Dynamic Table: Implementation and

Amortized Analysis.

K S Sahazeer

Computer Science Engineering

PES University

Bengaluru, India.

Sahazeer123@gmail.com

***Abstract*—Programming languages such as Java and C++ support ‘vector’ which is a dynamic array that grows or shrink based on the application. Here we tried to analyze dynamic table in detail by considering two operations, namely, insert and delete. Since an application may perform a sequence of insert and delete in some order, it is appropriate to investigate dynamic tables from the perspective of amortized analysis.**

**Keywords: Vector, Load factor, Amortized analysis.**

## **INTRODUCTION**

In computer science, amortized analysis is a method for analyzing given algorithm’s complexity, or how much of time or memory, it takes to execute. The motivation for amortized analysis is that looking for worst-case run time per operation, rather than per algorithm, can be too pessimistic.

# **INSERTION**

Consider a sequence of n insertions. If i th insert does not trigger an expansion, i.e., there is a free slot in the table, then the cost of insert is O (1). Otherwise, the table is full. In such a case, the cost of insert includes the cost of expanding the table to a larger size. Initially, the table size is one, the cost of I1 is one. For I2, since the table is full, we create a new table whose size is twice the size of the previous size. That is, we create a table of size two and insert I2. The cost of I2 is 1 + 1 = 2 (1: for copy, 1: for insert). For I3, the table is full again, we create a new table of size 4 and insert I3. The cost of I3 is 2 + 1 = 3. For I4, it is just one as there is a free slot. For I5, it is 4 + 1 = 5, and so on. In general, if i th insert triggers expansion, then the cost is 2i−1 + 1. We next present the aggregate analysis by considering a sequence of n insertions.

## **Analysis using a potential function**

## Amortized cost= Actual cost + change in potential. We shall work with the following potential function; φ(T) = 2 · Num(T) − Size(T). N = Num(T), the number of elements in table T. S = Size(T), the size of the table T. Initially, Num(T) = 0 and Size(T) = 0. Therefore, φ(T) = 0 Units.

* . If Num(T) = Size(T) =⇒ φ(T) = Num(T).
* If Num(T) = Size(T) 2 =⇒ φ(T) = 0.

Load Factor α = Num(T) Size(T). A measure of how much percentage of table is filled.

ACi = ci + (φi − φi−1) There are two cases possible here, either the i th insertion triggers an expansion or it does not trigger an expansion. Case 1: i th insertion does not trigger an expansion. Si−1 = Si Ni = Ni−1 + 1 ACi = 1 + 2Ni − Si − 2(Ni − 1) + Si−1 = 1 + 2Ni − Si − 2Ni + 2 + Si = 3 Case 2: i th insertion triggers an expansion. Since contents of the old table must be copied to the new table followed by insert, the actual cost is Ni. Si−1 = Si 2 Ni−1 = Ni − 1 ACi = Ni + 2Ni − Si − 2(Ni − 1) + Si−1 = Ni + 2Ni − Si − 2Ni + 2 + Si 2 = Ni + 2 − Si 2 But, Ni = Si 2 + 1 ACi = Si 2 + 1 + 2 − Si 2 = 3 Thus, the amortized cost of insert is 3 = O(1). Therefore, for a sequence of n inserts, the amortized cost is n · O(1) = O(n).

We continue to double the size when an object is inserted into a full table but, we contract the table when deletion causes α < 1 4 . Therefore, 1 4 ≤ α ≤ 1. We now define a new potential function corresponding to the new strategy. Note that φ = 2Ni − Si does not work fine when α < 1 2 and hence we need a different potential function if α is less than 1 2 . φ(T) = 2Num(T) − Size(T) for α ≥ 1 2 = Size(T) 2 − Num(T) for α < 1 2 Now, the load factor α oscillates between 1 4 and 1. Most importantly, the choice of potential function described above is also dictated by α. Observe that, Just before expansion; α = 1, φ = 2Ni − Si = 2Ni − Ni = Ni Just after expansion; α = 1 2 , φ = 2Ni − Si = 2 Si 2 − Si = 0 That is, just after expansion, the potential associated with the table is zero, and each insert increases the potential by 2, when the table is full, the potential is Ni which is sufficient enough to pay for an expansion. Similarly, Just before contraction; α = 1 4 , φ = Si 2 − Ni = Si 2 − Si 4 = Si 4 = Ni Just after contraction; α = 1 2 , φ = 2Ni − Si = 2 Si 2 − Si =0

# **CONCLUSION**

In the above discussion, the contraction is performed when α goes below 1/4 . One can also perform contraction, when α = 1/4 on deletion. Since the amortized cost of each operation is bounded above by a constant, for any sequence of n operations consisting of insert and delete, on a Dynamic Table is O(n) and O(1) amortized per operation. Thus, the strategy yields the desired result.