Par-Randomized-CC-2(V, E, L)

(Input is an unweighted undirected graph with vertex set V and edge set E. For each $v \in V$, $L[v]$ is set to v before the invocation of this function. When this function terminates, for each $v \in V$, $L[v]$ contains the unique id of the connected component containing v. We call an edge (u, v) *live* provided $L[u] \neq L[v]$.)

if $|E| = 0$ ***then return*** *{no edge to contract}*
parallel for each $(u, v) \in E$ ***do*** $N[u] \leftarrow v$, $N[v] \leftarrow u$ *{try to associate the edge (u, v) with u and v}*
Random-Hook(V, L, N) *{hook among vertices in V based on the edges chosen in the previous step}*
$V' \leftarrow \{ v \mid v \in V \wedge v = L[v] \}$ *{V' contains only the roots after hooking}*
$E' \leftarrow \{ (L[u], L[v]) \mid (u, v) \in E \wedge L[u] \neq L[v] \}$ *{E' contains only edges among roots, and no duplicate edges and self loops}*
Par-Randomized-CC-2(V', E', L) *{recurse on the smaller instance}*
parallel for each $v \in V$ ***do*** $L[v] \leftarrow L[L[v]]$ *{map the solution back to the current instance}*

Random-Hook(V, L, N)

(Input is an unweighted undirected graph with vertex set V. For each $v \in V$, $L[v]$ is set to v before the invocation of this function. For each $u \in V$, $N[u]$ is set to a v such that (u, v) is an edge in the graph. This function randomly hooks vertices in V to their neighbors in such a way that after the function terminates these vertices form a set of disjoint stars. For each $v \in V$, $L[v]$ is set to u (possibly $u = v$) provided u is the center of the star containing v.)

parallel for each $u \in V$ ***do*** *{for each vertex in V}*
    $C_u \leftarrow$ Random{ Head, Tail } *{toss a coin}*
    $H_u \leftarrow$ False *{record that this vertex has not yet been hooked}*
parallel for each $u \in V$ ***do*** *{for each vertex u in V}*
    $v \leftarrow N[u]$ *{will try to hook u with $v = N[u]$}*
    if $C_u =$ Tail ***and*** $C_v =$ Head ***then*** *{if u tossed Tail and v tossed Head}*
        $L[u] \leftarrow v$ *{make u point to v}*
        $H_u \leftarrow$ True, $H_v \leftarrow$ True *{record that both u and v are hooked}*
parallel for each $u \in V$ ***do*** *{manipulate the coin tosses to hook more in a second try}*
    if $H_u =$ True ***then*** $C_u \leftarrow$ Head *{if u is already hooked, will try to hook unhooked vertices pointing to u}*
    else if $C_u =$ Tail ***then*** $C_u \leftarrow$ Head ***else*** $C_u \leftarrow$ Tail *{if u is not hooked, flip C_u}*
parallel for each $u \in V$ ***do*** *{try to hook again}*
    $v \leftarrow N[u]$ *{will try to hook u with $v = N[u]$}*
    if $C_u =$ Tail ***and*** $C_v =$ Head ***then*** *{if u has Tail and v has Head}*
        $L[u] \leftarrow L[v]$ *{make u point to whatever v is pointing to}*

Figure 1: Parallel connected components (CC) on a graph.

Par-Randomized-CC-3($V,\ E,\ L,\ PhD,\ N,\ U,\ d$)

(Input is an unweighted undirected graph with vertex set V and edge set E. The recursion depth of the function is given by d which is set to 0 when the function is invoked for the first time. Let n be the number of vertices in the graph when $d = 0$, and $m = |E|$. Each $v \in V$ is an integer in $[1, n]$. Pointers L (label), PhD (potentially high degree), N (neighbor) and U (updated) point to arrays $L[1 : n]$, $PhD[1 : n]$, $N[1 : n]$ and $U[1 : n]$, respectively. For each $v \in [1, n]$, $L[v]$ is set to v, and $PhD[v]$ is set to True before the initial invocation of this function. When this function terminates, for each $v \in V$, $L[v]$ contains the unique id of the connected component containing v. We assume that $\alpha = \sqrt{\frac{15}{16}}$ and $d_{max} = \left\lceil \frac{1}{4} \log_{\frac{1}{\alpha}} n \right\rceil$. We call an edge (u, v) *live* provided $L[u] \neq L[v]$. Edge (u, v) is *heavy* provided $PhD[u] = PhD[v]$, otherwise it is *light*.)

if $d \leq d_{max}$ ***then*** $\{$*need to recurse more to sufficiently reduce #vertices with PhD status*$\}$
    $m_d \leftarrow \lceil m \cdot \alpha^d \rceil$ $\{$*size of edge sample which geometrically decreases with d*$\}$
    $\widehat{E} \leftarrow$ a sample of size m_d chosen uniformly at random from E $\{$*do not always touch all edges in E*$\}$
    parallel for each $v \in V$ ***do*** $U[v] \leftarrow$ False $\left\{\text{\textit{flag } } U[v] \text{ \textit{keeps track if an edge in} } \widehat{E} \text{ \textit{hits} } v\right\}$
    parallel for each $(u, v) \in \widehat{E}$ ***do*** $\{$*check each edge in the sample*$\}$
        $u' \leftarrow L[u],\ v' \leftarrow L[v]$ $\{$*find the root of the tree containing each endpoint*$\}$
        if $u' \neq v'$ ***and*** $PhD[u'] = PhD[v'] =$ True ***then*** $\{$*if the edge* (u', v') *is live and heavy*$\}$
            $N[u'] \leftarrow v',\ N[v'] \leftarrow u'$ $\{$*try to associate the edge with* u' *and* $v'$$\}$
            $U[u'] \leftarrow$ True, $U[v'] \leftarrow$ True $\left\{\widehat{E} \text{ \textit{hits} } u' \text{ \textit{and} } v'\right\}$
    parallel for each $v \in V$ ***do*** $\{$*check each vertex v in V*$\}$
        if $U[v] =$ false ***then*** $PhD[v] \leftarrow$ False $\left\{\text{\textit{if} } \widehat{E} \text{ \textit{does not hit} } v \text{ \textit{then} } v \text{ \textit{loses its PhD status}}\right\}$
    $\widehat{V} \leftarrow \{\ v \mid v \in V\ \wedge\ U[v] = \text{True}\ \}$ $\left\{\widehat{V} \text{ \textit{contains the vertices from} } V \text{ \textit{which still have PhD status}}\right\}$
    Random-Hook($\widehat{V},\ L,\ N$) $\left\{\text{\textit{hook among vertices in} } \widehat{V} \text{ \textit{(see Figure 1)}}\right\}$
    $V' \leftarrow \{\ v \mid v \in V\ \wedge\ v = L[v]\ \}$ $\{$*V' contains only the roots after hooking*$\}$
    Par-Randomized-CC-3($V',\ E,\ L,\ PhD,\ N,\ U,\ d+1$) $\{$*recurse on the smaller instance*$\}$
    parallel for each $v \in V$ ***do*** $L[v] \leftarrow L[L[v]]$ $\{$*map the solution back to the current instance*$\}$
else
    $E' \leftarrow \{\ (L[u], L[v]) \mid (u, v) \in E\ \wedge\ L[u] \neq L[v]\ \}$ $\{$*E' contains only edges among roots, and no duplicate edges and self loops*$\}$
    Par-Randomized-CC-2($V,\ E',\ L$) $\{$*use the algorithm from Figure 1 to solve the problem once the number of edges reduces to a sufficiently small number*$\}$

Figure 2: Parallel connected components (CC) based on edge sampling.