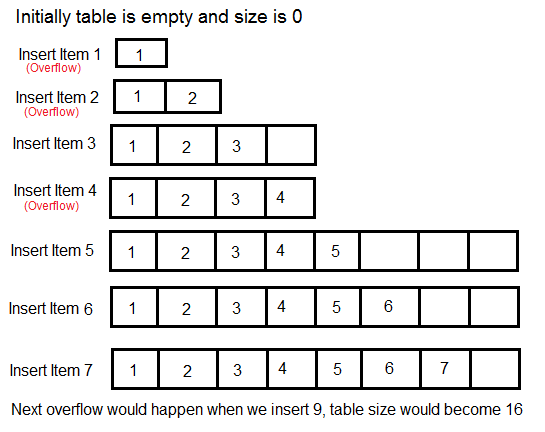
[Amortized Analysis](http://en.wikipedia.org/wiki/Amortized_analysis" \t "_blank) is used for algorithms where an occasional operation is very slow, but most of the other operations are faster. In Amortized Analysis, we analyze a sequence of operations and guarantee a worst-case average time that is lower than the worst-case time of a particularly expensive operation.   
The example data structures whose operations are analyzed using Amortized Analysis are Hash Tables, Disjoint Sets, and Splay Trees.

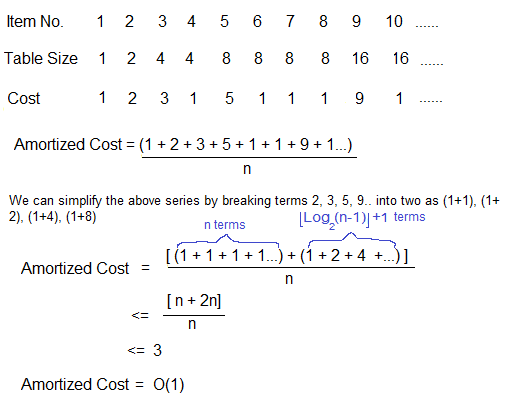
Let us consider an example of simple hash table insertions. How do we decide on table size? There is a trade-off between space and time, if we make hash-table size big, search time becomes low, but the space required becomes high.



The solution to this trade-off problem is to use [Dynamic Table (or Arrays)](http://en.wikipedia.org/wiki/Dynamic_array). The idea is to increase the size of the table whenever it becomes full. Following are the steps to follow when the table becomes full.   
1) Allocate memory for larger table size, typically twice the old table.   
2) Copy the contents of the old table to a new table.   
3) Free the old table.

If the table has space available, we simply insert a new item in the available space.

**What is the time complexity of n insertions using the above scheme?**   
If we use simple analysis, the worst-case cost of insertion is O(n). Therefore, the worst-case cost of n inserts is n \* O(n) which is O(n2). This analysis gives an upper bound, but not a tight upper bound for n insertions as all insertions don’t take Θ(n) time.



So using Amortized Analysis, we could prove that the Dynamic Table scheme has O(1) insertion time which is a great result used in hashing. Also, the concept of the dynamic table is used in [vectors in C++](http://www.cplusplus.com/reference/vector/vector/) and [ArrayList in Java](http://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html).

Following are a few important notes.   
**1)** Amortized cost of a sequence of operations can be seen as expenses of a salaried person. The average monthly expense of the person is less than or equal to the salary, but the person can spend more money in a particular month by buying a car or something. In other months, he or she saves money for the expensive month.

**2)** The above Amortized Analysis was done for Dynamic Array example is called ***Aggregate Method***.

There are two more powerful ways to do Amortized analysis called [***Accounting Method***](http://en.wikipedia.org/wiki/Accounting_method)and [***Potential Method***](http://en.wikipedia.org/wiki/Potential_method).

**3)** The amortized analysis doesn’t involve probability. There is also another different notion of average-case running time where algorithms use randomization to make them faster and the expected running time is faster than the worst-case running time. These algorithms are analyzed using Randomized Analysis. Examples of these algorithms are Randomized Quick Sort, Quick Select and Hashing. We will soon be covering Randomized analysis in a different post.