Syntax and Costs for Sequences, Sets and Tables

Parallel and Sequential Data Structures and Algorithms, 15-210 (Spring 2014)

January 13th, 2014

1 Pseudocode Syntax

The pseudocode we use in the class will use the following notation for operations on sequences, sets and tables. In the translations e, e_1, e_2 represent expressions, and p, p_1, p_2, k, k_1, k_2 represent patterns. The syntax described here is not meant to be complete, but hopefully sufficient to figure out any missing rules. Warning: Since we have been refining the notation as we go, this notation might not be completely consistent across the lectures.

1.1 Sequences

S_i	`nth` S i
$\|S\|$	`length(`S`)`
$\langle\ \rangle$	`empty()`
$\langle v \rangle$	`singleton(`v`)`
$\langle i, \ldots, j \rangle$	`tabulate` (**fn** $k \Rightarrow i + k$) $(j - i + 1)$
$\langle e : p \in S \rangle$	`map` (**fn** $p \Rightarrow e$) S
$\langle e : i \in \langle 0, \ldots, n-1 \rangle \rangle$	`tabulate` (**fn** $i \Rightarrow e$) n
$\langle p \in S \mid e \rangle$	`filter` (**fn** $p \Rightarrow e$) S
$\langle e_1 : p \in S \mid e_2 \rangle$	`map` (**fn** $p \Rightarrow e_1$) (`filter` (**fn** $p \Rightarrow e_2$) S)
$\langle e : p_1 \in S_1, p_2 \in S_2 \rangle$	`flatten(map` (**fn** $p_1 \Rightarrow$ `map` (**fn** $p_2 \Rightarrow e$) S_2) S_1)
$\langle e_1 : p_1 \in S_1, p_2 \in S_2 \mid e_2 \rangle$	`flatten(map` (**fn** $p_1 \Rightarrow \langle e_1 : p_2 \in S_2 \mid e_2 \rangle$) S_1)
$\sum_{p \in S} e$	`reduce add 0 (map` (**fn** $p \Rightarrow e$) S)
$\sum_{i=k}^{n} e$	`reduce add 0 (map` (**fn** $i \Rightarrow e$) $\langle k, \ldots, n \rangle$)
$\underset{p \in S}{\text{argmax}}(e)$	`argmax compare (map` (**fn** $p \Rightarrow e$) S)

The meaning of `add`, `0`, and `compare` in the `reduce` and `argmax` will depend on the type. The $\sum$ can be replaced with min, max, $\cup$ and $\cap$ with the presumed meanings. The function `argmax f S : ` $(\alpha \times \alpha \rightarrow$ `order`$) \rightarrow (\alpha$ `seq`$) \rightarrow$ `int` returns the index in S which has the maximum value with respect to the order defined by the function f. $\text{argmin}_{p \in S}\, e$ can be defined by reversing the order of compare.

1.2 Sets

$\|S\|$	`size`(S)
$\{\}$	`empty`
$\{v\}$	`singleton`(v)
$\{v \in S \mid e\}$	`filter` $(\textbf{fn}\ v \Rightarrow e)$ S
$S_1 \cup S_2$	`union`(S_1, S_2)
$S_1 \cap S_2$	`intersection`(S_1, S_2)
$S_1 \setminus S_2$	`different`(S_1, S_2)
$\sum_{k \in S} e$	`reduce add 0 (Table.tabulate` $(\textbf{fn}\ k \Rightarrow e)$ S`)`

1.3 Tables

$\|T\|$	`size`(T)
$\{\}$	`empty()`
$\{k \mapsto v\}$	`singleton`(k, v)
$\{e : v \in T\}$	`map` $(\textbf{fn}\ v \Rightarrow e)$ T
$\{k \mapsto e : (k \mapsto v) \in T\}$	`mapk` $(\textbf{fn}\ (k, v) \Rightarrow e)$ T
$\{k \mapsto e : k \in S\}$	`tabulate` $(\textbf{fn}\ k \Rightarrow e)$ S
$\{v \in T \mid e\}$	`filter` $(\textbf{fn}\ v \Rightarrow e)$ T
$\{(k \mapsto v) \in T \mid e\}$	`filterk` $(\textbf{fn}\ (k, v) \Rightarrow e)$ T
$\{e_1 : v \in T \mid e_2\}$	`map` $(\textbf{fn}\ v \Rightarrow e_1)$ `(filter` $(\textbf{fn}\ v \Rightarrow e_2)$ T`)`
$\{k : (k \mapsto \_) \in T\}$	`domain`(T)
$\{v : (\_ \mapsto v) \in T\}$	`range`(T)
$T_1 \cup T_2$	`merge` $(\textbf{fn}\ (v_1, v_2) \Rightarrow v_2)$ (T_1, T_2)
$T \cap S$	`extract`(T, S)
$T \setminus S$	`erase`(T, S)
$\sum_{v \in T} e$	`reduce add 0 (map` $(\textbf{fn}\ v \Rightarrow e)$ T`)`
$\sum_{(k \mapsto v) \in T} e$	`reduce add 0 (mapk` $(\textbf{fn}\ (k, v) \Rightarrow e)$ T`)`
$\underset{(k \mapsto v) \in T}{\text{argmax}}(e)$	`argmax max (mapk` $(\textbf{fn}\ (k, v) \Rightarrow e)$ T`)`

2 Function Costs

ArraySequence	*Work*	*Span*
`length`(T) `singleton`(v) `nth` S i `empty()`	1	1
`tabulate` f n	$\sum_{i=0}^{n-1} W(f(i))$	$\max_{i=0}^{n-1} S(f(i))$
`map` f S	$\sum_{e \in S} W(f(e))$	$\max_{e \in S} S(f(e))$
`map2` f S_1 S_2	$\sum_{i=0}^{\min(\|S_1\|,\|S_2\|)-1} W(f(S_{1i}, S_{2i}))$	$\max_{i=0}^{\min(\|S_1\|,\|S_2\|)-1} S(f(S_{1i}, S_{2i}))$
`filter` f S	$\sum_{s \in S} W(f(s))$	$\log \|S\| + \max_{s \in S} S(f(s))$
`reduce` f b S	$O\left(\|S\| + \sum_{f(x,y) \in \mathcal{O}_r(f,b,S)} W(f(x,y))\right)$	$\log \|S\| \max_{f(x,y) \in \mathcal{O}_r(f,b,S)} S(f(x,y))$
`scan` f b S	$O\left(\|S\| + \sum_{f(x,y) \in \mathcal{O}_s(f,b,S)} W(f(x,y))\right)$	$\log \|S\| \max_{f(x,y) \in \mathcal{O}_s(f,b,S)} S(f(x,y))$
`iter` f b_0 S	$O\left(\sum_{i=0}^{\|S\|-1} W(f(b_i, S_i))\right)$	$\sum_{i=0}^{\|S\|-1} S(f(b_i, S_i))$
`iterh` f b_0 S	$O\left(\sum_{i=0}^{\|S\|-1} W(f(b_i, S_i))\right)$	$\sum_{i=0}^{\|S\|-1} S(f(b_i, S_i))$
`showt` S `showti` S f	$\|S\|$	1
`showl` S	$\|S\|$	1
`hidet(NODE`(L,R)`)`	$\|L\| + \|R\|$	1
`hidel(CONS`(x, xs)`)`	$\|S\|$	1

`hidel(NIL)`
`hidet(ELT` e`)`
`hidet(EMPTY)`
	1	1

ArraySequence	*Work*	*Span*
`append`(S_1, S_2)	$\lvert S_1\rvert + \lvert S_2\rvert$	1
`drop`(S, n)	$\lvert S\rvert - n$	1
`take`(S, n) `drop`(S, n) `subseq` $S\ (s, n)$	n	1
`rake` $S\ (a, b, s)$	$\frac{\lvert b-a\rvert}{s}$	1
`splitMid`(S, i)	$\lvert S\rvert$	1
`flatten` S	$\lvert S\rvert + \sum_{e \in S} \lvert e\rvert$	$\log \lvert S\rvert$
`inject` $I\ S$	$\lvert I\rvert + \lvert S\rvert$	1
`partition` $I\ S$	$\lvert I\rvert + \lvert S\rvert$	1
`argmax` $f\ S$	$\lvert S\rvert$	$\log \lvert S\rvert$
`merge` $f\ S_1\ S_2$	$\lvert S_1\rvert + \lvert S_2\rvert$	$\log(\lvert S_1\rvert + \lvert S_2\rvert)$
`sort` $f\ S$	$\lvert S\rvert \log \lvert S\rvert$	$\log^2 \lvert S\rvert$
`collate` $f\ (S_1, S_2)$	$\lvert S_1\rvert + \lvert S_2\rvert$	$\log(\min(\lvert S_1\rvert, \lvert S_2\rvert))$
`collect` $f\ S$	$\lvert S\rvert \log \lvert S\rvert$	$\log^2 \lvert S\rvert$
`fromList`(S) `%`(S)	$\lvert S\rvert$	$\lvert S\rvert$
`toString` $f\ S$	$\sum_{e \in S} W(f(e))$	$\sum_{e \in S} S(f(e))$
`fields` $f\ S$ `tokens` $f\ S$	$\lvert S\rvert$	$\log \lvert S\rvert$

For `reduce`, $\mathcal{O}_r(f, i, S)$ represents the set of applications of f as defined in the documentation. For `scan`, $\mathcal{O}_s(f, i, S)$ represents the applications of f defined by the implementation of scan in the lecture notes. For `iter` and `iterh`, $b_i = f(b_{i-1}, S_{i-1})$. For `showti`, `argmax`, `merge`, `sort`, `collate`, `collect`, `fields`, and `tokens` the given costs assume that the work and span of the application of f is constant.

TreeSequence	*Work*	*Span*
`nth` $S\ i$	$\log n$	$\log n$
`tabulate` $f\ n$	$---$	$\log n + \max_{i=0}^{n} S(f(i))$
`map` $f\ S$	$---$	$\log \lvert S \rvert + \max_{s \in S} S(f(s))$
`showt` S	$\log \lvert S \rvert$	$\log \lvert S \rvert$
`hidet(NODE(`L,R`))`	$\log(\lvert L \rvert + \lvert R \rvert)$	$\log(\lvert L \rvert + \lvert R \rvert)$
`append(`S_1, S_2`)`	$\log(\lvert S_1 \rvert + \lvert S_2 \rvert)$	$\log(\lvert S_1 \rvert + \lvert S_2 \rvert)$
`drop(`S, n`)`	$\log(\lvert S \rvert - n)$	$\log(\lvert S \rvert - n)$
`take(`S, n`)` `subseq` $S\ (s, n)$	$\log n$	$\log n$
`partition` $I\ S$	$\sum_{p \in S} p$	$\log(\lvert I \rvert + \lvert S \rvert)$
`inject` $I\ S$	$\lvert I \rvert \lg(\lvert I \rvert + \lvert S \rvert)$	$\lg^2 \lvert I \rvert + \log \lvert S \rvert$
`merge` $f\ S_1\ S_2$	$\min(\lvert S_1 \rvert, \lvert S_2 \rvert) \cdot \lg(\lvert S_1 \rvert + \lvert S_2 \rvert)$	$\lg(\lvert S_1 \rvert + \lvert S_2 \rvert)$
`sort` $f\ S$	$\lvert S \rvert \log \lvert S \rvert$	$\log^2 \lvert S \rvert$
`collect` $f\ S$	$\lvert S \rvert \log \lvert S \rvert$	$\log^2 \lvert S \rvert$

For `singleton`, `length`, `filter`, `reduce`, `scan`, `sort` and `collect` the costs are the same as in ArraySequence. All $---$ entries are the same as ArraySequence. For `merge`, `sort`, and `collect` the costs assume that the work and span of the application of f is constant.

Single Threaded ArraySequence	*Work*	*Span*
`nth` $S\ i$ `update` $(i, v)\ S$	$O(1)$	$O(1)$
`inject` $I\ S$	$\lvert I \rvert$	1
`fromSeq` S `toSeq` S	$O(\lvert S \rvert)$	$O(1)$

Tree Sets and Tables	*Work*	*Span*
`size`(T) `singleton`(k,v)	$O(1)$	$O(1)$
`filter` f T	$O\left(\sum_{(k,v)\in T} W(f(v))\right)$	$O\left(\lg\lvert T\rvert + \max_{(k,v)\in T} S(f(v))\right)$
`map` f T	$O\left(\sum_{(k,v)\in T} W(f(v))\right)$	$O\left(\max_{(k,v)\in T} S(f(v))\right)$
`tabulate` f S	$O\left(\sum_{k\in S} W(f(k))\right)$	$O\left(\max_{k\in S} S(f(k))\right)$
`find` T k `insert` f (k,v) T `delete` k T	$O(\lg\lvert T\rvert)$	$O(\lg\lvert T\rvert)$
`merge` f (T_1,T_2) `extract` (T,S) `erase` (T,S)	$O\left(m\lg(\frac{n+m}{m})\right)$	$O\left(\lg(n+m)\right)$
`domain` T `range` T `toSeq` T	$O(\lvert T\rvert)$	$O(\lg\lvert T\rvert)$
`collect` S `fromSeq` S	$O(\lvert S\rvert\lg\lvert S\rvert)$	$O(\lg^2\lvert S\rvert)$
`union` (S_1,S_2) `intersection` (S_1,S_2) `difference` (S_1,S_2)	$O\left(m\lg(\frac{n+m}{m})\right)$	$O\left(\lg(n+m)\right)$

where $n = \max(\lvert T_1\rvert, \lvert T_2\rvert)$ and $m = \min(\lvert T_1\rvert, \lvert T_2\rvert)$. For `reduce` you can assume the cost is the same as `Seq.reduce f init (range(T))`. In particular `Seq.reduce` defines a balanced tree over the sequence, and `Table.reduce` will also use a balanced tree. For `merge` and `insert` the bounds assume the merging function has constant work.

TreeTables

function	*type*	*Work*	*Span*
`size`(T) `singleton`(k, v)	$\mathbb{T} \to \mathbb{N}$ $\mathbb{K} \times \alpha \to \mathbb{T}_\alpha$	1	1
`filter` $f\ T$	$(\alpha \to \mathbb{B}) \to \mathbb{T}_\alpha \to \mathbb{T}_\alpha$	$\sum_{(k,v)\in T} W(f(v))$	$\lg\lvert T\rvert + \max_{(k,v)\in T} S(f(v))$
`map` $f\ T$	$(\alpha \to \beta) \to \mathbb{T}_\alpha \to \mathbb{T}_\beta$	$\sum_{(k,v)\in T} W(f(v))$	$\max_{(k,v)\in T} S(f(v))$
`tabulate` $f\ T$	$(\mathbb{K} \to \alpha) \to \mathbb{S} \to \mathbb{T}_\alpha$	$\sum_{k\in S} W(f(k))$	$\max_{k\in S} S(f(k))$
`find` $T\ k$ `insert` $f\ (k, v)\ T$ `delete` $k\ T$	$\mathbb{T}_\alpha \to \mathbb{K} \to (\mathbb{K}\ \text{opt})$ $(\alpha \times \alpha \to \alpha) \to (\mathbb{K} \times \alpha) \to \mathbb{T}_\alpha \to \mathbb{T}_\alpha$ $\mathbb{K} \to \mathbb{T}_\alpha \to \mathbb{T}_\alpha$	$\lg\lvert T\rvert$	$\lg\lvert T\rvert$
`merge` $f\ (T_1, T_2)$ `extract` (T, S) `erase` (T, S)	$(\alpha \times \alpha \to \alpha) \to (\mathbb{T}_\alpha \times \mathbb{T}_\alpha) \to \mathbb{T}_\alpha$ $\mathbb{T}_\alpha \times \mathbb{S} \to \mathbb{T}_\alpha$ $\mathbb{T}_\alpha \times \mathbb{S} \to \mathbb{T}_\alpha$	$m \lg(\frac{n+m}{m})$	$\lg(n+m)$
`domain` T `range` T `toSeq` T	$\mathbb{T}_\alpha \to \mathbb{S}$ $\mathbb{T}_\alpha \to (\alpha\ \texttt{Seq})$ $\mathbb{T}_\alpha \to ((\mathbb{K} \times \alpha)\ \texttt{Seq})$	$\lvert T\rvert$	$\lg\lvert T\rvert$
`collect` S `fromSeq` S	$(\mathbb{K} \times \alpha)\ \texttt{Seq} \to \mathbb{T}_{\alpha\texttt{Seq}}$ $(\mathbb{K} \times \alpha)\ \texttt{Seq} \to \mathbb{T}_\alpha$	$\lvert S\rvert \lg\lvert S\rvert$	$\lg^2\lvert S\rvert$

TreeSets

function	*type*	*Work*	*Span*
`size`(S) `singleton`(k)	$\mathbb{S} \to \mathbb{N}$ $\mathbb{K} \to \mathbb{S}$	1	1
`filter` $f\ S$	$(\mathbb{K} \to \mathbb{B}) \to \mathbb{S} \to \mathbb{S}$	$\sum_{k\in S} W(f(k))$	$\lg\lvert S\rvert + \max_{k\in S} S(f(k))$
`find` $S\ k$ `insert` $k\ S$ `delete` $k\ S$	$\mathbb{S} \to \mathbb{K} \to \mathbb{B}$ $\mathbb{K} \to \mathbb{S} \to \mathbb{S}$ $\mathbb{K} \to \mathbb{S} \to \mathbb{S}$	$\lg\lvert S\rvert$	$\lg\lvert S\rvert$
`fromSeq` S	$\mathbb{K}\ \texttt{Seq} \to \mathbb{S}$	$\lvert S\rvert \log\lvert S\rvert$	$\lg^2\lvert S\rvert$
`union` (S_1, S_2) `intersection`(S_1, S_2) `difference`(S_1, S_2)	$\mathbb{S} \times \mathbb{S} \to \mathbb{S}$ $\mathbb{S} \times \mathbb{S} \to \mathbb{S}$ $\mathbb{S} \times \mathbb{S} \to \mathbb{S}$	$m \lg(\frac{n+m}{m})$	$\lg(n+m)$

Where $n = \max(|T_1|, |T_2|)$ and $m = \min(|T_1|, |T_2|)$. For `reduce` you can assume the cost is the same as `Seq.reduce f init (range(T))`. In particular `Seq.reduce` defines a balanced tree over the sequence, and `Table.reduce` will also use a balanced tree. For `merge` and `insert` the bounds assume the merging function has constant work.

3 Further Help

As always, don't hesitate to ask a TA for help if you're unclear about anything!