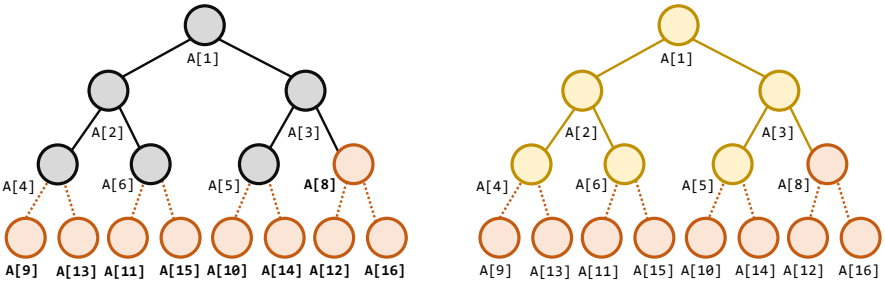

Figure 2: (**left**) The elements that directly need to change. (**right**) The elements that indirectly need to change because of the immutability of OCaml.

And this in turns means that we need to update all other nodes in the functional array by cascading updates (see Figure 2), as datatypes are immutable in functional OCaml.

2 General example

To be precise, we need to make the above example general and show that it affects $\Omega(n)$ of the nodes in the functional array for an arbitrarily large n.

[1] Of course there are some elements that can be removed efficiently (e.g. if they are very close to the beginning or end of the array). For example you can remove the second one, by removing the first two and then re-adding the first one.

Consider $n = 2^d - 1$ (above $d = 4$). We will remove the element at index $x = 2^{d-1} - 1$ (above $x = 7$). This element is the last one at depth $d - 1$. Hence, removing it means that all elements in the last level need to be moved. The last level contains $2^{d-1} = (n + 1)/2 = \Omega(n)$ elements, so we need to update the links to their parents (and as we said above these actually trigger updates to all elements in the functional array). So, in order to implement this change we need to modify $\Omega(n)$ nodes.

3 Appendix: Removing the first element

For completeness, we give the diagrams for removing the first element of the functional array and note that these are just $\Theta(\log n)$.

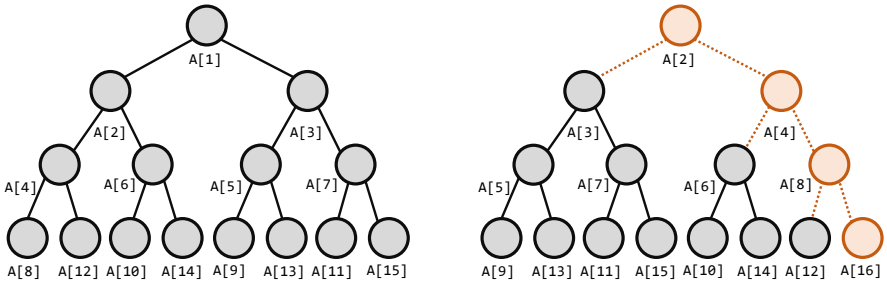

Figure 3: (**left**) The original functional array. (**right**) The functional array with the removal of the first element (the indices refer to the original functional array).

4 Further reading

There do exist data structures which allow to add/remove/access an element at an arbitrary position (and more) in $\mathcal{O}(\log n)$ time. One such data structure you will learn in the Part IA Algorithms course and it is called the *red-black tree*, which is a special type of BST. Others include: treaps, skip lists, etc.