Mathematics 5365 (Analysis of Algorithms)

Assignment 3: Structurize this!

Submit your solutions typeset in TeX or calligraphed no later than Tuesday, September 18.

1 Background

See Homework 2 for most of the conventions, which remain unchanged here.

Data structures are collections of algorithms that share an (abstract) *persistent state*, a collection of variables that are remain in the memory when different algorithms are run. For example, here is how a traditional array of length n with values in V could be represented as a data structure:

Persistent state: $x : V[n]$. Operations (the first set of arguments denotes the input data, the second set of arguments denotes the output data, i.e., what is computed by the algorithm):

- $\mathrm{get}(i : \mathbf{N})(v : V)$: $v \leftarrow x[i]$;
- $\mathrm{set}(i : \mathbf{N}, v : V)()$: $x[i] \leftarrow v$.

The persistent state is *not* directly accessible to the calling program (e.g., the calling program cannot use the expression $x[i]$, only $\mathrm{get}(i)$). Furthermore, the data structure itself need *not* store its internal state in the given form (e.g., the algorithms that implement get and set need not actually use an array $x : V[n]$, but rather may choose a radically different form of organization, such as a binary search tree, etc.).

2 Problems

1. Persistent state: $n : \mathbf{N}$, $x : \mathbf{B}[n]$ (here x represents a subset $X = \{i \in [0, n) \mid x[i] = 1\} \subset [0, n)$). Operations, with worst-case running time:

- $\mathrm{clear}()()$: $n \leftarrow 0$; $O(n)$;
- $\mathrm{belongs}(i : \mathbf{N})(a : \mathbf{B})$: $a \leftarrow x[i]$; $O(1)$;
- $\mathrm{add}(i : \mathbf{N})()$: $x[i] \leftarrow 1$; $O(1)$;
- $\mathrm{delete}(i : \mathbf{N})()$: $x[i] \leftarrow 0$; $O(n)$;
- $\mathrm{min}()(i : \mathbf{N})$: $i \leftarrow \min_{j \in [0,n), x[j]=1} j$; $O(1)$;
- $\mathrm{isempty}()(a : \mathbf{B})$: $a \leftarrow [n = 0]$; $O(1)$.

2. Same persistent state and operations as above, but the delete operation must use $O(1)$ time, whereas the add operation may use $O(n)$ time.

3. Persistent state: $S : \mathsf{Order}$, $x \subset S$ (here x represents a finite subset of S, where S could be any ordered set, e.g., the real numbers). (This is an example of the abstractness of persistent state: we *must* model the set x using some other representation.) (The last two problems are a special case of this problem for $S = \mathbf{B}$.) We denote the (finite) cardinality of x by $n = \#x$. Operations, with worst-case running time:

- $\mathrm{clear}()()$: $x \leftarrow \emptyset$; $O(1)$;
- $\mathrm{belongs}(s : S)(a : \mathbf{B})$: $a \leftarrow [s \in x]$; $O(\log n)$;
- $\mathrm{add}(s : S)()$: $x \leftarrow x \cup \{s\}$; $O(n)$;
- $\mathrm{delete}(s : S)()$: $x \leftarrow x \setminus \{s\}$; $O(n)$;
- $\mathrm{min}()(s : S)$: $s \leftarrow \min x$; $O(1)$;
- $\mathrm{isempty}()(a : \mathbf{B})$: $a \leftarrow [n = 0]$; $O(1)$.

4. Denote by S the free monoid on $\mathbf{B} = \{0, 1\}$, i.e., the set of all finite sequences of zeros and ones. Persistent state and operations: same as in the previous problem, but without min and isempty. Worst-case running time: clear: $O(1)$; the other three must run in $O(|s|)$, where $|s|$ denotes the length of a finite sequence $s \in S$. Available memory: $O(N)$, where N is the sum of lengths of all sequences in x.

5. Persistent state: $S : \mathsf{Order}$, $n : \mathbf{N}$, $x : S[n]$. Operations, with worst-case running time:

- $\mathrm{clear}(s : S)()$: $x[i] \leftarrow s$ for all $i \in [0, n)$; $O(n)$;
- $\mathrm{get}(i : \mathbf{N})(s : S)$: $s \leftarrow x[i]$; $O(1)$;
- $\mathrm{set}(i : \mathbf{N}, s : S)()$: $x[i] \leftarrow s$; $O(\log n)$;
- $\mathrm{findmin}()(i : \mathbf{N})$: $i \leftarrow \min\{k : \mathbf{N} \mid x[k] = \min_{j \in [0,n)} x[j]\}$; $O(\log n)$.

6. Persistent state: M : Monoid, $l, m, n : \mathbf{N}$, $x : M[l][m][n]$. (Remember that M is not necessarily a group, only a monoid, so there is no subtraction.) Operations, with worst-case running time:

- $\mathrm{clear}(s : M)()$: $x[i][j][k] \leftarrow s$ for all $(i, j, k) \in [0, l) \times [0, m) \times [0, n)$; $O(lmn)$;
- $\mathrm{get}(i, j, k : \mathbf{N})(s : M)$: $s \leftarrow x[i][j][k]$; $O(1)$;
- $\mathrm{set}(i, j, k : \mathbf{N}, s : M)()$: $x[i][j][k] \leftarrow s$; $O((\log l)(\log m)(\log n))$;
- $\mathrm{sum}(l_0, l_1, m_0, m_1, n_0, n_1 : \mathbf{N})(s : M)$: $s \leftarrow \sum_{(i,j,k) \in [l_0,l_1) \times [m_0,m_1) \times [n_0,n_1)}$; $O((\log l)(\log m)(\log n))$.