Lecture 14, Oct 25, 2023

Amortized Analysis Example: Dynamic Tables

- Consider a table T occupying a contiguous region in memory, which supports the insert and delete operations; let $\alpha(T)$ be the load factor (ratio of items stored to size)
- When we insert to reach $\alpha(T) = 1$, we have to expand the table by allocating a new table larger than T and then copy over all the elements
  - Typically, we double the size of the table every time it is full
  - With this, $\alpha(T) \geq \frac{1}{2}$ (we don't waste more than half of the table)
  - Each insertion where the table is full costs 1 for insertion and the cost to copy the entire table
- What is the amortized cost per insertion?
  - Starting with aggregate analysis:
    * Starting from an empty table of size 1, what is the cost of n insertions?
    * Notice that after n insertions, we will have expanded the table $\lfloor \log_2 n \rfloor$ times; each expansion costs the same as the size of the table at that point
    * Therefore the total cost of n inserts is n (insertions) plus the sum of all powers of 2 smaller than n (expansions)
    * The cost is $n + \sum_{k=0}^{\lfloor \log_2 n \rfloor} 2^k = n + (2^{\lfloor \log_2 n \rfloor} - 1) \leq n + 2n = 3n$
    * Therefore the amortized cost is $O(1)$
  - Now with the accounting method:
    * Let c_i be the actual cost of the i-th operation and $\hat{c}_i$ be the cost charged for that operation
    * We need $\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$ for all n, or equivalently the credit $\sum_i \hat{c}_i - \sum_i c_i \geq 0$ at all times
    * For a heuristic, consider just having finished an expensive operation so we have no credit, and think about when the next expensive operation will come
      - The expensive operation would be a table expansion; right after an expansion to a table of size n we have half of the entries, $\frac{n}{2}$, being filled
      - The next expensive operation will come in $\frac{n}{2}$ operations when the table is filled completely, which will cost n – giving us an extra charge of $\frac{n}{\frac{n}{2}} = 2$
      - Since inserting the item itself has a cost of 1, we charge 3 per insertion
    * Consider charging 3 per insertion; 1 will go towards the insertion of the element, and the rest 2 are stored as credit
      - 1 credit is attached to the element itself and the last one is attached to another element in the table
      - By doing this, whenever the table is filled completely we will have a credit on every single element
      - Now we can do the copying with the credits attached to the elements, so the credit invariant is maintained
    * For a sequence σ of n inserts, we charge 3 per insert to maintain the credit invariant, so the total cost is $3n$, giving us an amortized cost of $O(1)$
- When we delete elements such that $\alpha(T)$ is too low, we reallocate the table to reduce the amount of memory wasted, so that $\alpha(T) \geq c$, a constant, and to keep the amortized cost per operation constant
  - A naive approach would be to half the size of the table when $\alpha(T) < \frac{1}{2}$
    * This does ensure the load factor is at least 1/2, but if we have alternating insert and delete, we will be doubling and halving the size constantly
    * Consider a sequence σ of $\frac{n}{2}$ inserts, followed by an alternating sequence of 2 inserts and 2 deletes; now every 2 operations will give a cost of $\frac{n}{2}$, so we get $\Omega(n^2)$ complexity or $\Omega(n)$ per

operation

- A better approach would be to half the table when $\alpha(T) \leq \frac{1}{4}$
  * When we move to a smaller table, we will have half the space being filled
  * Regardless of deletion or insertion, the load factor will be 1/2 after moving; this will simplify analysis
  * Consider a sequence σ of inserts and deletes and use the accounting method
    - After an expensive operation, the table will be half full
    - The next expensive operation will be either an expansion or contraction
      - For an expansion, we need another $\frac{n}{2}$ insertions; the expansion costs n, so the averaged out cost is 2 extra per insertion
      - For a contraction, we need another $\frac{n}{4}$ deletions; the contraction costs $\frac{n}{4}$, so the averaged cost is 1 extra per deletion
    - Therefore we charge 3 per insertion and 2 per deletion
  * The amortized cost is then $O(1)$ per operation