Implementation of Dynamic Tables using C++

Hrishikesh V. *Department of Computer Science*
PES University
PES1201700276
hrishi.vish@outlook.com

***Abstract*—The first assignment of Advanced Algorithms course comprises the implementation of a Dynamic Table.**

***Index Terms*—C++, dynamic table, outofbounds, pop, append**

I. Concept

The dynamic table is a way of ensuring that the cost of performing the operations of insertion and deletion on a table is O(1).

The table uses increase factor and load factor to determine by what factor to change the size of the table and when to do so.

The two primary parameters of the table are size and capacity. Size refers to the number of elements in the table and capacity refers to the space allocated to the table.

If the size of the array is a product of capacity and load factor, then the capacity is reduced to half its original value.

If the size and capacity are equal, then the capacity is doubled.

The elements are appended and removed from the rear end of the table.

When the table is to be modified, a new table of the required capacity is first created. The elements are copied to the new table, after which, the old one is deleted. The pointer of the old table is re-assigned to the new one.

II. Implementation

The table has been implemented using an array. Upon creating an object, the size, the capacity, load factor and increase factor are all set to zero. They can be changed later.

A. Insert

Insert has three possibilities. If the array is empty, then memory is dynamically allotted to the array. the capacity is set to 1 and the element is inserted.

If the size and the capacity are equal, when a new element is inserted, the capacity needs to be doubled.

To do so, another is created. The capacity of this array is determined by the increase factor, after which, every element of the first array is copied to the second. The element to be inserted is appended at the end. The first array is deleted and the pointer is assigned to the second one.

The third case is when the size is less than the capacity, in which case, the element is inserted at the rear end of the array.

B. Delete

Much like insert, delete has to be done in three different ways depending on the state of the array.

If the ratio of capacity to size is not equal to load factor, the last element is popped from the array and the size is decremented by 1.

If the array is empty, then, nothing is done. If the size becomes zero upon deleting the element, the pointer is freed.

If, on deletion, the ratio of capacity to size is equal to load factor, then a new array, whose capacity is half the original one, is created and populated with the elements of the first array. The first array is deleted and the pointer is assigned to the new one.

C. Other operations

1) get: get returns the element at the specified index. If the index is beyond the bounds of the array, the exception is manually thrown.

2) initializing load factor and increase factor: The required factors are assigned to the member variables of the class.

III. Conclusion

Dynamic arrays help rectify the limitations of static arrays. If the size is not known. Not only do they ensure a constant amortized cost, but they also efficiently manage memory.