

Recursion in LabVIEW

Tomi Maila

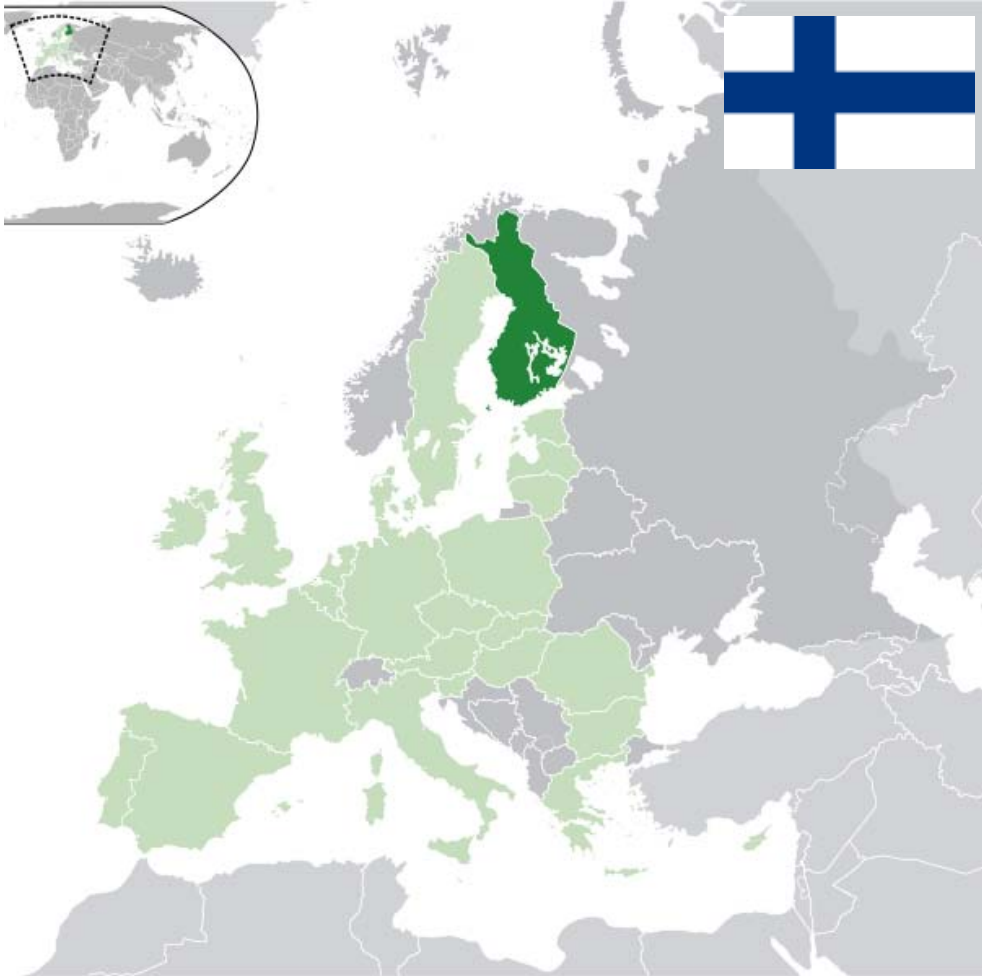
2/24/2010

LabVIEW User Group Meeting

Mountain View, CA



My Background



Tomi Maila

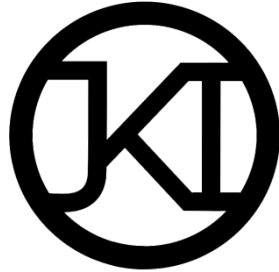
Senior Software Engineer
James Kring Inc.

LabVIEW Champion

LabVIEW blogger

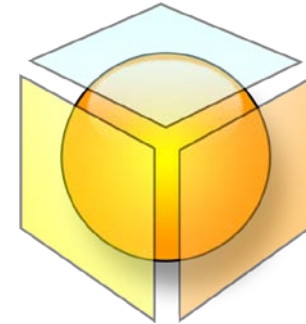
<http://expressionflow.com>





JKI Consulting

enable our customers to
be highly successful on the
LabVIEW platform



JKI Software

make significant and
innovative contributions
to LabVIEW and its
community

3



Types of recursion

- Recursive algorithms
- Structural recursion
- Type recursion



Types of recursion

- Recursive algorithms
- Structural recursion
- ~~Type recursion~~



- Recursive algorithms
- Structural recursion
- ~~Type recursion~~



Introduction of recursion in LabVIEW

- General support introduced in LabVIEW 2009
- Class member dynamic dispatch VI supporting recursion since LabVIEW 8.5



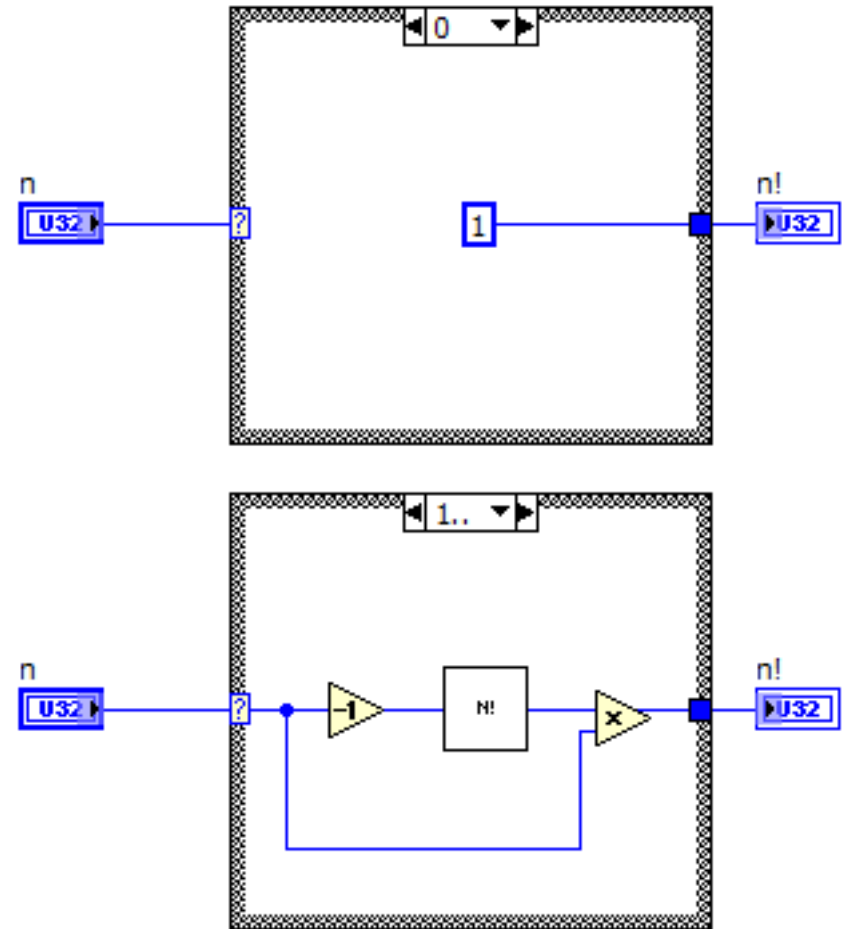
Factorial definition

$$\text{fact}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot \text{fact}(n - 1) & \text{if } n > 0 \end{cases}$$



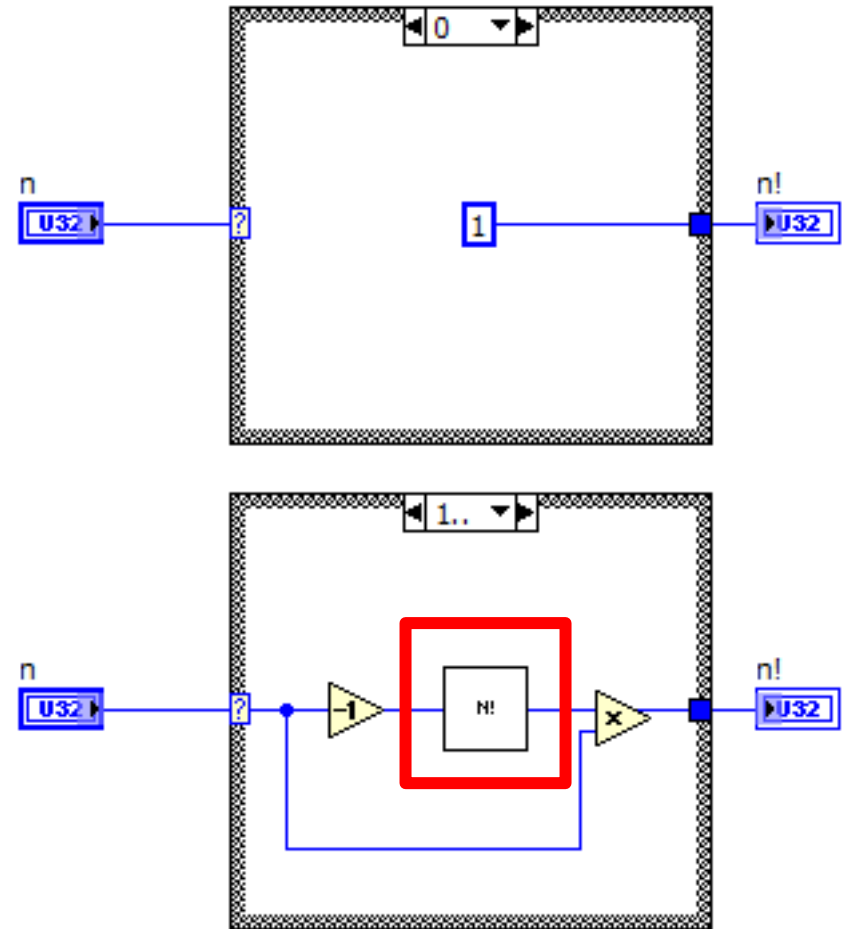
Recursive factorial in LabVIEW

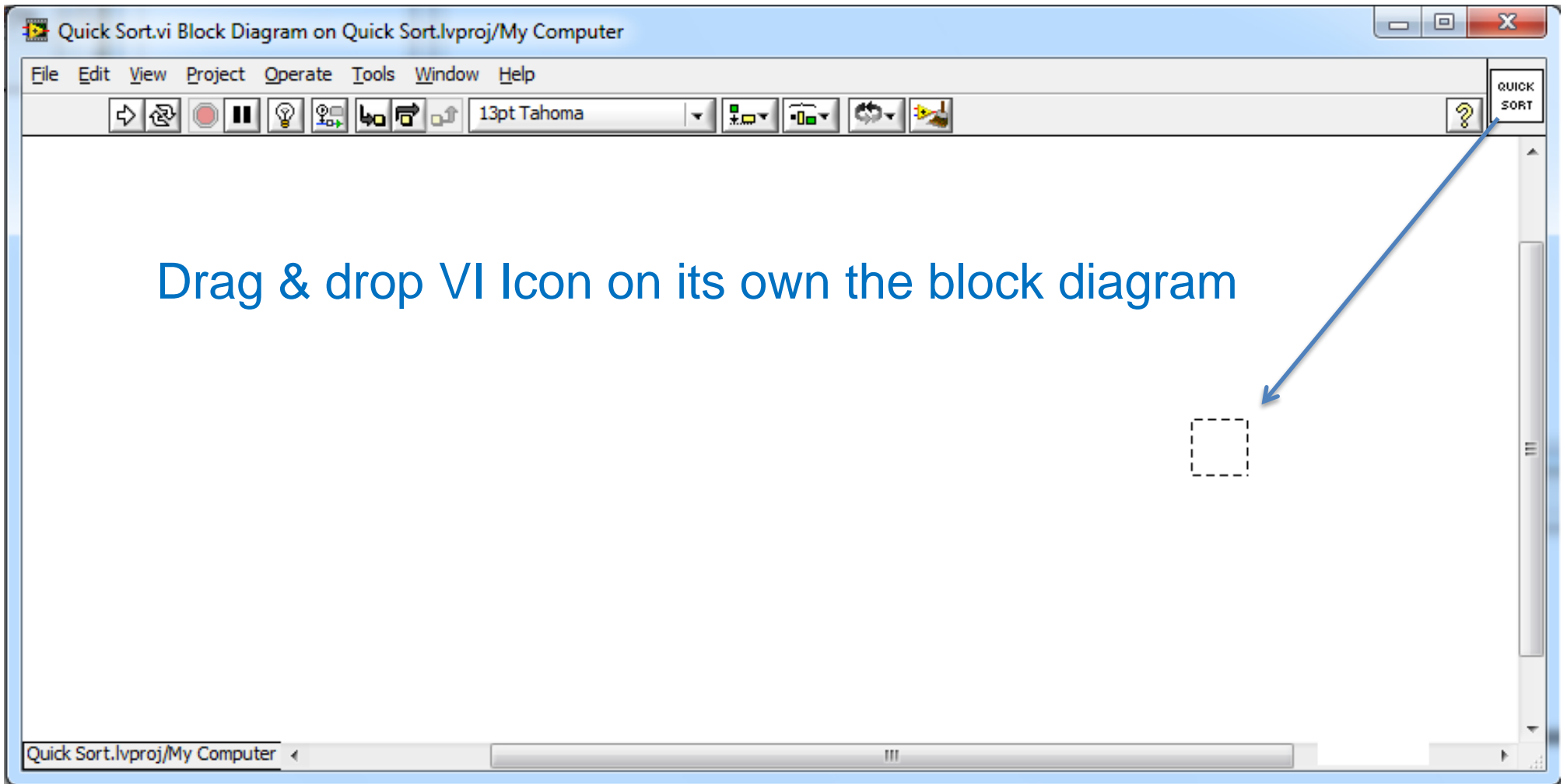
$$\text{fact}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot \text{fact}(n - 1) & \text{if } n > 0 \end{cases}$$



Recursive factorial in LabVIEW

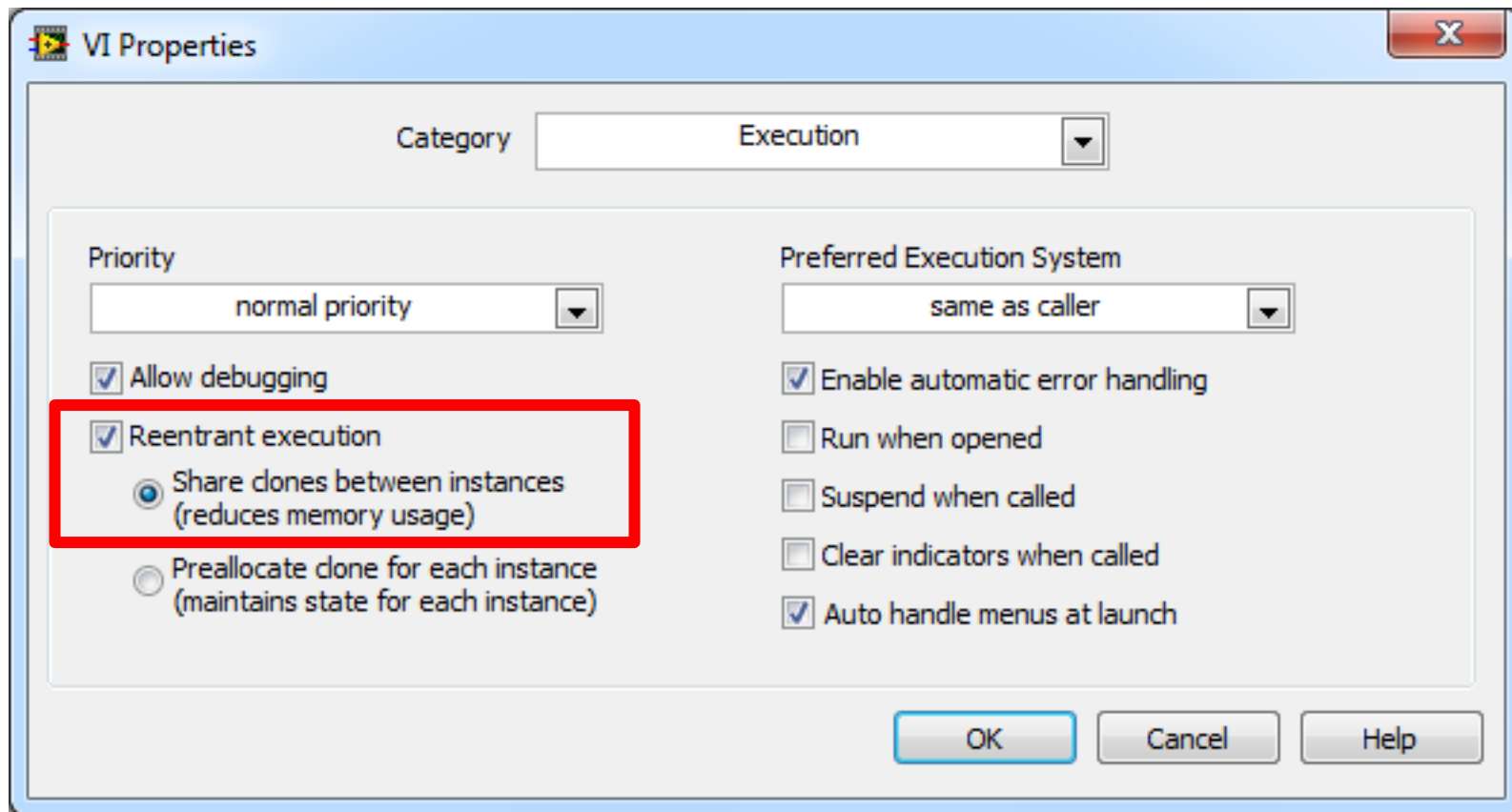
$$\text{fact}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \cdot \text{fact}(n - 1) & \text{if } n > 0 \end{cases}$$



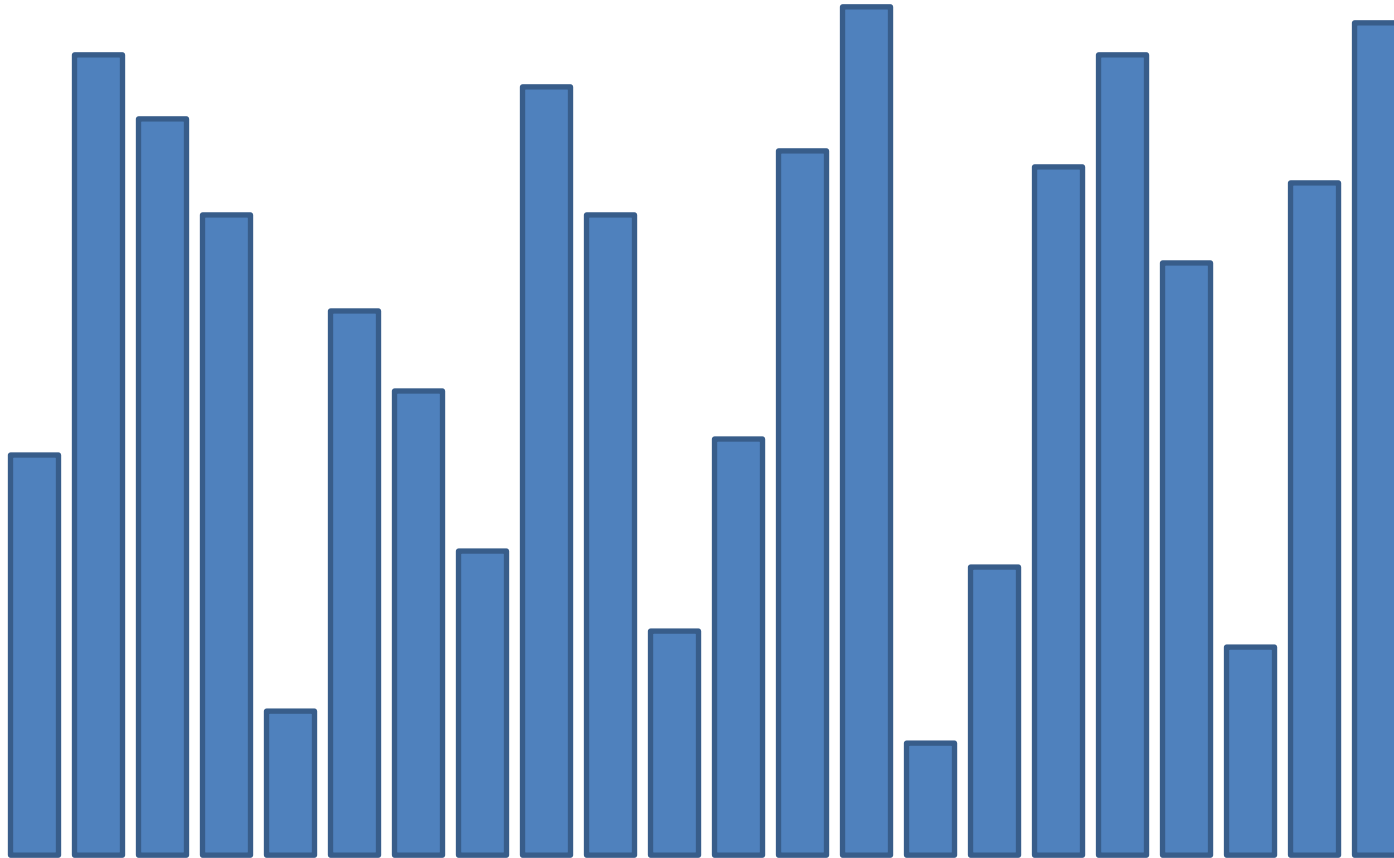


Preparing VI for recursive execution

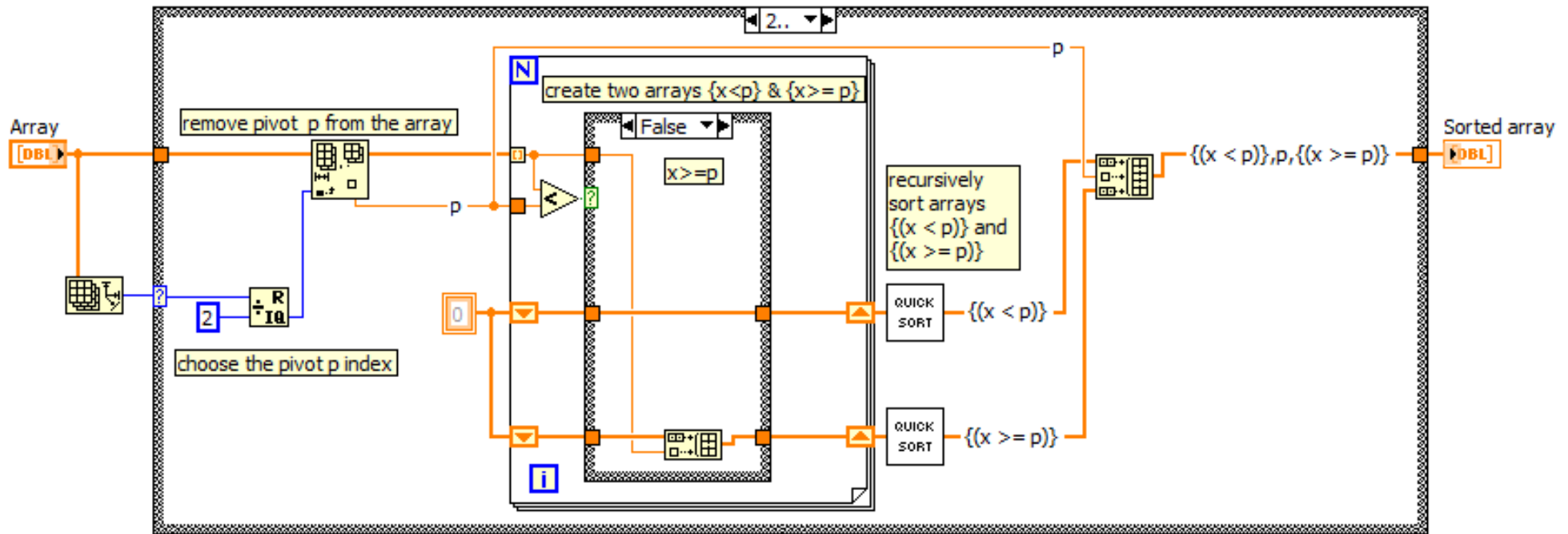
CTRL + I



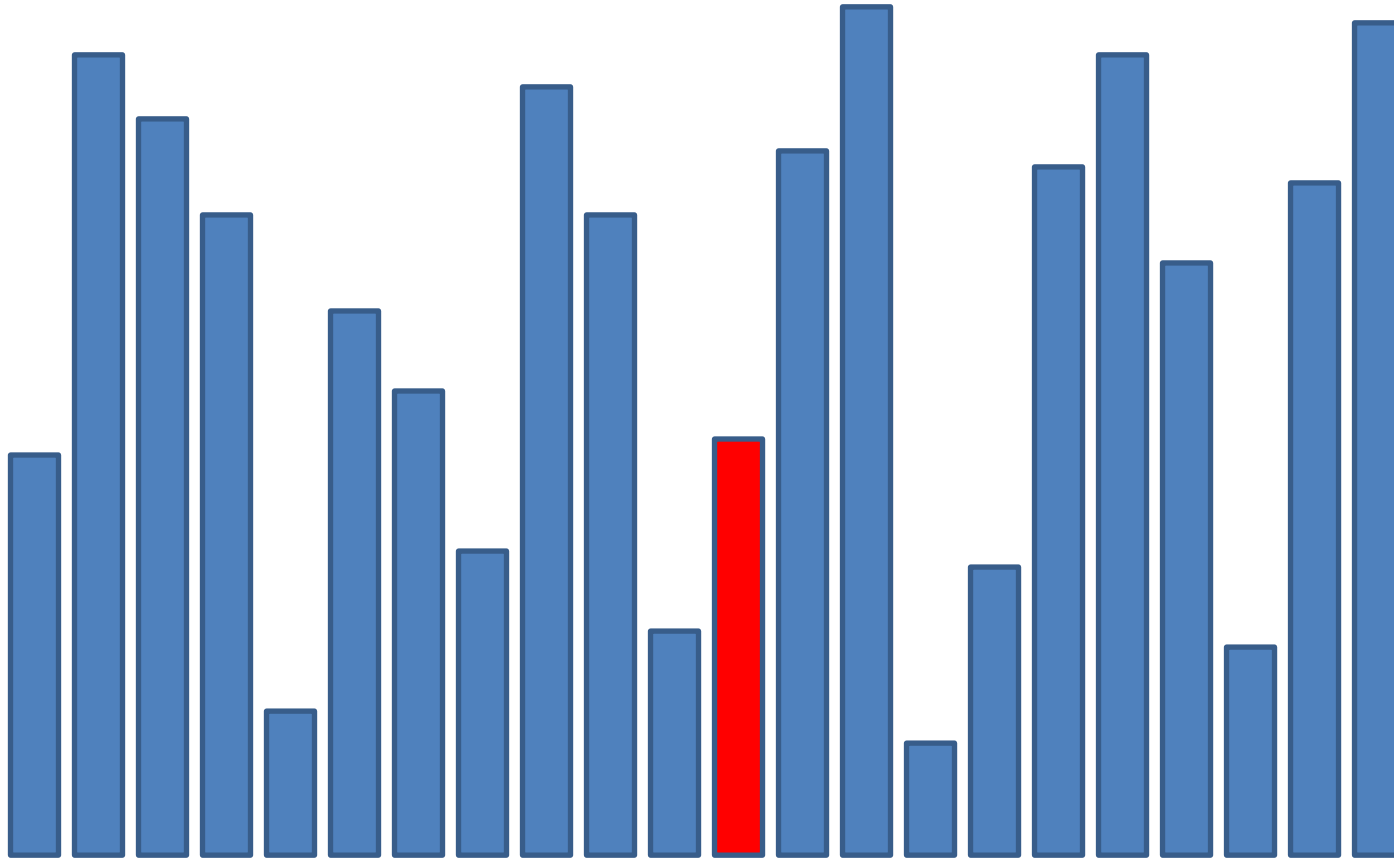
Sorting unordered lists with Quick Sort



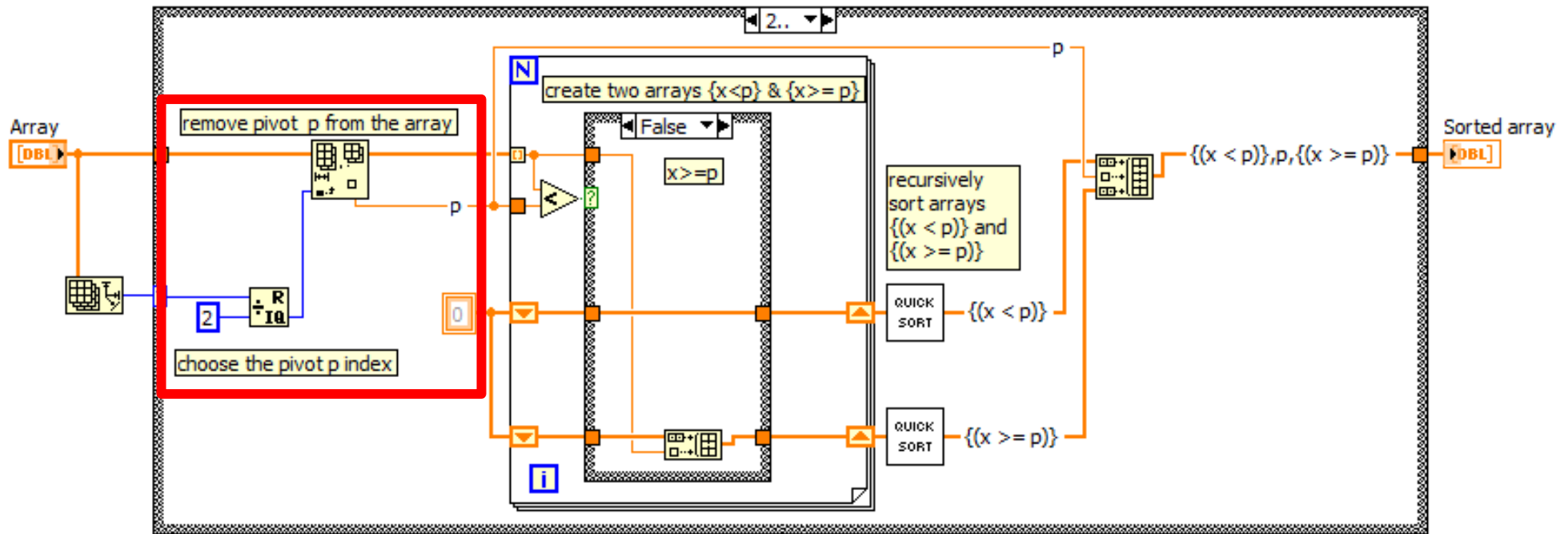
Quick sort implementation in LabVIEW



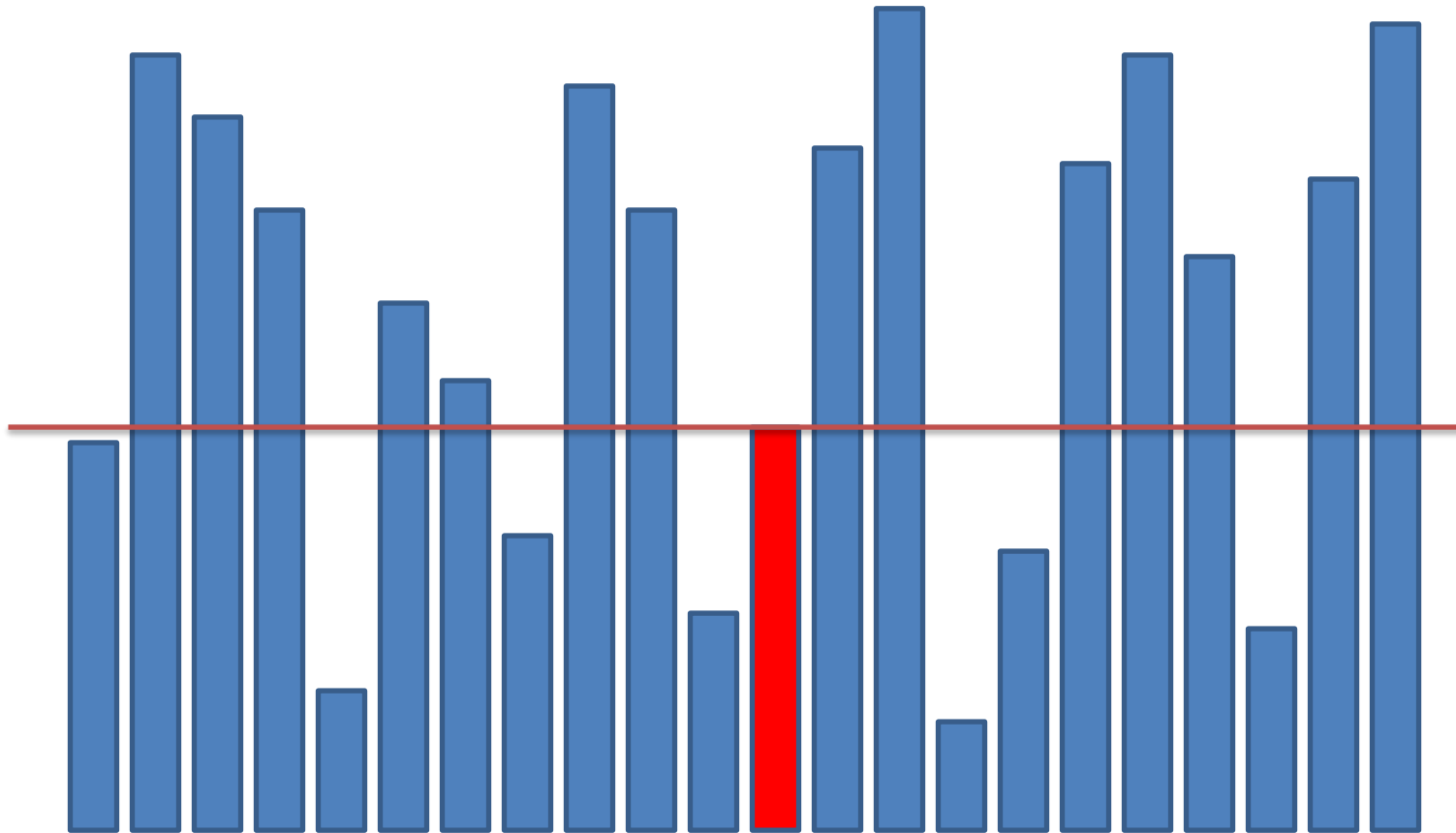
Choose a pivot



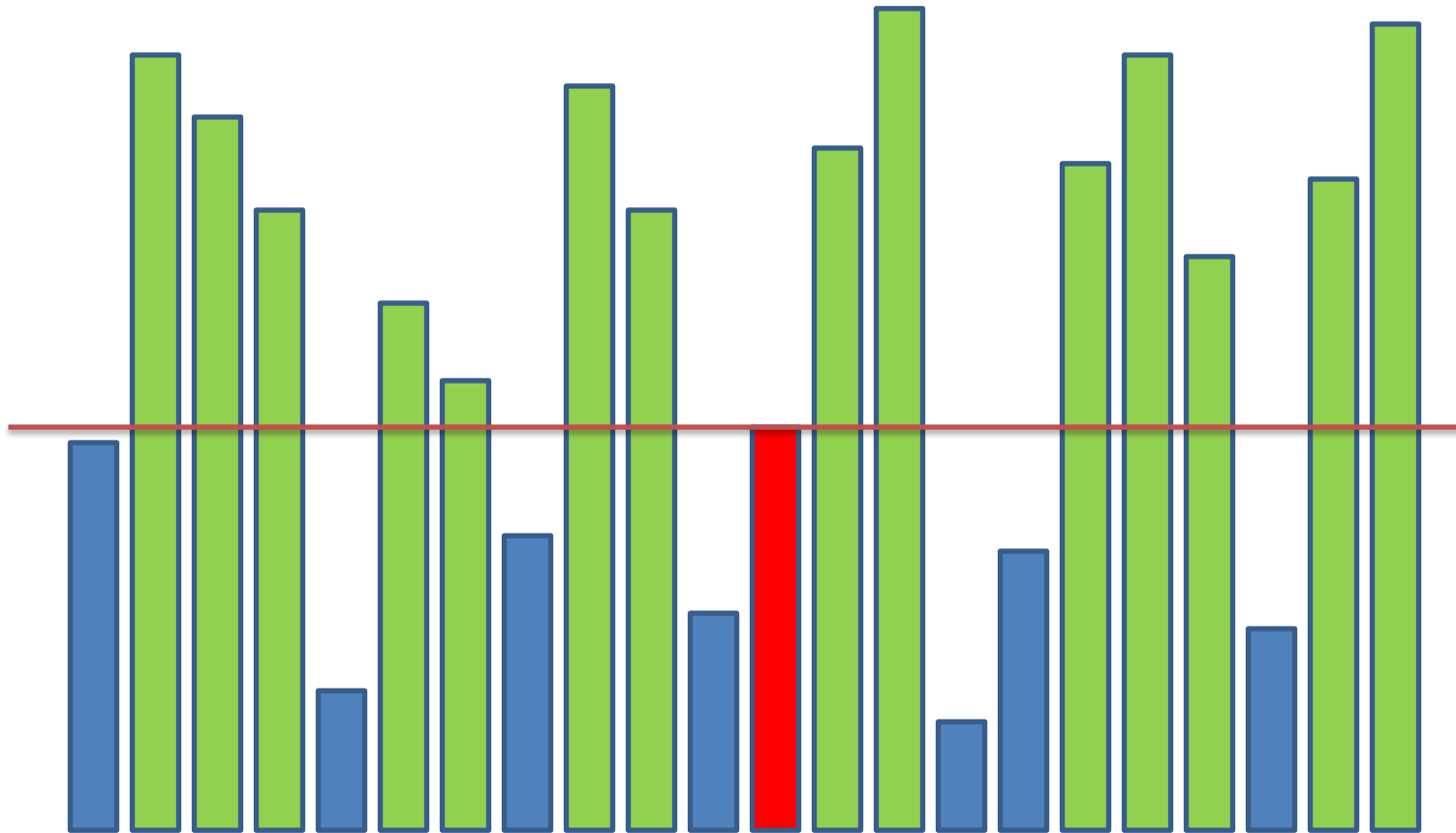
Quick sort implementation in LabVIEW



Compare all items against pivot

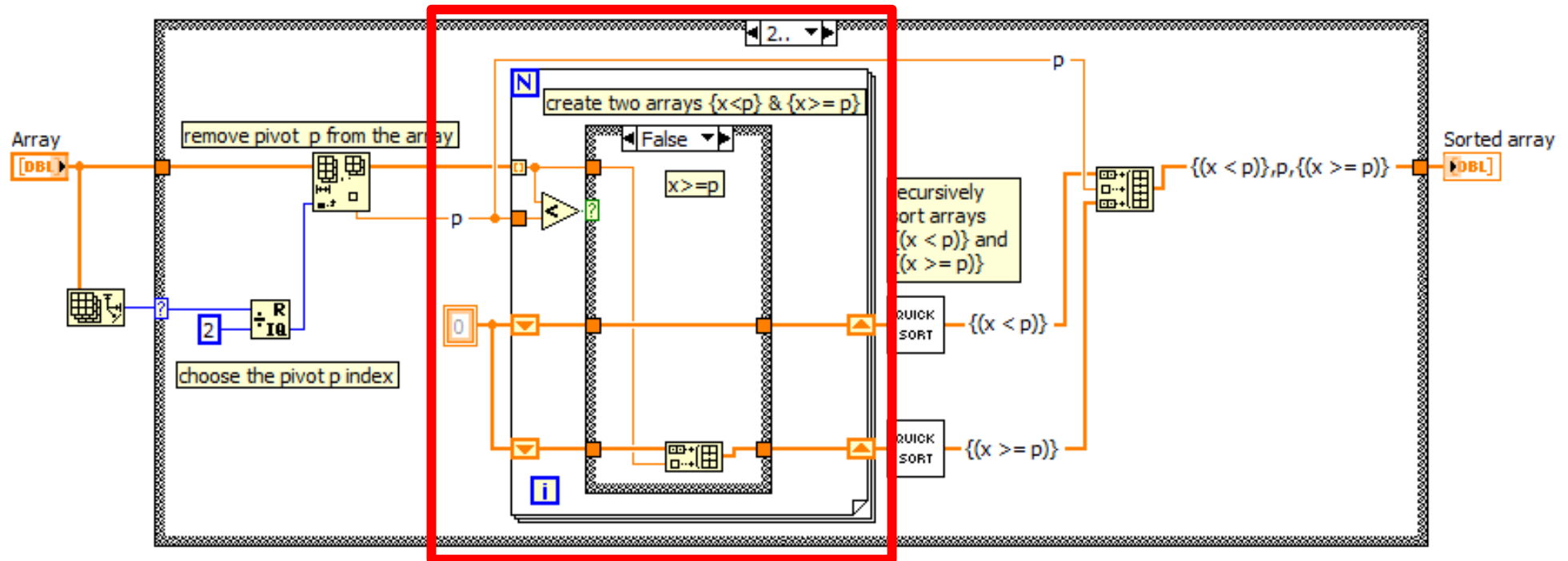


Create two lists smaller and larger than pivot

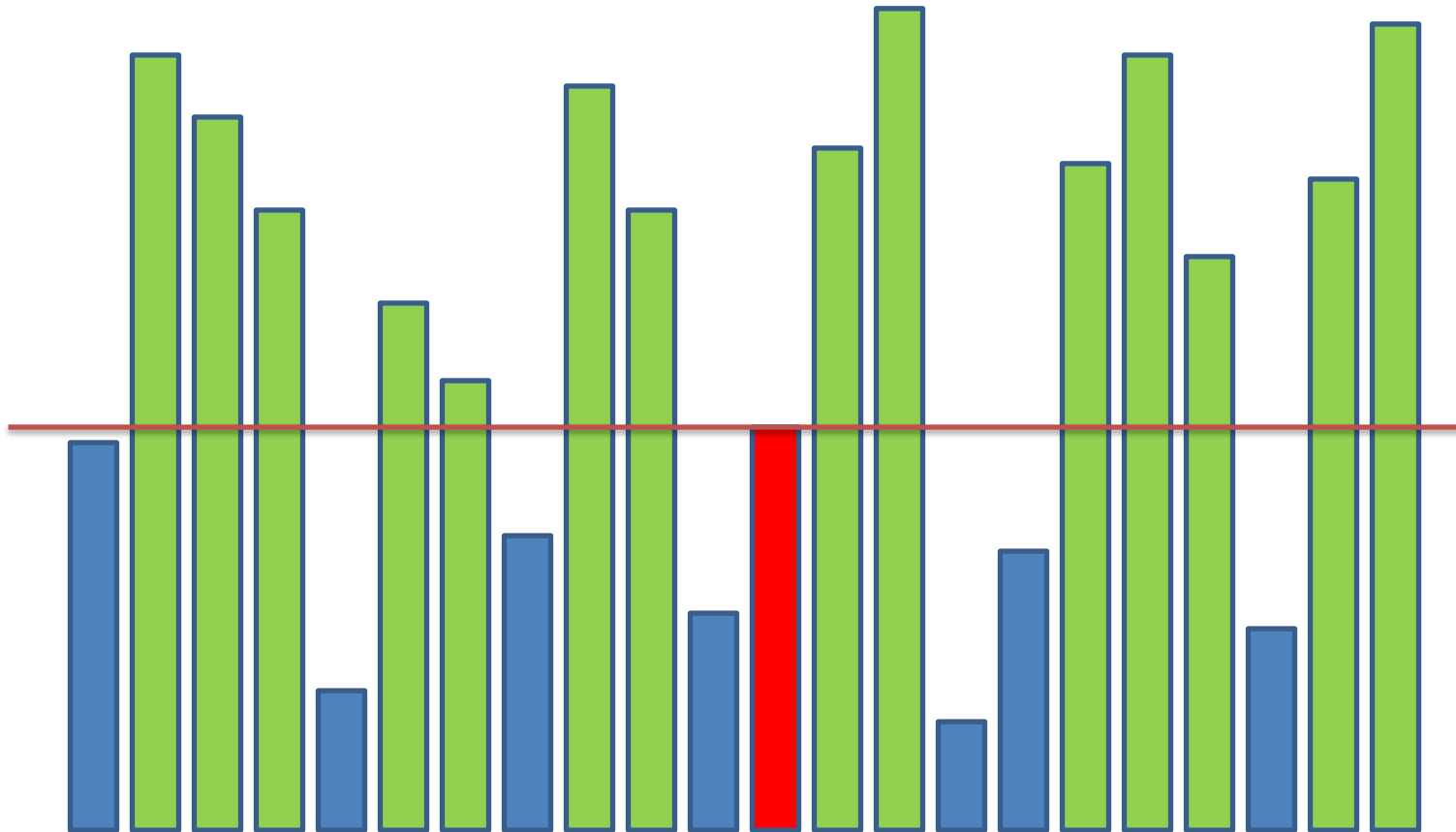


Recursion in LabVIEW

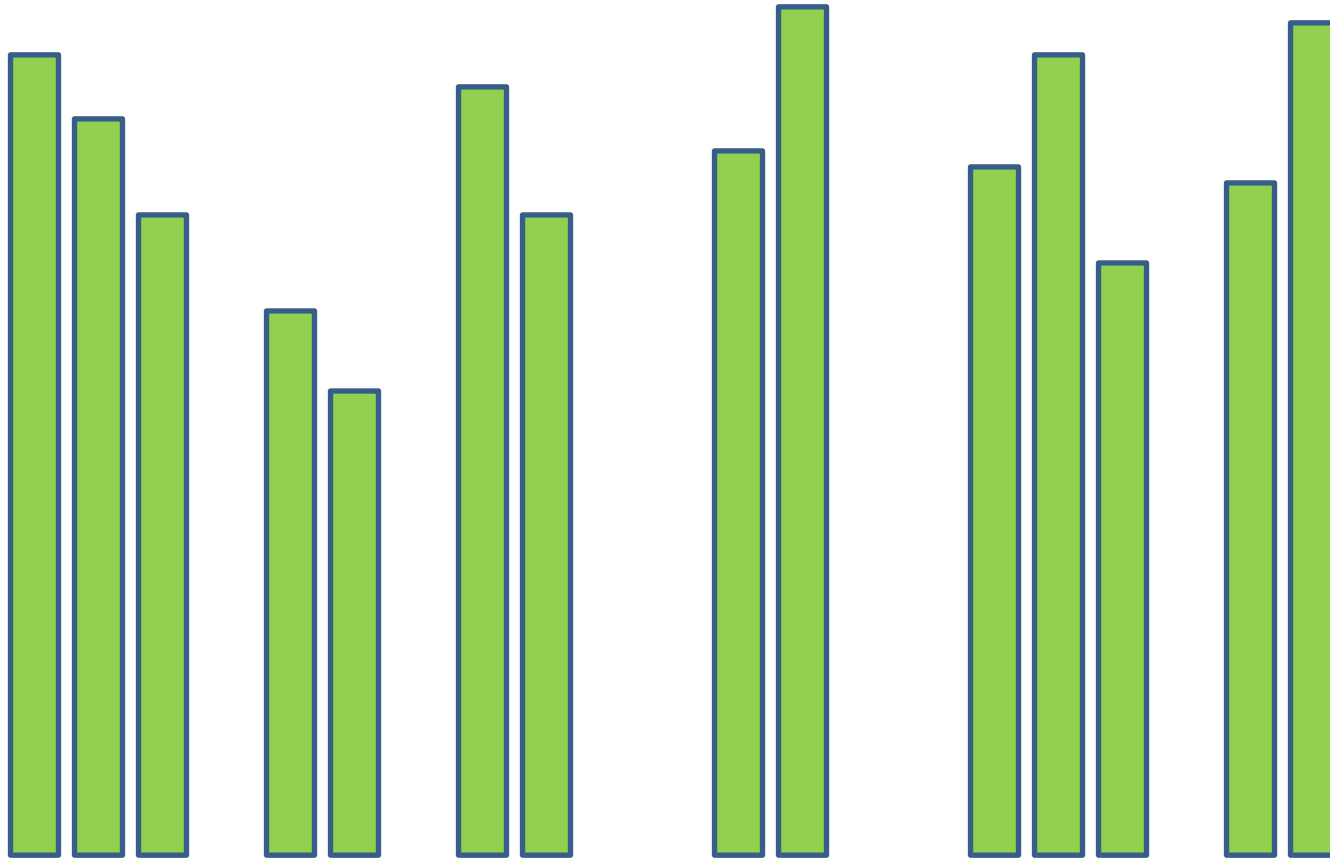
Quick sort



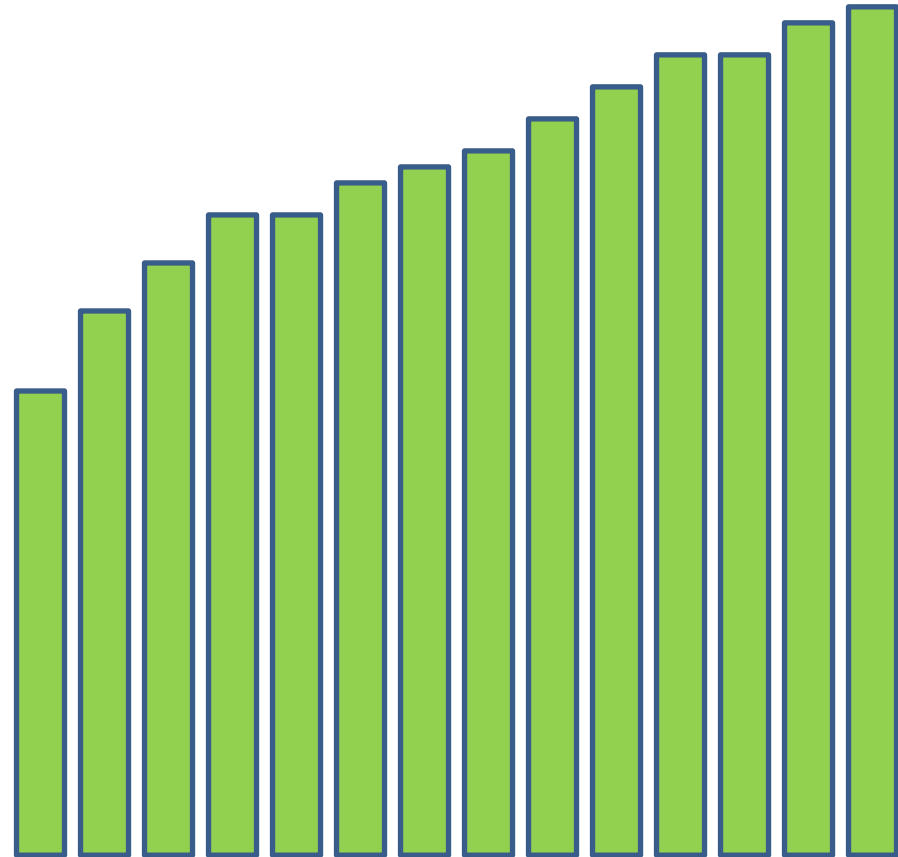
Create two lists smaller and larger than pivot



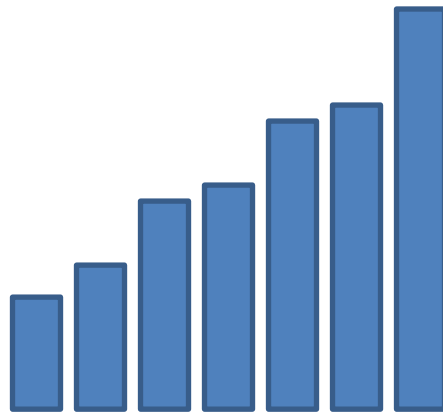
Items larger than pivot

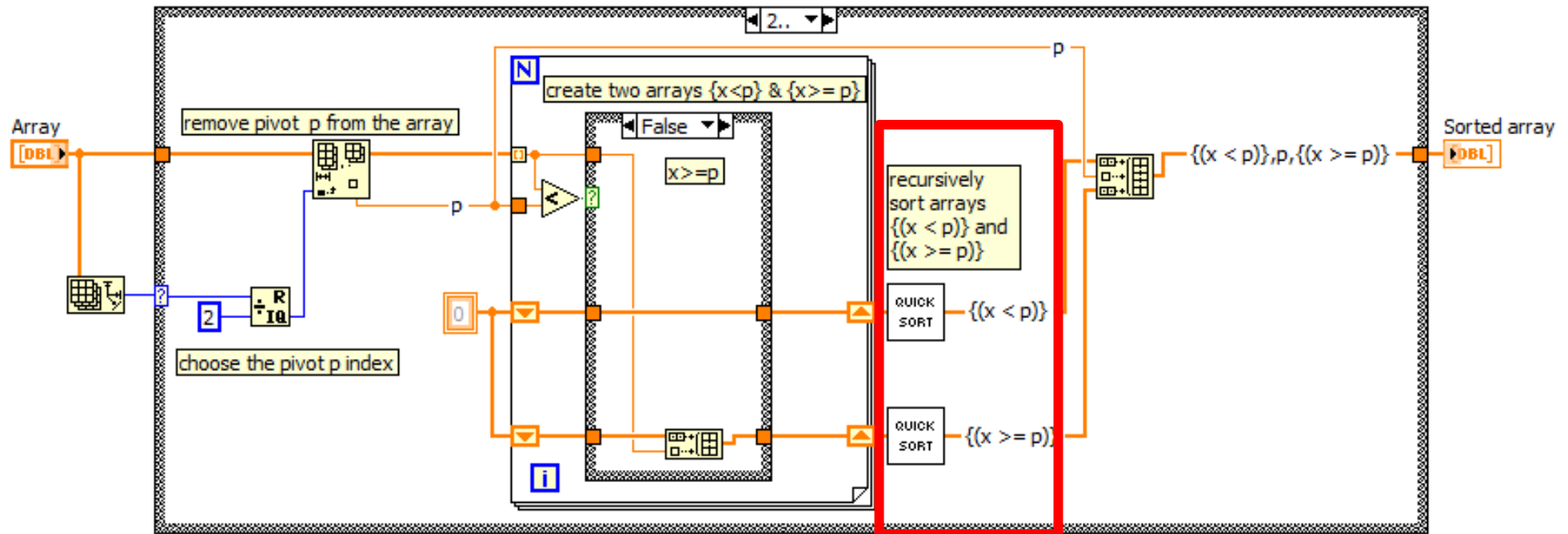


Call Quick Sort recursively to sort the list

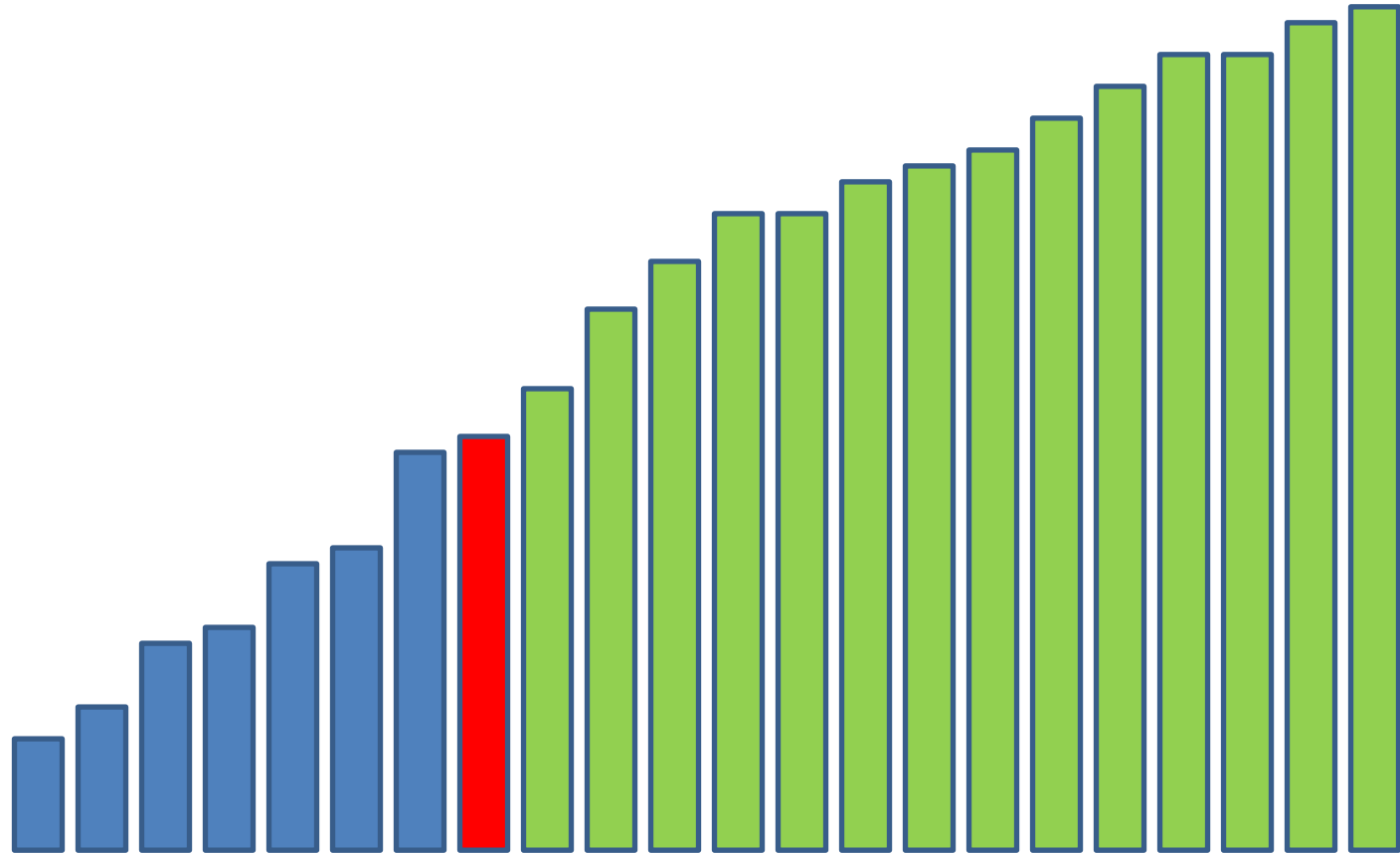


Call Quick Sort recursively to sort the list



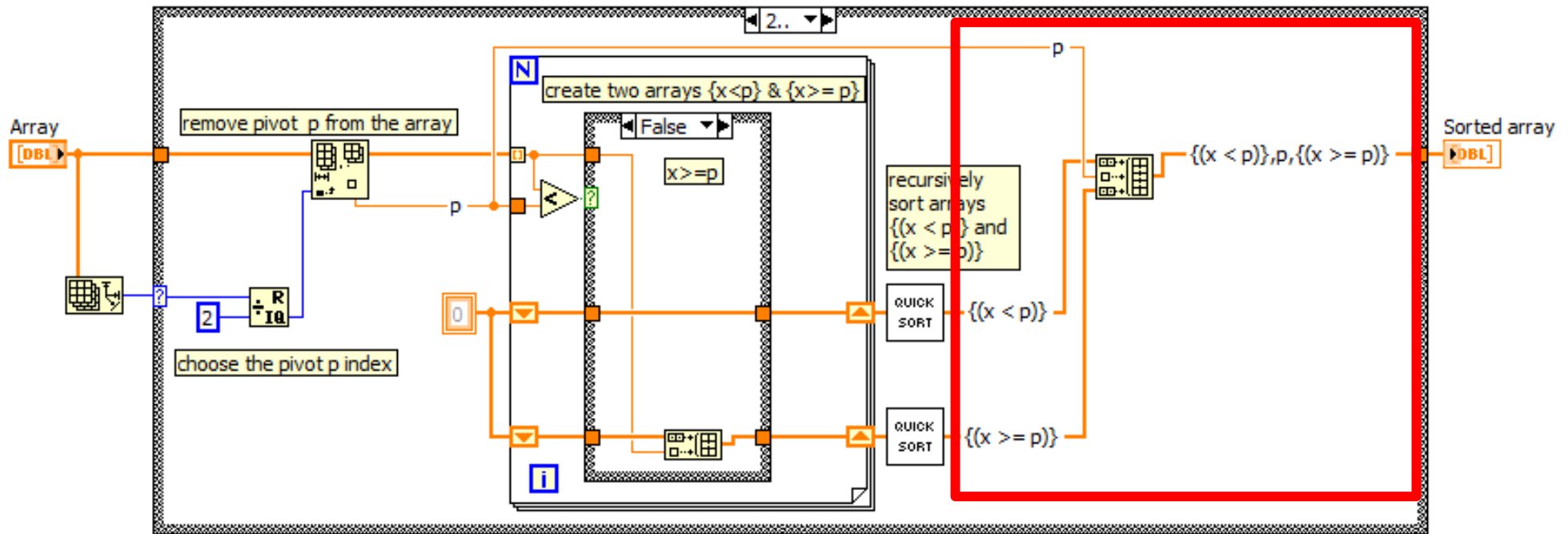


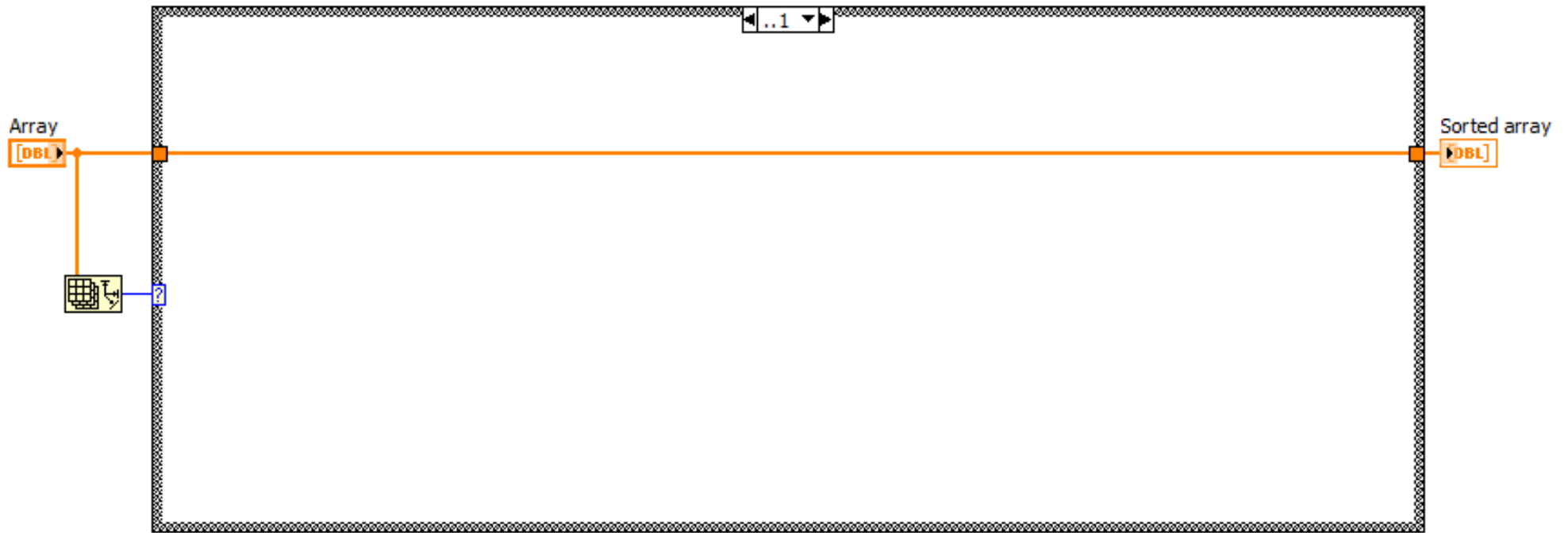
Join the pivot with the two sorted lists

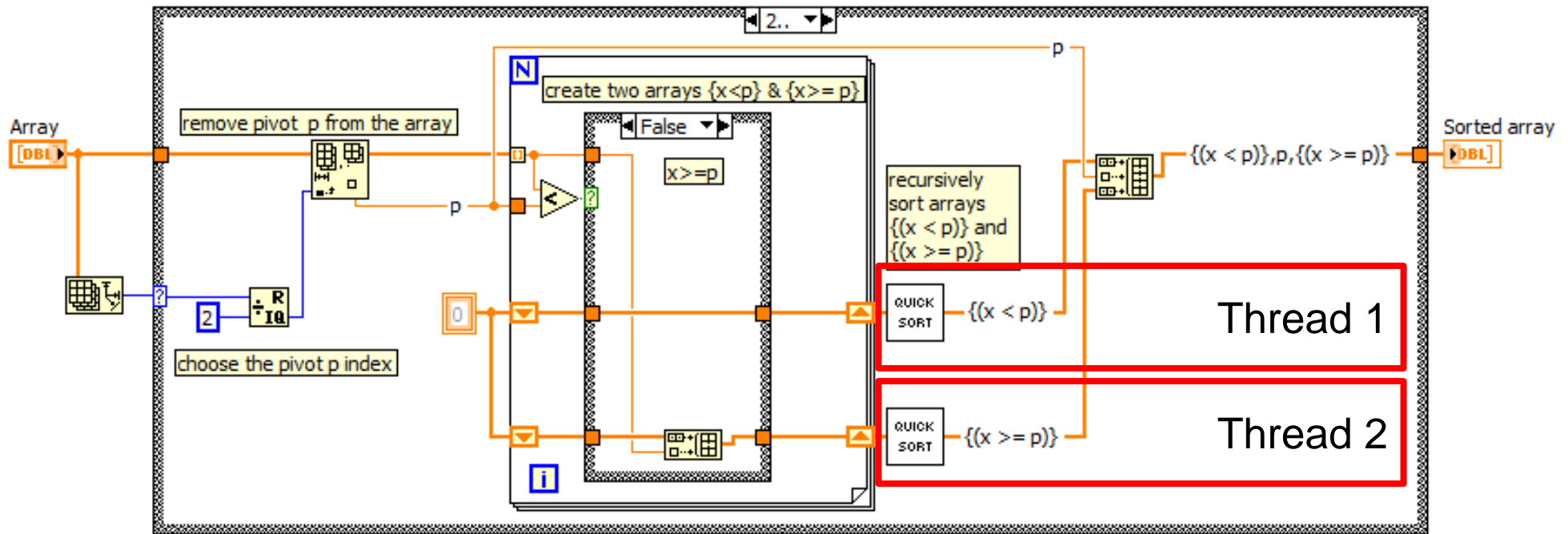


Recursion in LabVIEW

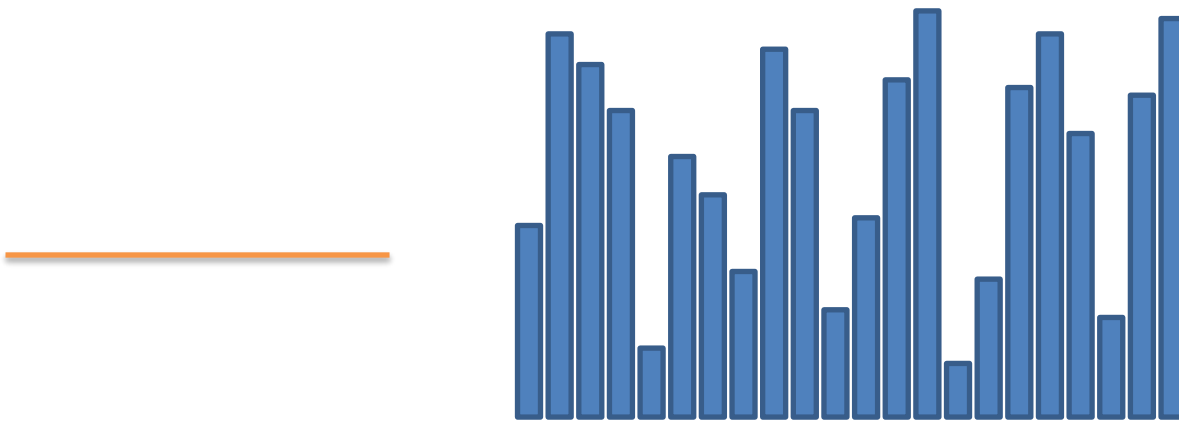
Quick sort



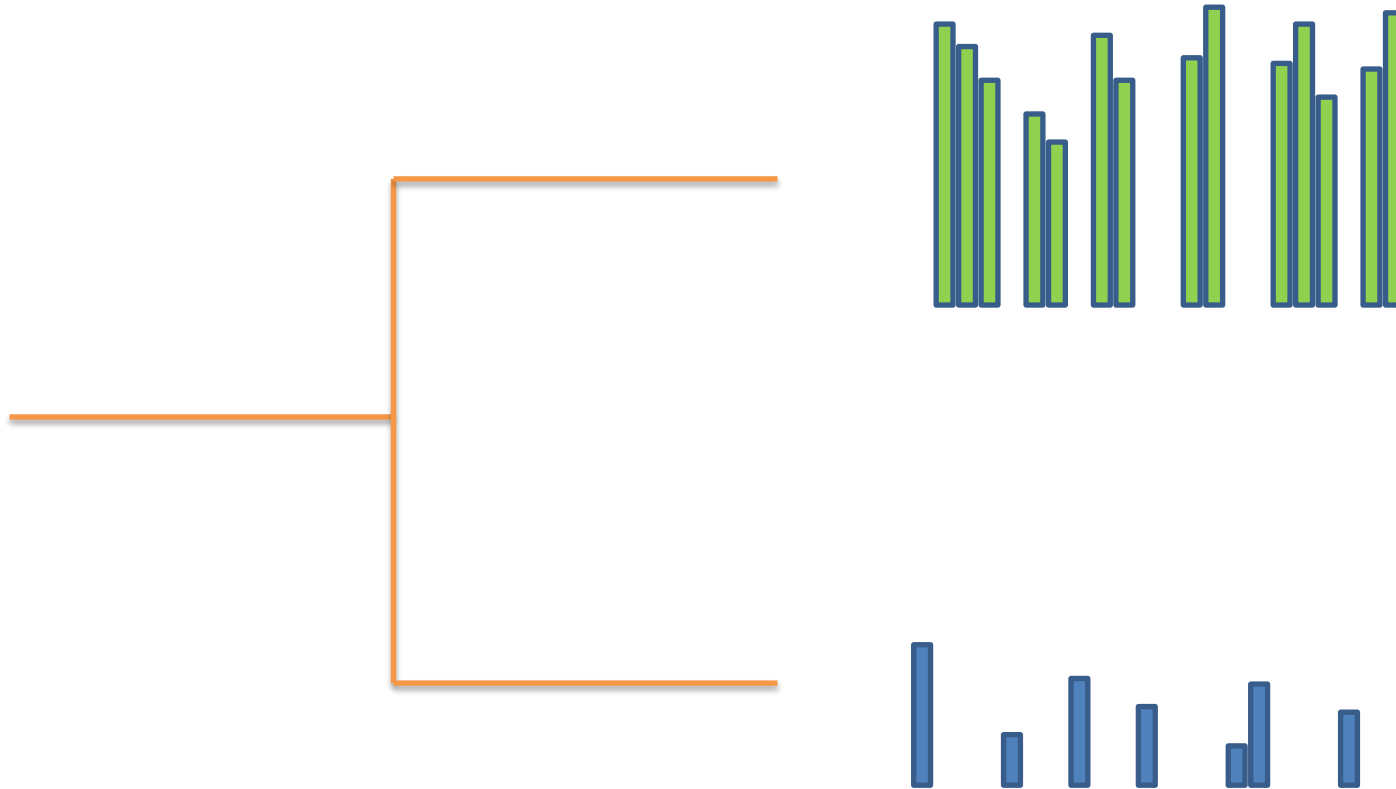




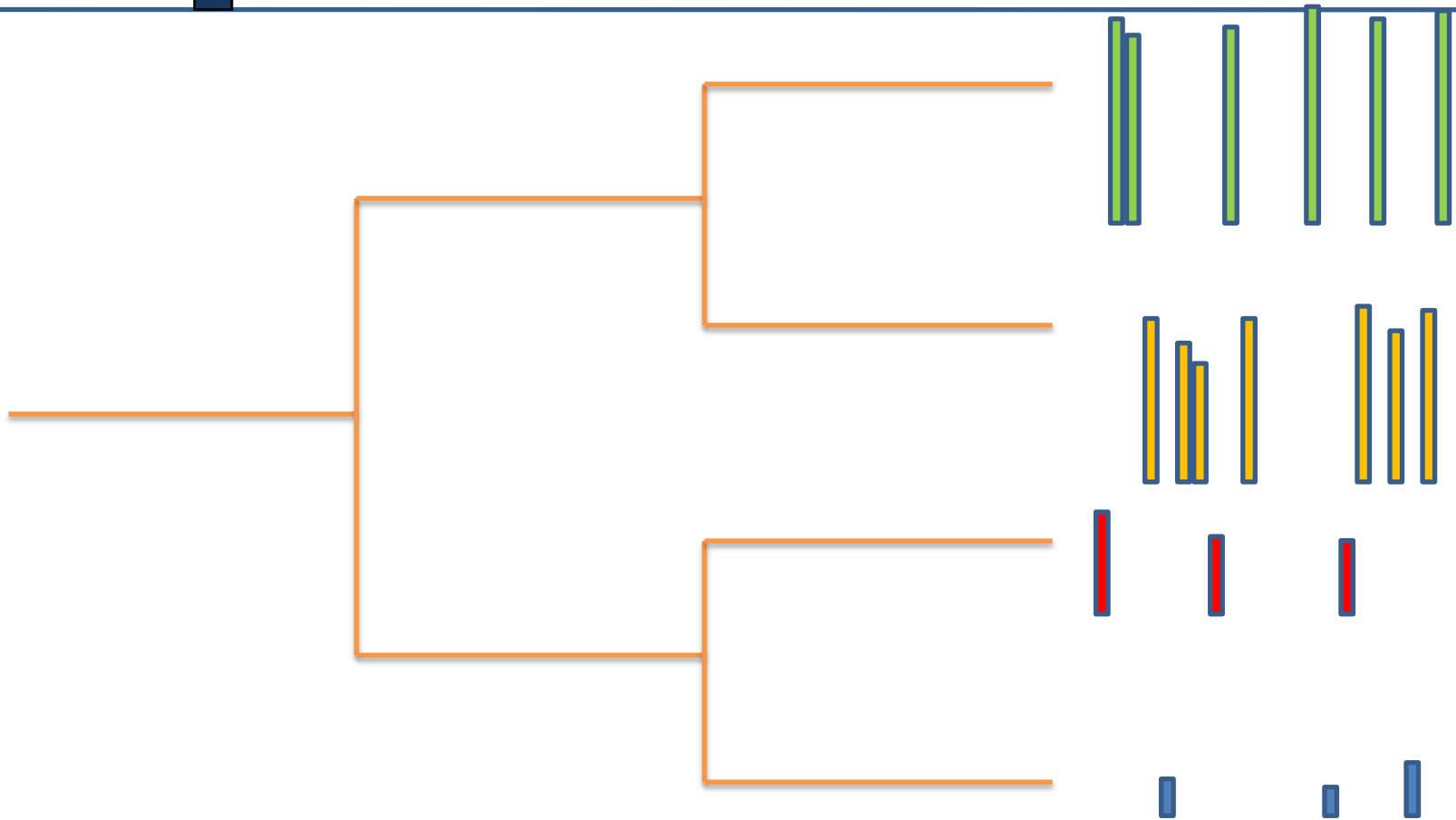
Quick sort, parallelization



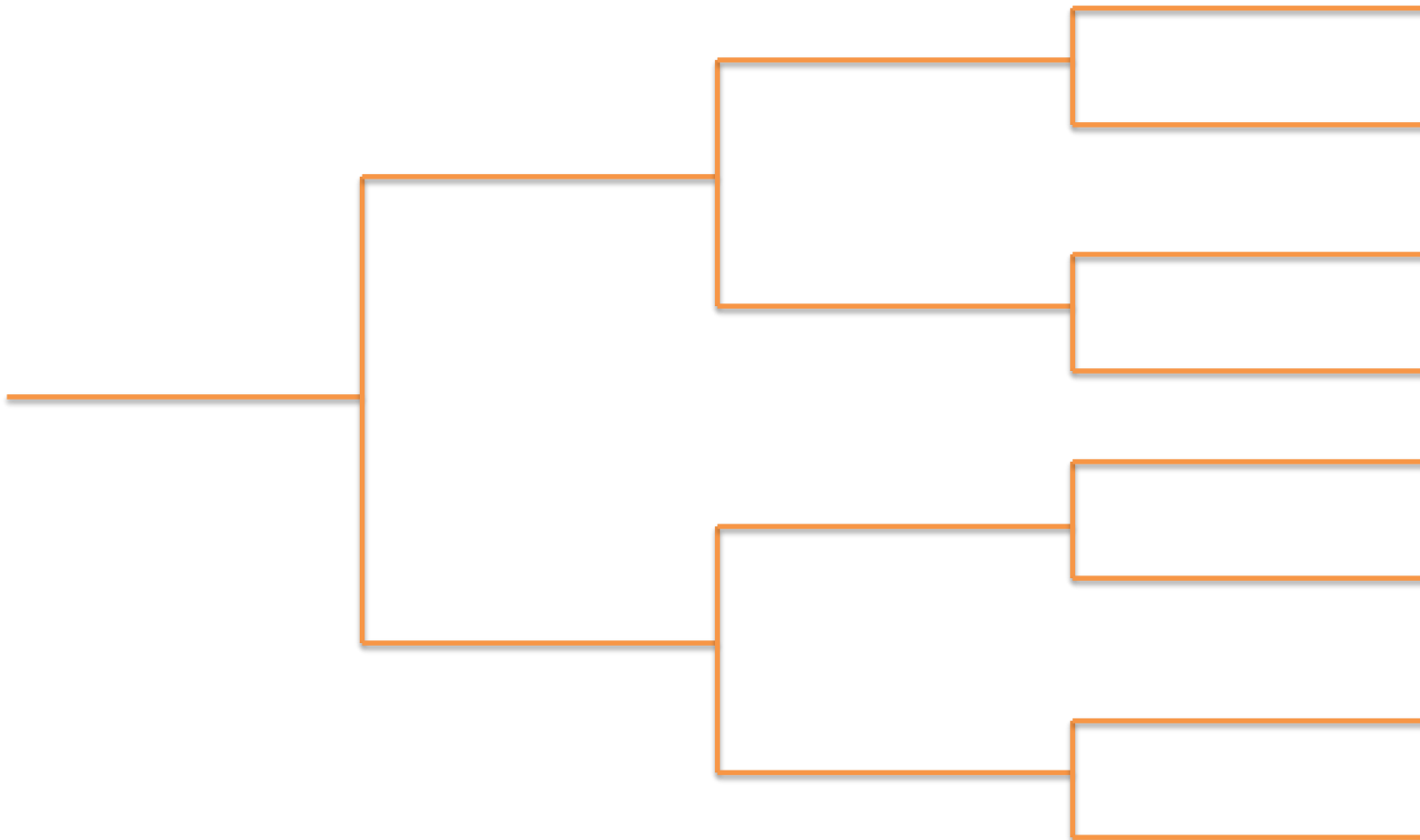
Quick sort, parallelization Level 2

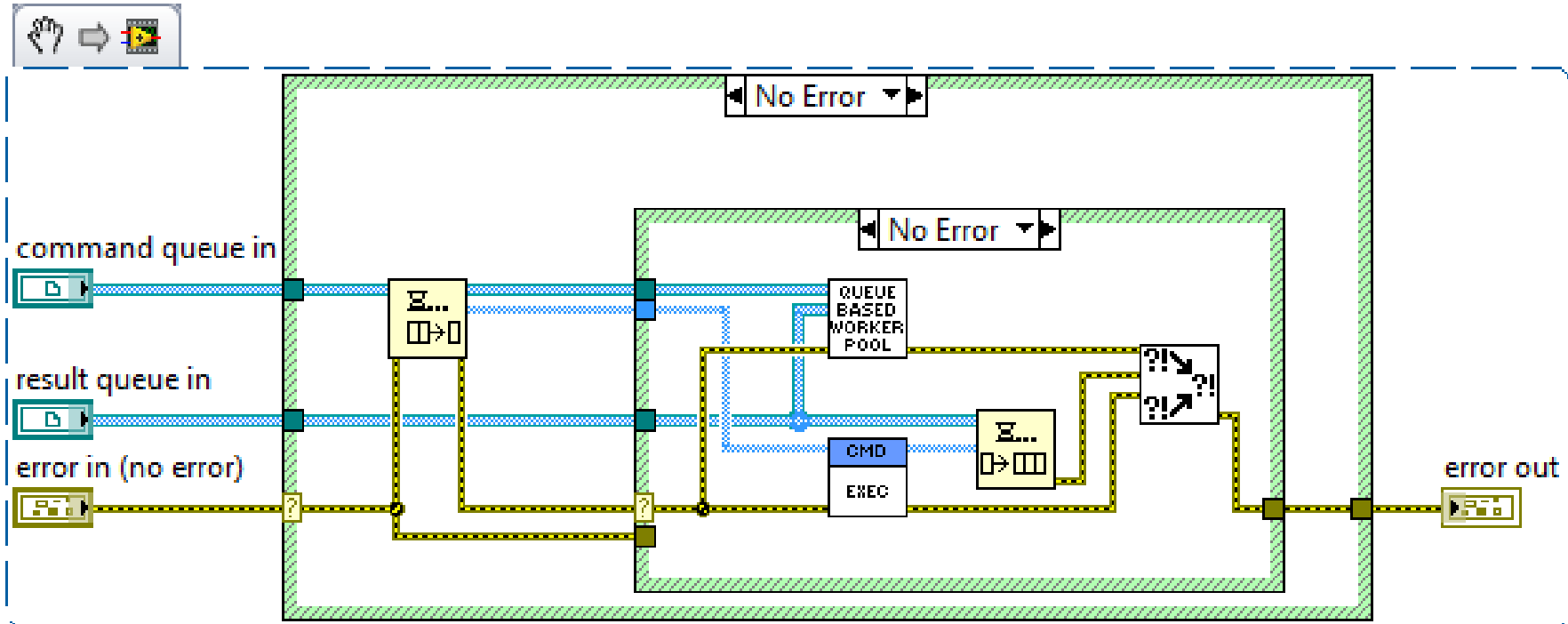


Quick sort, parallelization Level 3



Quick sort, parallelization Level 4

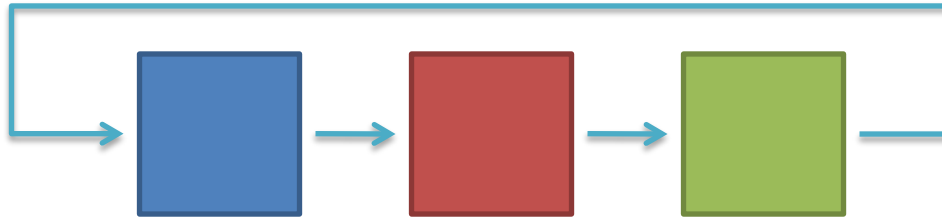




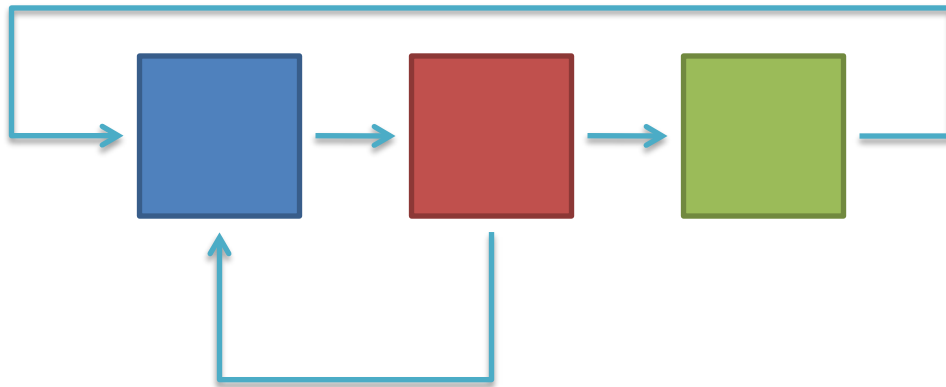
<http://expressionflow.com/2009/11/04/worker-pool/>

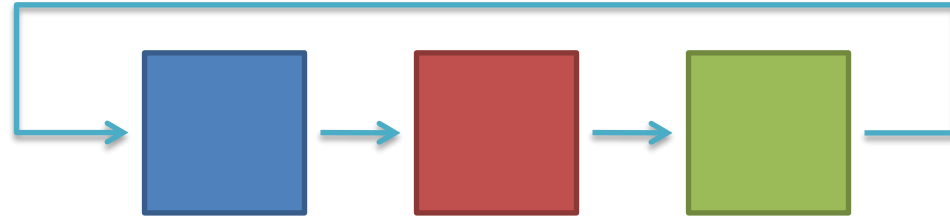


Recursive call chain

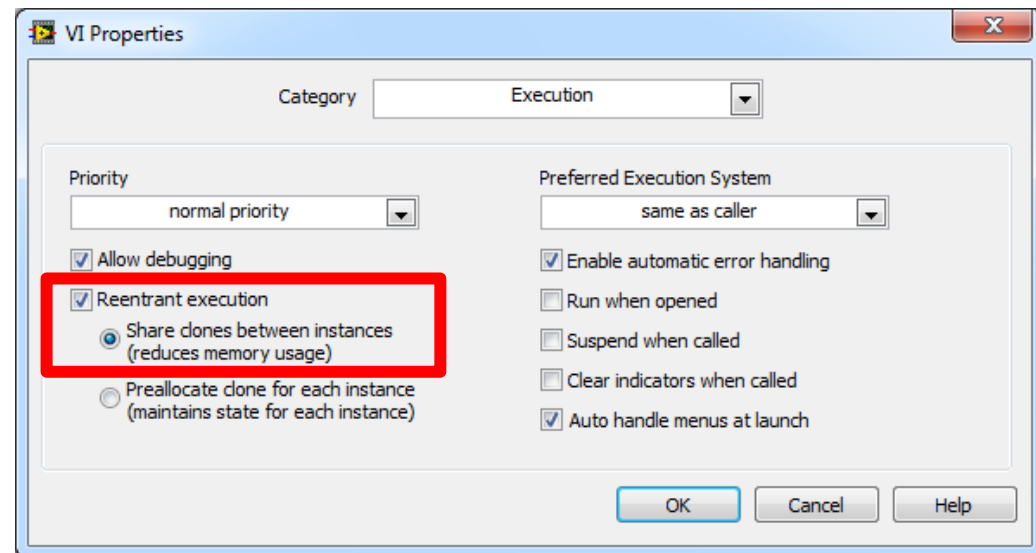


Recursive call chain





- All VIs participating recursion must be set to share clones between instances



Recursion and dynamic dispatch method VIs

- All VIs in the class hierarchy must be set to *share clones between instances*
- If you think some descendant class may need recursion, set *share clones between instances*

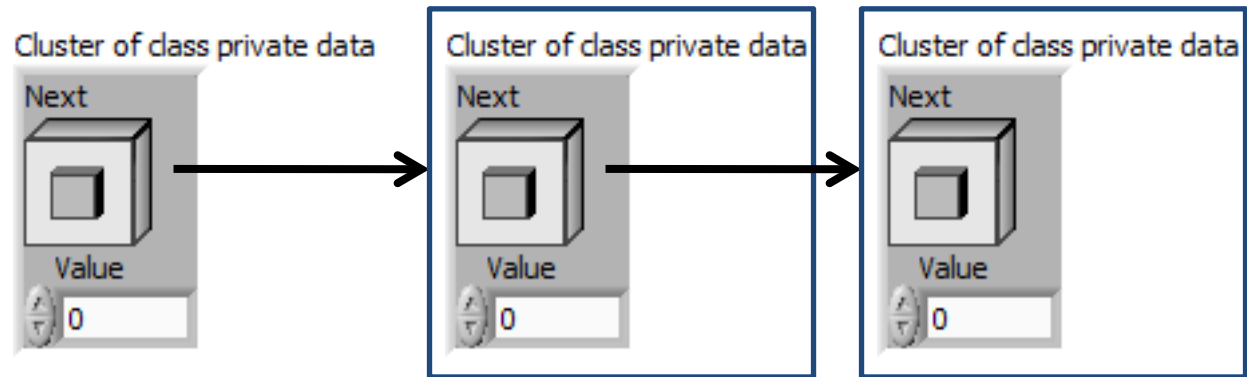


- Recursive algorithms
- Structural recursion
- ~~Type recursion~~

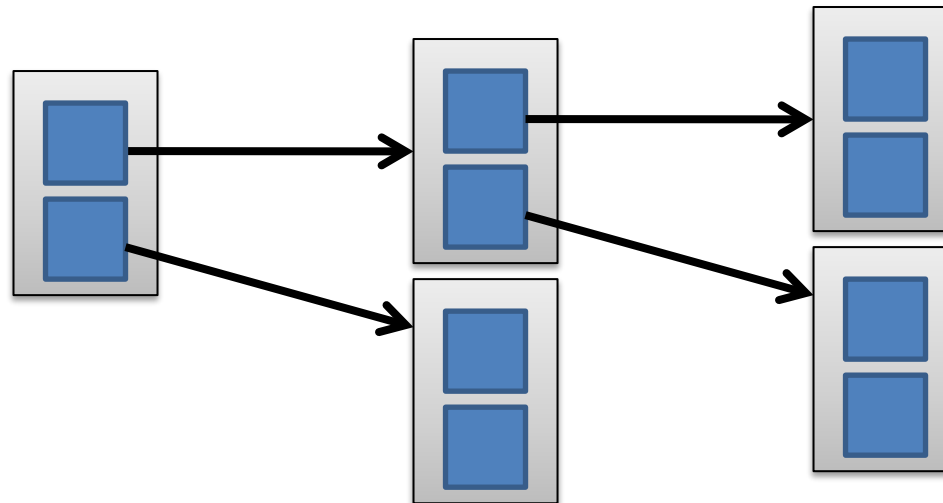


Types of recursive data structures

Lists

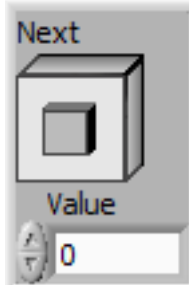


Trees



Representing recursive data structure with LV classes

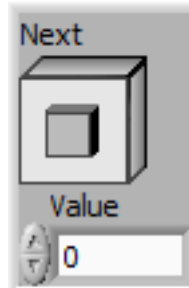
Cluster of class private data



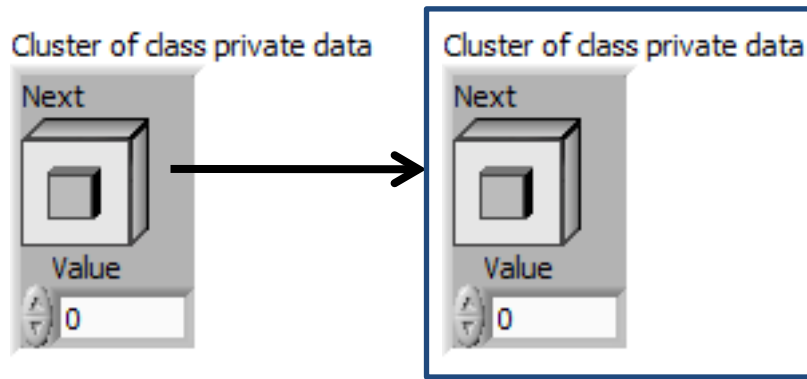
- Represented by a LabVIEW class
- Class private data contains one or more elements of more generic type such as LabVIEW object type
- At runtime, more generic type element gets a value of a class itself



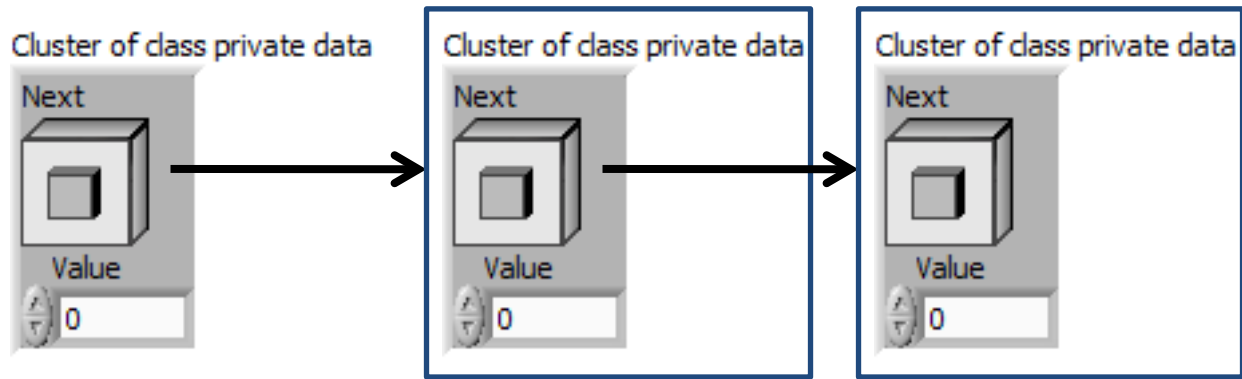
Cluster of class private data



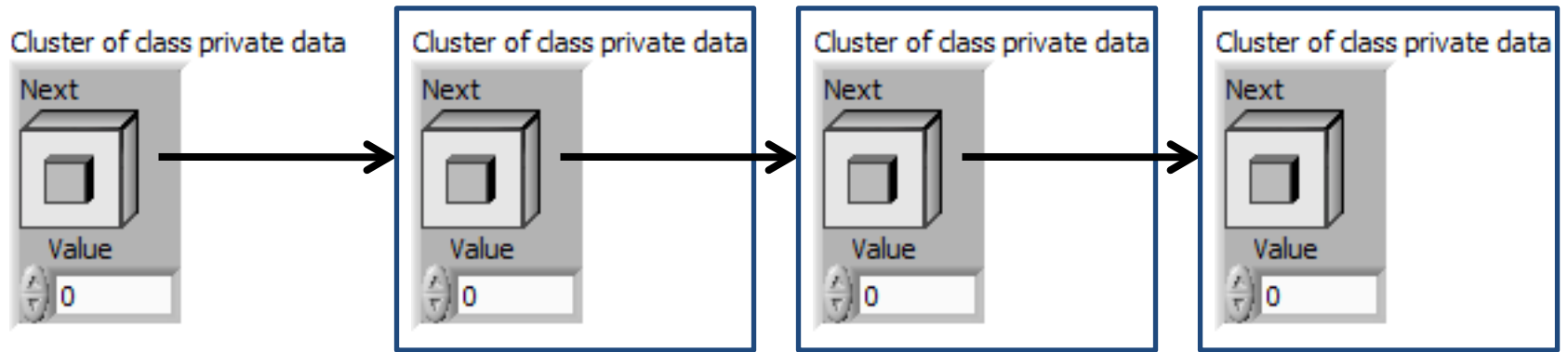
List with one element



List with two elements



List with three elements



Relation between execution and structural recursion

Structurally recursive data structures



manipulation simplest
with recursive algorithms



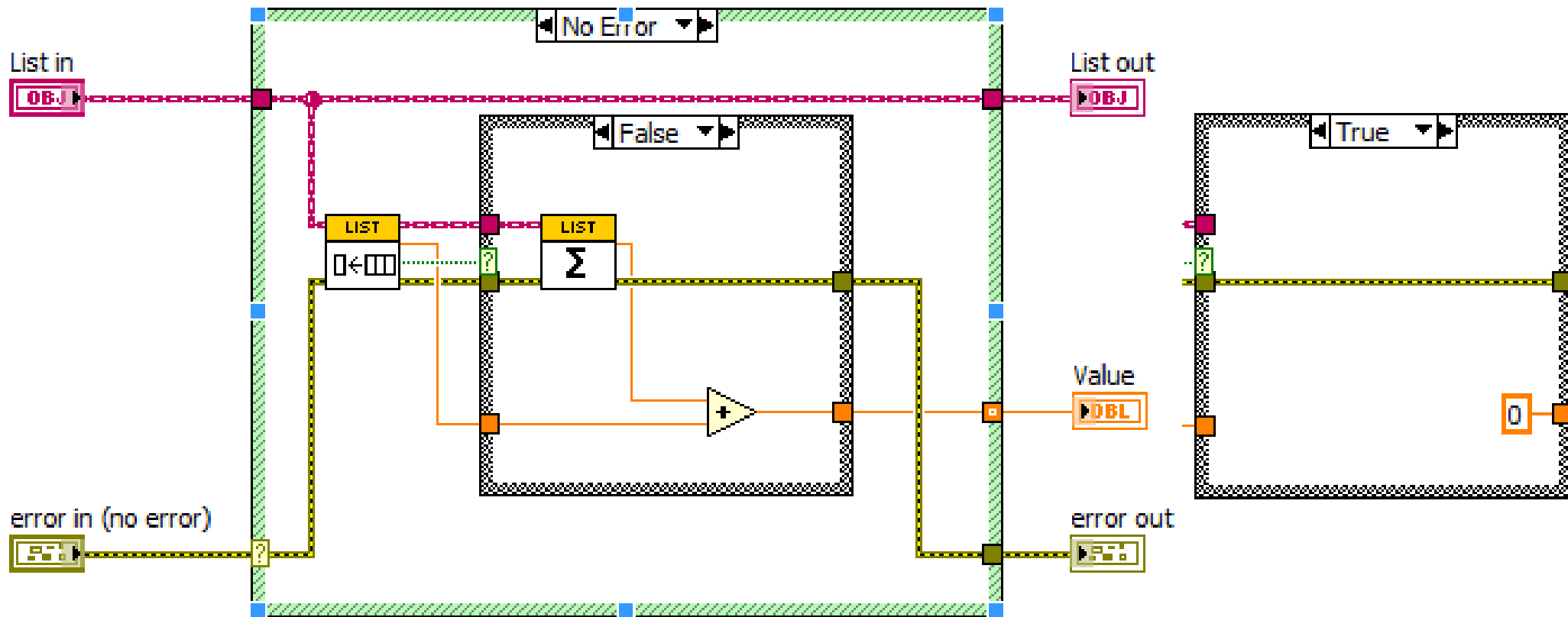
Summing list elements

- We have a list class and we want to add the elements in the list
- Simplest solution is to solve the problem recursively

Remove First.vi



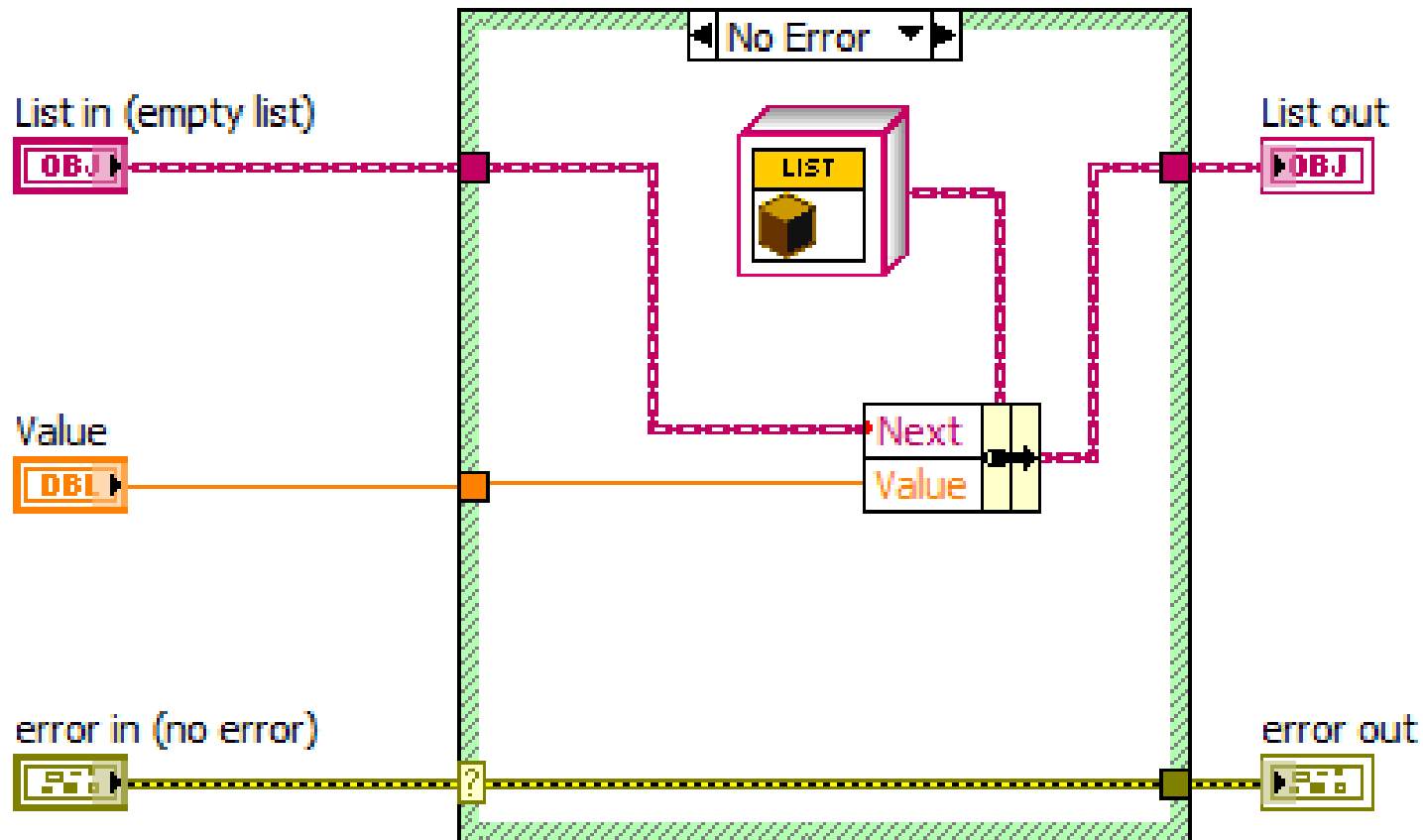
Summing list elements



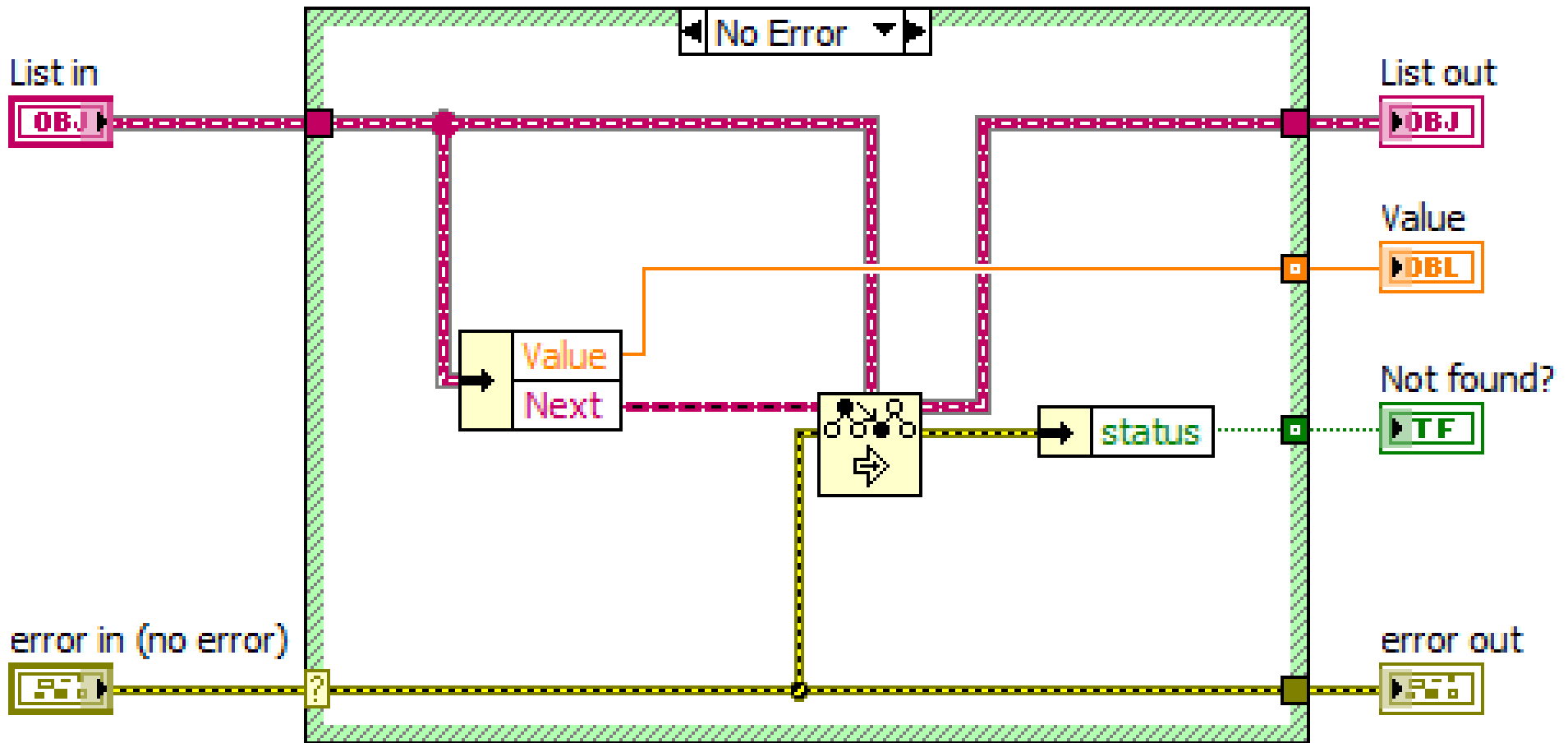
- Insert Element to the Beginning of the List
- Insert Element to the End of the List
- Remove First Element
- Remove Last Element



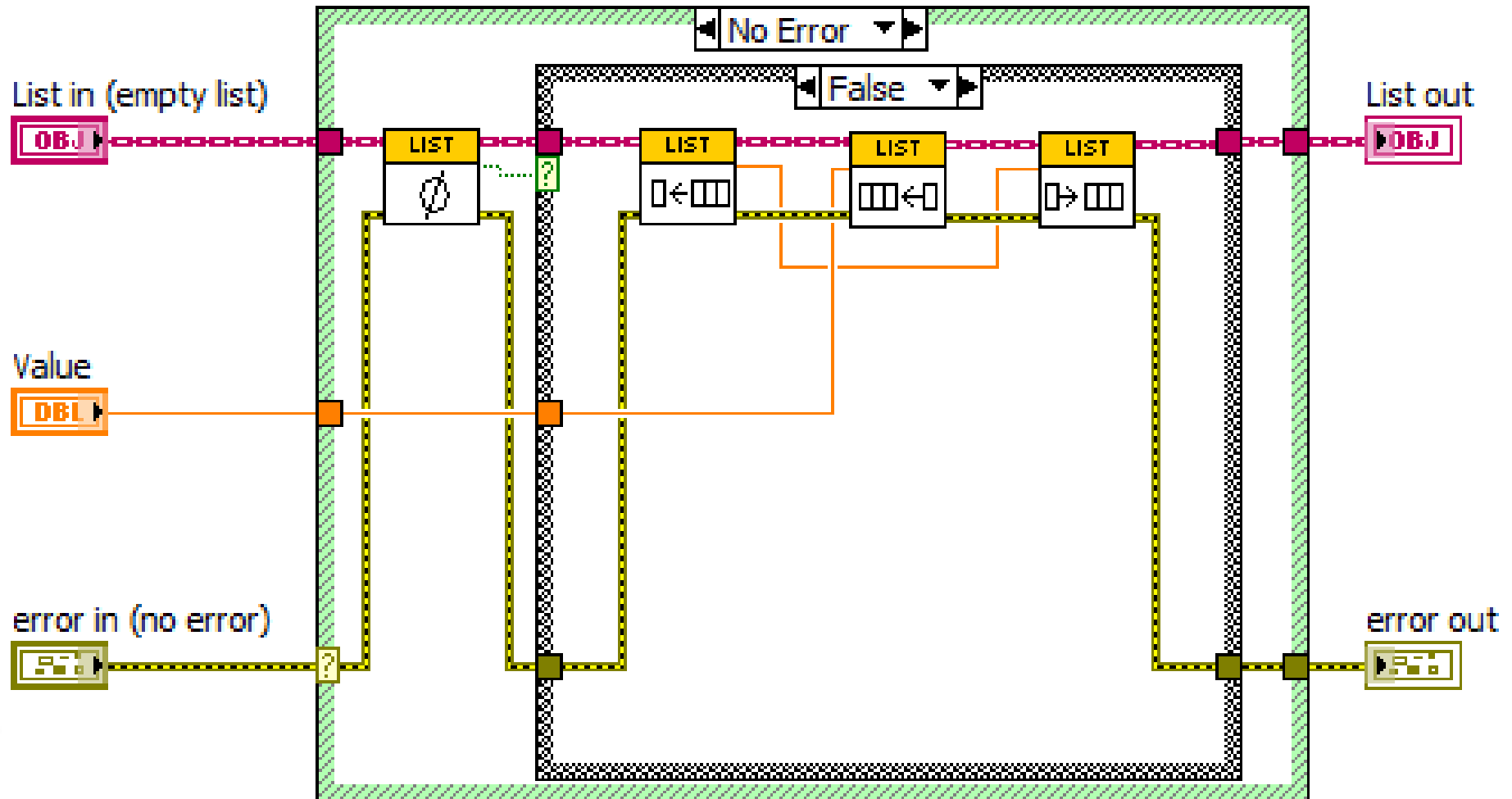
Insert element to the front of the list



Remove element from the front of the list

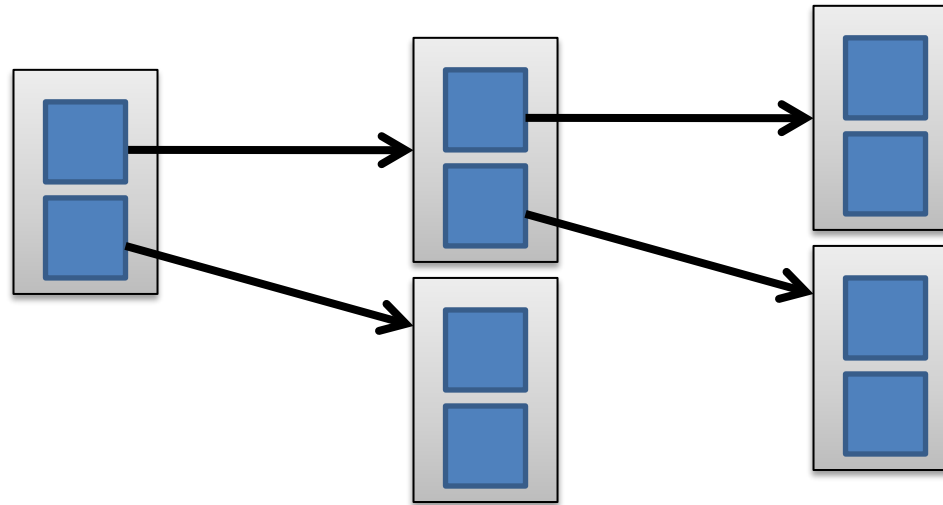


Insert element to the end of the list



Tree data structures

Trees

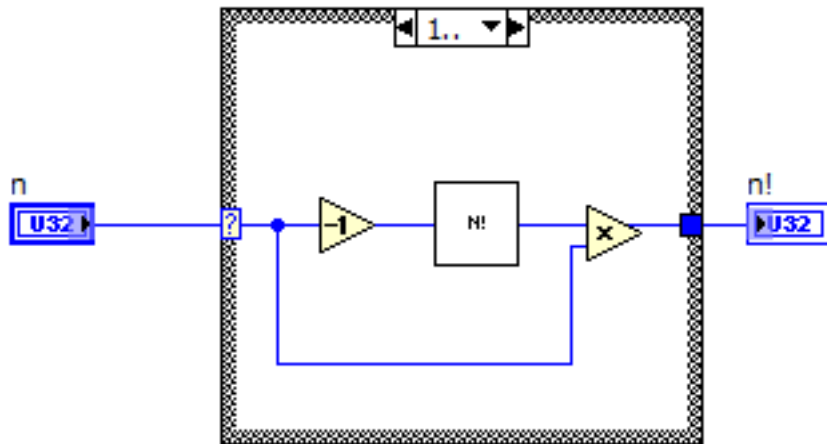
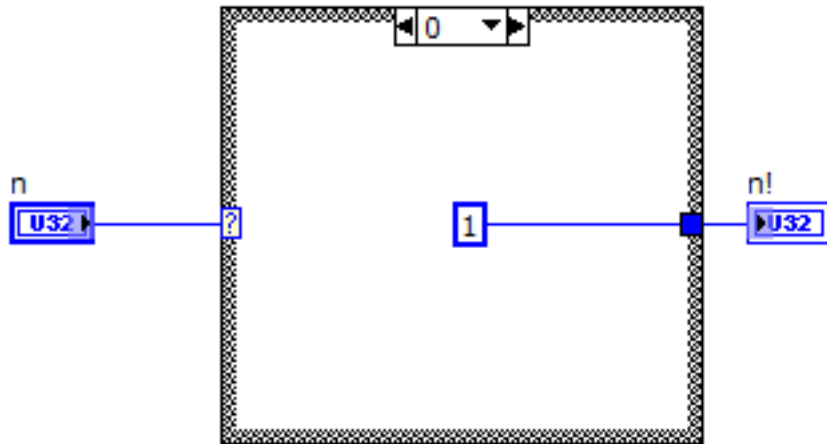


[Map, implemented with classes, for 8.5](#)

tinyurl.com/yggq993d



Tail recursion is NOT supported by LabVIEW

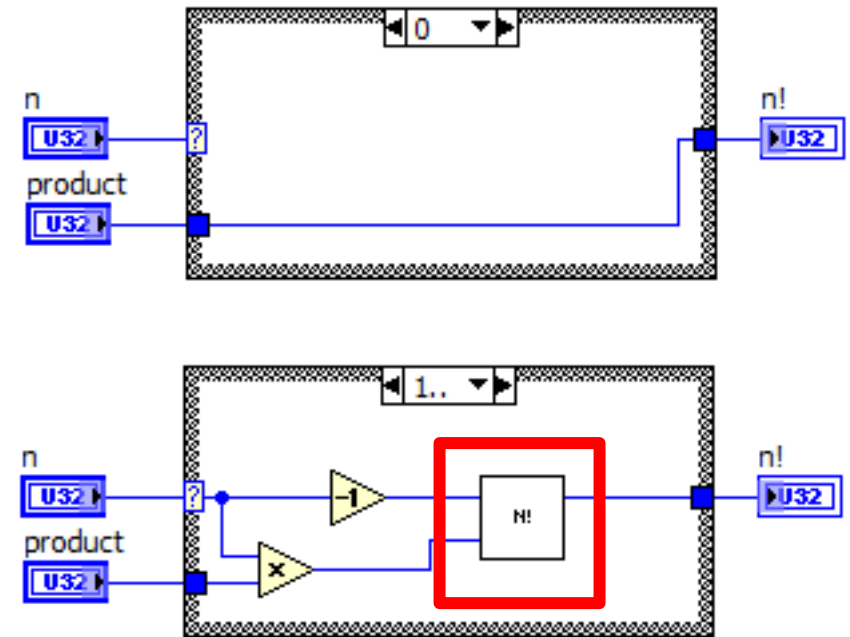
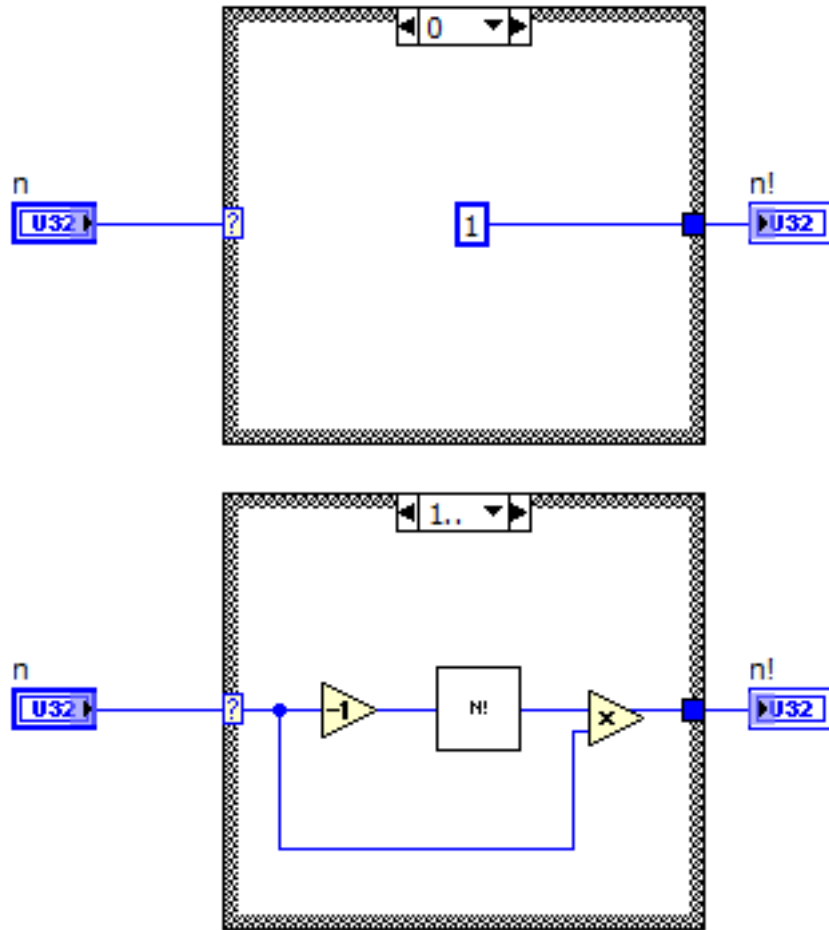


If recursive subVI call is the last operation on the block diagram, then in general, LabVIEW could reuse the caller subVI instance



Tail recursion ready code

Tail recursion is NOT supported by LabVIEW



Tomi Maila

James Kring Inc.

Email: Tomi.Maila@jameskring.com

Blog: <http://expressionflow.com>

Twitter: <http://twitter.com/expressionflow>

