# PARALLEL PROGRAMMING: 1
## *(DRAFT NOTES)*

RAUL H.C. LOPES

## 1. P+1-ARY SEARCH

This section presents a (p+1)-ary search algorithm, for searching an ordered sequence for a given item, using $p$ processors. Dijkstra presents in [1] a derivation of a binary search algorithm that can be easily generalized to a (p+1)-ary search with $p$ processors.

1.1. **Generalizing Dijkstra' binary search.** Given an

$$X \in \mathbf{Int}$$

and

$$S.0..M-1 \in \mathbf{Int} \text{ where } \forall i,j : 0 \leq i < j < M : S.i < S.j$$

one is required to calculate *present* such that

$$present \leq \exists i : 0 \leq i < M : X = S.i$$

The existential quantifier in the formula above is better understood from a constructive point of view: it demands that an $i$ is produced that will validate $X = S.i$, or else *present* will be false. The message that quantifier to be sending is is: *find and $i$ that validates $x = S.i$ or prove that any legal $i$ makes $X = S.i$ a contradiction.*[1]

Determining *present* is then relaxed to finding an $i$

$$R \leq 0 \leq i < M \wedge S.i \leq X < S.i + 1$$

assuming that $S.M$ is defined to be greater than $X$.

Dijkstra's program has the form

$$
\begin{array}{l}
|[\mathbf{Var}\ i \in \mathbf{Int} \\
\quad ;\ establish\ R \\
\quad ;\ present := (S.i = X) \\
]|
\end{array}
$$

[1]Dijkstra[1] says in his argument that it "is hard to visualize an algorithm correctly establishing present ... without establishing for some integer" $i$ such that $S.i = X \wedge 0 \leq i < M$.

Generalizing $R$ by replacing $(i+1)$ by $j$ gives an invariant for the case when $X$ occurs in $S$:

$$0 \leq i < j \leq M \wedge S.i \leq X < S.j$$

That together with the fact $X$ may be less than $S.0$ gives the invariant

$$P \leq 0 \leq i < j \leq M \wedge (S.i \leq X < S.j \vee X < S.0)$$

That invariant establishes at any transition that $S.i..j-1$ must be searched for a pair of indexes $(i,j)$ delimiting the area where $X$ can be found in $S$. A progress condition should state that the area to be scanned is decreasing

$$(j-i) > (j-i)'$$

and a fixed point should bound progress: it doesn't make much sense to search a singleton subsequence:

$$(j-i) = 1$$

"establish R" is then refined to

$|[$**Var** $j \in$ **Int**
 ; $i, j := 0, M$
 ; $\{P\}$
 $*|[$
  $j - i \neq 1 \rightarrow shrink(S, j, i)\{P\}$
 $]|\{R\}$
$]|\ \{R\}$

Shrinking $(j-i)$ can be done by either increasing $i$ to some $h$, if $S.h \leq X$, or decreasing $j$ to $h$, if $S.h < X$, or both simultaneously, why not?

$shrink(S, X, i, j)$ **is**
 $|[$**Var** $h \in$ **Int**
  ; $h := (i+j)/2$
  ; **if** $S.h \leq X \rightarrow i := h$
   $[]\ X < S.h \rightarrow j := h$
   **fi**
  $]|$

**Exercise 1.**

(1) *Prove the invariant $P$ holds for the program above.*

(2) *Prove the program terminates.*
(3) *Prove that it correctly determines $q$.*

1.2. **Generalizing to (p+1)-ary search.** Determining *present* is now relaxed to finding an interval with size up to $p$ that can be tested in parallel for the presence of $X$.

$$R \leq 0 \leq i < M \wedge S.i \leq X < S.min(i+p, M)$$

assuming that $S.M$ is defined to be greater than $X$.

The program becomes

$|[$**Var** $i \in$ **Int**
 ; *establish* $R$
 ; $|[||\ i : i \leq k < min(i+p, M)$ :
  $q := (X = S.k)$
 $]|$
$]|$

The loop invariant is the same. The fixed point now is reached when $(j-i)$ is less than or equal to $p$.

"establish R" is then refined to

$|[$**Var** $j \in$ **Int**
 ; $i, j := 0, M$
 ; $\{P\}$
 $*|[$
  $j - i > p \rightarrow pshrink(S, j, i, p)\{P\}$
 $]|\{R\}$
$]|\ \{R\}$

Shrinking $(j-i)$ can be performed by either

- increasing $i$ to some $h$, if $S.h \leq X$;
- or decreasing $j$ to $h$, if $S.h < X$;
- or both simultaneously.

This is performed by a parallel test.

$pshrink(S, X, i, j)$ **is**
 $|[$**Var** $h \in$ **Int**
  ; $nel := (i+j)/(p+1)$
  ; $|[||\ k : 1 \leq k \leq p$ :

$$
\begin{array}{l}
\quad\quad\quad\quad \textbf{Var } h, hb \in \textbf{Int} \\
\quad\quad\quad\quad ;\ h, hb := i + (nel * k), i + (nel * (k-1)) \\
\quad\quad\quad\quad ;\ \textbf{if } k = p \wedge X \geq S.h \rightarrow i := h \\
\quad\quad\quad\quad \quad [\!]\ S.hb \leq X \wedge X < S.h \rightarrow i, j := hb, h \\
\quad\quad\quad\quad \quad \textbf{fi} \\
\quad\quad ]| \\
\quad ]|
\end{array}
$$

**Exercise 2.**
*Prove $P$ is still an invariant for the new program.*
*Evaluate its complexity in terms of: time (i.e, number of parallel transitions or supersteps), and work.*

## 2. Quicksort

**Exercise 3.** *Derive a quicksort algorithm that splits its input in $p+1$ intervals, when assigned p processors. Use a parallel scan operation for splitting around p pivots.*

## 3. Sequences

**Exercise 4.**

(1) *Given sequences $a.0..n-1 \in$* **Int**, *and $b.0..n-1 \in$* **Int**, *and $s.0 = 0$, give a program to compute, for $1 \leq i < n$:*
   (a) $s.i = (a.i * s.i - 1) + b.i$
   (b) $s.i = (a.i * s.i - 1) + (b.i * s.i - 2)$

(2) *Given a nondecreasing sequence $a.0..n-1 \in$* **Int**, *and a number $b \in$* **Int**, *derive a program to find the rank of b in a: the index of the largest element of a that is smaller than b.*

(3) *Derive a program to find the rank of all elements of a sequence $b.0..m-1 \in$* **Int**, *in a nondecreasing sequence $b.0..n-1 \in$* **Int**.

(4) *Derive a program to merge 2 nondecreasing sequences.*

## References

[1] Edsger W. Dijkstra, *Fillers at the YoP intitute*, in *Formal Development of Programs and Proofs* [2].

[2] Edsger W. Dijkstra (ed.), *Formal development of programs and proofs*, Addison-Wesley, 1990.