```
P-MERGE($A, p, q, r$)
    let $B[p:r]$ be a new array                    // allocate scratch array
    P-MERGE-AUX($A, p, q, q+1, r, B, p$)           // merge from $A$ into $B$
    parallel for $i = p$ to $r$                     // copy $B$ back to $A$ in parallel
        $A[i] = B[i]$

P-MERGE-AUX($A, p_1, r_1, p_2, r_2, B, p_3$)
    if $p_1 > r_1$ and $p_2 > r_2$                  // are both subarrays empty?
        return
    if $r_1 - p_1 < r_2 - p_2$                      // second subarray bigger?
        exchange $p_1$ with $p_2$                   // swap subarray roles
        exchange $r_1$ with $r_2$
    $q_1 = \lfloor (p_1 + r_1)/2 \rfloor$           // midpoint of $A[p_1 : r_1]$
    $x = A[q_1]$                                    // median of $A[p_1 : r_1]$ is pivot $x$
    $q_2$ = FIND-SPLIT-POINT($A, p_2, r_2, x$)     // split $A[p_2 : r_2]$ around $x$
    $q_3 = p_3 + (q_1 - p_1) + (q_2 - p_2)$         // where $x$ belongs in $B$ ...
    $B[q_3] = x$                                    // ... put it there
    // Recursively merge $A[p_1 : q_1 - 1]$ and $A[p_2 : q_2 - 1]$ into $B[p_3 : q_3 - 1]$.
    spawn P-MERGE-AUX($A, p_1, q_1 - 1, p_2, q_2 - 1, B, p_3$)
    // Recursively merge $A[q_1 + 1 : r_1]$ and $A[q_2 : r_2]$ into $B[q_3 + 1 : r_3]$.
    spawn P-MERGE-AUX($A, q_1 + 1, r_1, q_2, r_2, B, q_3 + 1$)
    sync                                            // wait for spawns
```