Data Structures and Algorithms Spring 2024 — Problem Sets

by Nikolai Kudasov

January 29, 2024

Week 2. Problem set

1. In [Cormen, Section 16.1], a stack with an extra operation MULTIPOP is discussed. What is the total cost of executing n of the stack operations PUSH, POP, and MULTIPOP, assuming that the stack begins with k_0 objects and finishes with k_n objects? Provide brief justification (1–2 sentences).

2. A sequence of PUSH, POP, and SAVE operations is performed on a stack. SAVE operation copies all the elements of the stack that have not been backed up before. To keep track of which elements have a backup, the stack is equipped with a pointer to the most recently pushed element with a backup. PUSH does not affect the pointer, POP only affects the pointer if it pointed to the top element (in this case the pointer will be updated to point to the new top element after POP), and SAVE copies all elements from the pointer to the top of the stack and updates the pointer to point to the top.

   Perform amortised time complexity analysis using the **accounting method** for a sequence of PUSH, POP, and SAVE operations performed on an initially empty stack:

   (a) Specify actual cost, amortized cost, and accumulated credit for each operation. Assume that n_i is the size of stack before operation and k_i is number of backed up elements in the stack.

operation	actual cost	amortized cost	credit
PUSH			
POP			
SAVE			

   (b) Show that the total amortized cost of a sequence of n operations provides an upper bound on the total actual cost of the sequence.

   (c) Write down the asymptotic complexity for a sequence of n operations.

3. **(+1% extra credit)** A sorted collection of n integers is represented by a linked list of k sorted arrays. The arrays are of sizes $2^{b_0}, 2^{b_1}, \ldots, 2^{b_k}$, such that $b_0 < b_1 < \ldots < b_k$.

   To ADD an integer i in a sorted collection, we add a singleton array with i to the beginning of the list of arrays. Then, to ensure the invariant, we resize the arrays: going from smaller arrays to larger, if two smallest arrays have the same size, we MERGE them (using linear time merging as in MERGE-SORT) and repeat the process until the smallest array is unique.

   For example,

   - a collection $\{1, 2, 3, 4, 5, 8, 9, 10, 11, 12\}$ can be represented by a list of three arrays:

     $$\boxed{2} \to \boxed{1, 3} \to \boxed{4, 5, 6, 8, 9, 10, 11, 12}$$

   - inserting 7 in this collection, we first add a singleton array $\boxed{7} \to \boxed{2} \to \boxed{1, 3} \to \boxed{4, 5, 6, 8, 9, 10, 11, 12}$
   - then, since we have two arrays of size 1, we MERGE them $\boxed{2, 7} \to \boxed{1, 3} \to \boxed{4, 5, 6, 8, 9, 10, 11, 12}$
   - then, since we have two arrays of size 2, we MERGE them $\boxed{1, 2, 3, 7} \to \boxed{4, 5, 6, 8, 9, 10, 11, 12}$
   - now, since we have only one array of the smallest size, we stop.

   Perform amortised time complexity analysis using the **potential method** for a sequence of n ADD operations performed on an initially empty sorted collection:

   (a) Define the potential function Φ on a sorted collection[1]; the potential function may depend on the number n of operations in the sequence;

   (b) Compute the value of the potential function for the following sorted collection:

   $$D = \boxed{2} \to \boxed{1, 3} \to \boxed{4, 5, 6, 8, 9, 10, 11, 12}$$

   (c) Compute $\Phi(D_i) - \Phi(D_{i-1})$

   (d) Compute amortized cost for ADD using your potential function;

   (e) Write down amortized asymptotic complexity for ADD.

References

[Cormen] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. *Introduction to Algorithms, Fourth Edition.* The MIT Press 2022

[1] Hint: take inspiration in the potential function for incrementing a binary counter example [Cormen, Section 16.3]; for each element in the collection there should be enough potential for all future merge events for this element.