Design and Analysis of Algorithms
CSE, IIT Kanpur

Worked out Assignment on *Amortized Analysis*

An Important guideline

- It is only through the assignments that one learns the most about the algorithms and data structures. For the current assignment, a sequence of hints are given. Try to look at a hint only after trying on your own for sufficiently long period of time. The onus of learning from a course lies first on you. So act wisely while working on this assignment.

Fully Dynamic Table with worst case $\mathcal{O}(1)$ update time and constant space utilization

We discussed in the course the problem of maintaining dynamic table under insertion or deletion of elements. The main constraints were the following.

1. The dynamic table has to be kept in a contiguous chunk of the RAM. If we change the table, we need to copy all the contents of the existing table into the new table.
2. We wish to maintain space utilization which has to be greater than a constant > 0.
3. The table is initially empty and we have no apriori knowledge of how many elements there will be in future.

In the lectures, we discussed an algorithm for maintaining a fully dynamic table in amortized $\mathcal{O}(1)$ time per update. The space utilization factor was at least 25%. However, the worst case time complexity of an operation could be linear in terms of the number of elements present in the table. This happens when we have to change the table. Note that there are application in the real world that cannot tolerate this inefficiency. So the question arises:

Can we maintain a fully dynamic table in worst case $\mathcal{O}(1)$ time ?

The answer to the above question is in affirmative. In this worked out assignment, we shall provide a sketch of this solution.

Let us first focus on the partial dynamic case. Suppose the table is empty in the beginning and there are only insertions. Spend some time pondering over the solution before the hint on the following page.

The current approach (copying all elements into another table when it becomes full) is a lazy approach. So the first hint is

Shed laziness and start planning well in time ...

Spend some time pondering over the solution before the solution on the following page.

When the table is created, it is half-full. Let us consider one such table T. Once T becomes 75% full, bring in the future table, say T'. Note that T' is empty initially and its size is double the size of T. Henceforth, for each new element to be inserted, insert it into T. Also copy this element and 3 more elements from T into T'. Note that T continue to remain the working-table till T becomes full. Once the T is full, we have all its elements stored in T' as well. So just declare the T' as the working table from now onwards.

What is the space utilization factor ?

Suppose we have a table storing n elements followed by a sequence of deletion of elements. Learn from the solution for insertion given in the previous slide, and design an algorithm that achieves worst case $\mathcal{O}(1)$ time per deletion.

What is the space utilization factor ?

Now let us look at the fully dynamic case. Learn from your solution for the partial dynamic cases given in the previous pages and strive to come up with an algorithm that achieves $\mathcal{O}(1)$ time per update (insertion or deletion of element). The challenge here is that we do not know whether the table will be shrink to one-quarter or become full in future. The biggest hint to handle the challenge is the following:

Plan for future well in time based on the current state of the table.

There may be many solutions that you may come up with. Aim for the simplest and most elegant solution. There is an algorithm which will be almost as complex as the previous ones.

Note:
This part of the assignment (fully dnamic table) is optional. It will not be asked in the 4th quiz or the end semester exam for this semester. But I sincerely hope some of you will still work on it and solve it, and that too perfectly ...