15-210: Parallel and Sequential Data Structures and Algorithms
Syntax and Costs for Sequences, Sets and Tables

October 25, 2011

1 Psuedocode Syntax

In the pseudocode in the class we 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 improving the notation as we go, this notation is not completely backward compatible with earlier lectures in the course.

Sequences

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

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` : $(\alpha \times \alpha \rightarrow \texttt{order}) \rightarrow (\alpha\ \texttt{seq}) \rightarrow \texttt{int}$ returns the index in S which has the maximum value with respect to the order defined by the function f. $\operatorname{argmin}_{p \in S} e$ can be defined by reversing the order of compare.

Sets

S_v	$\mathit{find}\ S\ v$
$\|S\|$	$\mathit{size}(S)$
$\{\}$	empty
$\{v\}$	$\mathit{singleton}(v)$
$\{p \in S \mid e\}$	$\mathit{filter}\ (\textbf{fn}\ p \Rightarrow e)\ S$
$S_1 \cup S_2$	$\mathit{union}(s_1, s_2)$
$S_1 \cap S_2$	$\mathit{intersection}(s_1, s_2)$
$S_1 \setminus S_2$	$\mathit{different}(s_1, s_2)$
$\sum_{k \in S} e$	$\mathit{reduce}\ \mathit{add}\ 0\ (\mathit{Table.tabulate}\ (\textbf{fn}\ k \Rightarrow e)\ S)$

Tables

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

2 Function Costs

ArraySequence	*Work*	*Span*
`length`(T) `singleton`(v) `nth` $S\ i$	$O(1)$	$O(1)$
`tabulate` $f\ n$	$O\left(\sum_{i=0}^{n} W(f(i))\right)$	$O\left(\max_{i=0}^{n} S(f(i))\right)$
`map` $f\ S$	$O\left(\sum_{s \in S} W(f(s))\right)$	$O\left(\max_{s \in S} S(f(s))\right)$
`filter` $f\ S$	$O\left(\sum_{s \in S} W(f(s))\right)$	$O\left(\log \|S\| + \max_{s \in S} S(f(s))\right)$
`reduce` $f\ i\ S$	$O\left(\|S\| + \sum_{f(a,b) \in \mathcal{O}_r(f,i,S)} W(f(a,b))\right)$	$O\left(\log \|S\| \max_{f(a,b) \in \mathcal{O}_r(f,i,S)} S(f(a,b))\right)$
`scan` $f\ i\ S$	$O\left(\|S\| + \sum_{f(a,b) \in \mathcal{O}_s(f,i,S)} W(f(a,b))\right)$	$O\left(\log \|S\| \max_{f(a,b) \in \mathcal{O}_s(f,i,S)} S(f(a,b))\right)$
`showt` S	$O(\|S\|)$	$O(1)$
`hidet` `NODE`(l, r)	$O(\|l\| + \|r\|)$	$O(1)$
`append`(S_1, S_2)	$O(\|S_1\| + \|S_2\|)$	$O(1)$
`flatten`(S)	$O\left(\|S\| + \sum_{s \in S} \|s\|\right)$	$O(1)$
`partition` $I\ S$	$O(\|I\| + \|S\|)$	$O(1)$
`inject` $I\ S$	$O(\|S\|)$	$O(1)$
`merge` $f\ \|S_1\|\ \|S_2\|$	$O(\|S_1\| + \|S_2\|)$	$O(\log(\|S_1\| + \|S_2\|))$
`sort` $f\ S$	$O(\|S\| \log \|S\|)$	$O\left(\log^2 \|S\|\right)$
`collect` $f\ S$	$O(\|S\| \log \|S\|)$	$O\left(\log^2 \|S\|\right)$
Single Threaded Array Sequence		
`nth` $S\ i$ `update` $(i, v)\ S$	$O(1)$	$O(1)$
`inject` $I\ S$	$O(\|I\|)$	$O(1)$
`fromSeq` S `toSeq` S	$O(\|S\|)$	$O(1)$

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 `merge`, `sort`, and `collect` the costs assume that the work and span of the application of f is constant.

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

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.