Implementation of Dynamic Array.

R Siva Girish
Computer Science
PES University
PES1201700159
sivagirish81@gmail.com

Abstract

A Dynamic Array is an array with a variable size determined by an increase or decrease factor as the case may be.Each time the array is filled we expand the array capacity by the increase factor.Whenever the array size drops as low as a threshold of capacity times decrease factor.The array is shrunk in size to capacity times the decrease factor and increase factor inorder to maintain the potential required for future expansions and contractions of the array.

Keywords

Dynamic Array; Amortized Analysis; increase factor; load reduction factor; Cost

I. Problem Statement

The Problem at hand is to develop a data structure whose size is adjustable as and when required.

II. Implementation

To implement a dynamic array we use an array along with a few other parameters namely size,capacity,increase factor and a decrease factor.

The implementation provides us with certain functionality :

❖ get_size() : Returns the number of elements
❖ get_capacity():Returns capacity of the array.
❖ set_increase_factor_table_size(increasefactor): Sets the value of the increase factor.
❖ set_load_factor_reduction(decrease_factor): Sets the value of the load reduction factor.
❖ append(element) : Appends the element to the end of the array.Allocates memory to the array if in case it is empty initially.As soon as the capacity becomes less than the size it immediately expands the array size by the increase_factor and copies the elements..
❖ pop(element):Removes the Last element of the array.Automatically shrinks the array when it's size reaches a threshold of capacity times the decrease factor and copies the contents.
❖ get(index):Returns the element at the given index if it is valid.Otherwise returns -1.

III. Analysis

Amortized Analysis of Dynamic Array using Accounting Method.By intuition we can say that the amortized cost of inserting into a dynamic array is 3 and popping is 2.We account for the cost of 3 operations in the insert cost itself.The three operations are inserting the actual element into the array(1 credit), copying and moving that element within the array and an extra credit for moving an other element that has already been moved inside the array due to either a contraction or expansion in memory.Now since we know the Amortized cost of each basic operation we can determine the necessary increase and decrease factor.In the aforementioned implementation we have used an increase factor of 2 i.e each time the array becomes full we double it in size and we set a load reduction factor of 0.25 as a threshold and shrink the array by a factor of increase factor times the load reduction factor.This is done to make sure that there is enough credits available to pay for subsequent inserts cum contraction and expansion.As each new element is inserted we accumulate 2 credits when the array has to be expanded we use up all our credits to pay for the expansion but we make sure that we have sufficient credits collected before the next expansion/contraction can occur.

Supposing we have n elements and we double the array size then credits is 0 but before the next expansion happens we would have collected 2n credits which would be sufficient for resizing the array.

Suppose we have n elements and we expand the array.We are left with 0 credits.If in case we try to pop elements we cannot shrink the array until we reach a threshold of n/2.During this period we have collected 3n/2 credits which would be sufficient for resizing the array in subsequent operations.

IV. Conclusion

The amortized cost of the insert operation is O(3),Pop operation is O(2) and all other operations is O(0).Hence we can say that the amortized cost of all operations is constant.