Rento Saijo

William Tarimo

Data Structures

23 February 2025

Programming Assignment 2 Part H Answers

1. Run-time of each method in HighArray.

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Big-Omega | Big-Theta | Big-O |
| display() | Ω(n) | ϴ(n) | O(n) |
| size() | Ω(1) | ϴ(1) | O(1) |
| find() | Ω(1) | ϴ(n) | O(n) |
| insert() | Ω(1) | ϴ(1) | O(1) |
| delete() | Ω(1)\* | ϴ(n) | O(n) |
| getMax() | Ω(n) | ϴ(n) | O(n) |
| removeMax() | Ω(n) | ϴ(n) | O(n) |
| reverse() | Ω(n) | ϴ(n) | O(n) |
| allDistinct() | Ω(1) | ϴ(n2) | O(n2) |

\*while the best-case (Big-Omega) for delete() can be constant in degenerate cases like an empty array or an array with a single element, for typical non-trivial arrays, the work done is linear.

1. Run-time of each method in OrdArray.

|  |  |  |  |
| --- | --- | --- | --- |
| Method | Big-Omega | Big-Theta | Big-O |
| display() | Ω(n) | ϴ(n) | O(n) |
| size() | Ω(1) | ϴ(1) | O(1) |
| find() | Ω(1) | ϴ(log(n)) | O(log(n)) |
| insert() | Ω(log(n)) | ϴ(n) | O(n) |
| delete() | Ω(1) | ϴ(n) | O(n) |
| sort() | Ω(n) | ϴ(n2) | O(n2) |
| merge() | Ω(n + m) | ϴ(n + m) | O(n + m) |
| removeDuplicates() | Ω(n) | ϴ(n) | O(n) |

\*where m denotes the length of the second list (that’s being merged to this object).

1. Why could/would binary search not work in the HighArray class, but work in the OrdArray class?

Binary search requires the array to be sorted because it repeatedly divides the search interval in half, making assumptions about the order of elements. In the HighArray class, the elements are stored in an unsorted order, so there is no guarantee that any element to the left or right of a midpoint follows a specific order; therefore, binary search cannot reliably locate a target value in an unsorted collection. On the other hand, the OrdArray class maintains its elements in sorted order, which is a necessary precondition for binary search to function correctly.

1. If OrdArray can be searched in O(log(n)) time, while HighArray needs ϴ(n) time, does this mean it is always better to use OrdArray?

Not necessarily. While OrdArray offers faster search times due to its sorted structure, there are trade-offs. For example, maintaining a sorted order requires extra work during insertions and deletions, which can lead to O(n) operations, so if your application involves many search operations with relatively few updates, the benefits of O(log(n)) searches in an ordered array might outweigh the cost; however, if updates are frequent, the additional overhead to keep the array sorted might make an unsorted structure like HighArray more attractive. Another consideration is data size: for small arrays, the constant factors and simplicity of an unsorted array might provide better overall performances despite the higher asymptotic search cost while larger arrays may reap the benefit of a much quicker find algorithm.

1. The find() method now uses a binary search, which, as mentioned, above has a run time of ϴ(log(n)). If we have modified the delete() and insert() methods to use binary search as well, do their worst-case run-times also improve to O(log(n))?

No, they do not. Even if you use binary search to quickly locate the position for insertion or deletion, both operations still require shifting elements to maintain the order. For insertion, after determining the correct index with binary search, in the worst case scenario, you would have to shift every n elements to make room for the new element; for deletion, similarly, after locating the element to be removed, in the worst case scenario, you would have to shift every n elements to fill in the gap.