MERGING TWO SORTED ARRAYS

An optimal RAM algorithm creates the merged list one element at a time.

- Requires at most n-1 comparisions to merge two sorted lists of n/2 elements.
- Time complexity $\Theta(n)$

- Can we do in lesser time?

PARALLEL MERGE

Consider two sorted lists of distinct elements of size n/2.

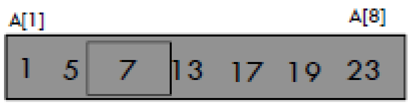
We spawn n processors, one for each element of the list to be merged.

In parallel, the processors perform binary search of the corresponding elements in the other half of the array.

- Element in the lower half of the array performs a binary search in the upper half.
- Element in the upper half of the array performs a binary search in the lower half.

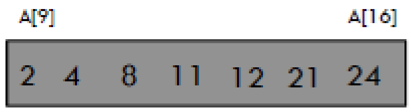
THE TASK OF P_3

A[i=3] is larger than **i-1=(3-1)=2** elements in the lower array (lower wrt. Index)



Thus, 7 is larger than 2 elements in the lower array, and larger than (**high-n/2)=10-8=2** elements in the upper array.

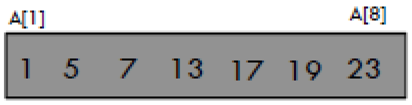
Perform a binary search with A[3] in the upper array.
Get a position high=index of the largest integer smaller than 7=>high=10.



So, P_3 can calculate the position of 7 in the merged list, ie. after (i-1)+(high-n/2), thus the position is (i+high-n/2).

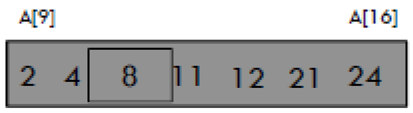
THE TASK OF P_{11}

A[i=11]=8 is larger than **i-(n/2+1)=(11-9)=2** elements in the upper array (lower wrt. Index)



Thus, 8 is larger than 2 elements in the upper array, and larger than **high=3** elements in the upper array.

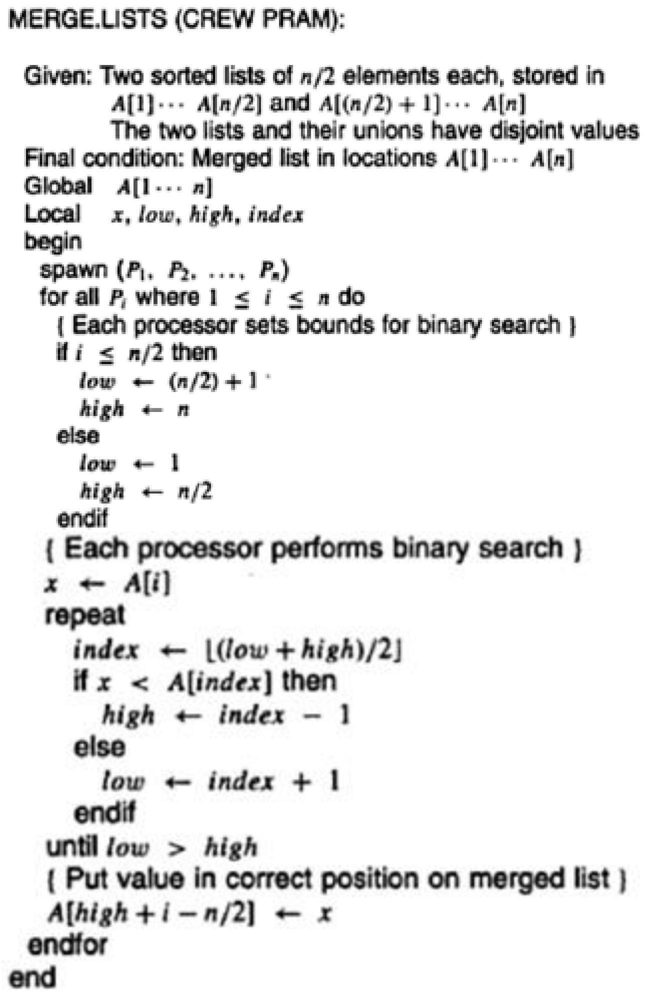
Perform a binary search with A[11] in the lower array.
Get a position high=index of the largest integer smaller than 8=>high=3.



So, P_{11} can calculate the position of 8 in the merged list, ie. after (i-n/2-1)+(high), thus the position is (i+high-n/2).

Thus the same expression is used to place the elements in their proper position in the merged list.

THE PRAM ALGORITHM



MERGE.LISTS (CREW PRAM):

```
 Given: Two sorted lists of $n/2$ elements each, stored in
          $A[1] \cdots A[n/2]$ and $A[(n/2)+1] \cdots A[n]$
          The two lists and their unions have disjoint values
 Final condition: Merged list in locations $A[1] \cdots A[n]$
 Global   $A[1 \cdots n]$
 Local    $x, low, high, index$
 begin
   spawn ($P_1, P_2, \ldots, P_n$)
   for all $P_i$ where $1 \le i \le n$ do
     { Each processor sets bounds for binary search }
     if $i \le n/2$ then
       $low \leftarrow (n/2)+1$
       $high \leftarrow n$
     else
       $low \leftarrow 1$
       $high \leftarrow n/2$
     endif
     { Each processor performs binary search }
     $x \leftarrow A[i]$
     repeat
       $index \leftarrow \lfloor (low + high)/2 \rfloor$
       if $x < A[index]$ then
         $high \leftarrow index - 1$
       else
         $low \leftarrow index + 1$
       endif
     until $low > high$
     { Put value in correct position on merged list }
     $A[high + i - n/2] \leftarrow x$
   endfor
 end
```

Note that the final writing into the array is done by the processors without any conflict. All the locations are distinct.
Also note that the total number of operations performed have increased from that in a sequential algorithm $\Theta(n)$ to $\Theta(nlogn)$ in the parallel algorithm.