<< Work-flow for enumerating the parsimonious indel histories
& calculating 1st-approximate multiplication factors for indel evolutionary model >>

[INPUT DATA]

MSA: (\@seqnames, \@in_msa)

NOTE: The input MSA must be pre-processed, e.g., via the script,
"**preprocess_msa_dawg.pl**" in "Sample_Scripts/" archived in "FA_LOLIPOG_P.verxxx.tgz",
so that each gapped segment will be replaced by a representative segment
with the identical homology structure;

Tree: (\%node2ch, \%node2pa, \%name2id);
Auxiliary information: (\%node2depth, \%node2ct_exoffs, $char_abs, $char_pres);

Reservoir of (column-wise) gap-pattern information known in advance:
@info_gpatterns0 = (\@set_gpatterns0, \@set_node2dparsstat0, \@set_br2change0)
gives information on the gap-patterns found in the reference & reconstructed MSAs, and the Dollo parsimonious scenarios yielding these gap-patterns, where

${$set_gpatterns0[$indx_gp]} is the string representing the $ndx_gp th gap-pattern in the input MSA, created by concatenating $CHAR_PRES/ABS in the sequences;

%{$set_node2dparsstat0[$indx_gp]} = ($node_id => [$state, $ct_pres_ext], ...) is for the Dollo parsimonious scenario yielding the $indx_gp th gap-pattern;

%{$set_br2change0[$indx_gp]} = ($branch_id => [$state_ue, $state_le], ...) for the Dollo parsimonious scenario yielding the $indx_gp th gap-pattern.

[STEP 1] Chop the alignment into gapped and gapless segments.

..... using "extract_cls_gblocks" in "MyTreeMap_indels_spt_odr.pm".

=> Now, the subroutine has been incorporated into the subroutine, "cal_init_psm_cands_for_sgl_msa", used in **step 2**.

[STEP 2] For each gapped segment, determine the Dollo parsimonious indel history
and ancestral gap states.

..... using "**cal_init_psm_cands_for_sgl_msa**" in "MyTreeMap_indels_spt_odr2.pm"

(or "create_initial_set_indels_spt_odr" in "MyTreeMap_indels_spt_odr.pm").

[NOTE: The "gap-pattern block" here refers to what was referred to as "gap-block" in the predecessor packages (like LOLIPOG).]

Output of "cal_init_psm_cands_for_sgl_msa": (\@info_init_psm_scn, \@info_gpatterns), where

@info_init_psm_scn = (\@set_cls_gpblocks, \@init_totcts_indels_for_cls, \@init_psm_scn_for_cls) gives information on the initial candidate of the parsimonious scenario giving rise to the reference/reconstructed MSA gap configuration, where

@{$set_cls_gpblocks[$cl]->[$gpb]} = ($indx_gp, $len_gpb, $start, $end) is for the $gpb th gap-pattern block in the $cl th cluster (i.e., gapped segment) in the input MSA, with
$indx_gp specifying the gap-pattern (stored in @{$set_gpatterns}),
$len_gpb being the length (in terms of #{columns}) of the gap-pattern block,
and $start/$end being the starting/ending coordinate of the gap-pattern block (in the 0-based full-closed convention),

$init_totcts_indels_for_cls[$cl] is the total number of insertion/deletion events in the initial candidate of the parsimonious scenario resulting in the $cl th cluster (i.e., gapped segment) in the input MSA,

@{$init_psm_scn_for_cls[$cl]} = (\%br2indels, \%node2blwise_gstat) is for the initial candidate of the parsimonious scenario resulting in the $cl th cluster (i.e., gapped segment) in the input MSA, where
%br2indels = ($branch_id => \@list_indels, ...) is the map of insertions/deletions onto branches, with
@{$list_indels[$e]} = ($state_ue, $state_le, \@indices_blk) for the $e th event (@indices_blk storing the indices of blocks involved), and where
%node2blwise_gstat = ($node_id => \@blwise_gapstats, ...) is the map of the gap states onto the nodes,
with $blwise_gapstats[$bl] is the gap-state ($CHAR_PRES/ABS) of the $bl th block at the node;;

@info_gpatterns = (\@set_gpatterns, \@set_node2dparsstat, \@set_br2change) gives information on the gap-patterns found in the input MSA, and the Dollo parsimonious scenarios yielding these gap-patterns, (on top of those initially given,) where

@set_gpatterns follows the same format as @info_gpatterns0 in the input,
@set_node2dparsstat follows the same format as @set_node2dparsstat0 in the input,
@set_br2change follows the same format as @set_br2change0 in the input.

[STEP 3] Using the Dollo parsimonious ancestral states,
vertically partition each gapped segment into **"indel blocks"**, each of which accommodates a set of effectively independent indel histories.

This step will be divided into 2 sub-steps, (A) and (B) (+ 3 additional sub-steps, (C)-(E)).
(Each of the following subroutines will handle a single gapped segment at a time.)

(A) Using the Dollo parsimonious ancestral states,

tentatively(?) partition the gapped segment into **gap-blocks** and **sequence-blocks**.

[NOTE: A "gap-block" here is NOT what was referred to as "gap-block" in the previous package (e.g., LOLIPOG),
which is now referred to as a "gap-pattern block".]

.... create a **genuinely new subroutine**, **"partit_into_gblocks_seqblocks"**, in "MyTreeMap_indels_spt_odr_finer.pm".

Output of "partit_into_gblocks_seqblocks": (\@list_gblocks, \@list_seqblocks), where

@{$list_gblocks{$gb]} = ($br, $up_or_lw, $start, $end, \@invlvd_gpbs, \@nesting) gives information on the $gb th gap-block.
$br is the ID of the branch delimiting the gap-block,
$up_or_lw = 'U'/'L' if the gap-block extends on the upper/lower side of the branch,
$start/$end is the start/end coordinate of the gap-block,
@invlvd_gpbs lists the gap-pattern blocks involved,
@nesting lists the sequence-blocks nesting the gap-block.

@{$list_seqblocks[$sb]} = ($br, $up_or_lw, $start, $end, \@invlvd_gpbs, \@nested) gives information on the $sb th sequence-block.

$br is the ID of the **primary** branch delimiting the sequence-block,
and $up_or_lw = 'U'/'L' if the sequence block extends on the upper/lower side of the branch;
$start/$end is the start/end coordinate of the sequence-block,
@invlvd_gpbs lists the gap-pattern blocks involved,
@nested lists the gap-blocks nested in the sequence-block.

(B) Using the Dollo parsimonious indel history (& its ancestral states), define "**effectively independent indel-blocks**",
each of which accommodates a set of effectively independent indel histories.

.... create a **genuinely new subroutine**, **"partit_into_eff_indep_indel_blocks"**, in "MyTreeMap_indels_spt_odr_finer.pm".

Output of "partit_into_eff_indep_indel_blocks": (\@eff_indep_indel_blocks, \@indices_incl_root), where

@{$eff_indep_indel_blocks[$eiidb]} = (\@set_delimiting_bds, $start, $end, \@union_invlvd_gpbs, \@invlvd_gbs, \@invlvd_sbs)
gives information on the $eiidb th effectively independent indel block.
@set_delimiting_bds stores information on all boundaries delimiting the indel block,
with each element being of the form [$br, $up_or_lw, $lm_gpb, $rm_gpb, \@invlvd_gpbs]
$br is the ID of the branch delimiting the indel block,
$up_or_lw = 'U'/'L' if the upper/lower-end of $br is the indel block,
$lm/rm_gpb is the leftmost/rightmost gap-pattern block involved,
@invlvd_gpbs lists the gap-pattern-blocks involved in this boundary,
$start/$end is the start/end coordinate of the boundary,
@union_invlvd_gpbs lists all the gap-pattern-blocks involved in this indel block,
@invlvd_gbs lists the gap-blocks involved, @invlvd_sbs is the sequence-blocks involved.
(NOTE ADDED: At this stage, @invlvd_gbs and @invlvd_sbs are empty.)

@indices_incl_root stores the indices of the effectively independent indel blocks that include root (top node).
It is empty if no effectively independent indel block includes root. (.... examined by a satellite subroutine, "if_idb_includes_root".)

(C) Examine what gap-blocks and sequence-blocks are involved in each of the effectively independent indel blocks.

.... create a genuinely new subroutine, **"find_invlvd_in_eff_indep_indel_blocks"**, in "MyTreeMap_indels_spt_odr_finer.pm".

Output of "find_invlvd_in_eff_indep_indel_blocks": None
This subroutine just fills in @invlvd_gbs and @invlvd_sbs in each element of @eff_indep_indel_blocks,
which is the output of "partit_into_eff_indep_indel_blocks"

(D) Using an element of the main output of "partit_into_eff_indep_indel_blocks",
create an association from each node to a gap-state corresponding to each indel block.

.... create a genuinely new subroutine, **"mk_node2blwise_gstat"**, in "MyTreeMap_indels_spt_odr_finer.pm".

Output of "mk_node2blwise_gstat": \%node2blwise_gstat, where

%node2blwise_gstat = ($node_id => \@blwise_gapstats, ...) is the map of the gap states onto the nodes,
with $blwise_gapstats[$gpb] is the gap-state ($CHAR_PRES/ABS) of the $gpb th gap-pattern block at the node.

(E) Using an element of the main output of "partit_into_eff_indep_indel_blocks",
create an association from each branch to a set of indels.

.... create a genuinely new subroutine, **"mk_br2indels"**, in "MyTreeMap_indels_spt_odr_finer.pm".

Output of "mk_br2indels": \%br2indels, where

%br2indels = ($branch_id => \@indels, ...) is the map of the indel events onto the branches,
with @{$indels[$ev]} = ($state_ue, $state_le, \@indices_blk) represents the $ev th indel event along the branch, $branch_id,
with @indices_blk storing the indices of blocks involved.

[STEP 4] Attempt to ***enumerate*** **parsimonious indel histories** within each **effectively independent indel block**.

..... create a **new subroutine**, **"enum_psm_indel_hsts_in_eff_indp_indel_blk"**, in "MyTreeMap_indels_spt_odr_finer.pm"
(by modifying "enum_psm_indel_scns_in_cls_blks" in "MyTreeMap_indels_spt_odr.pm").

Output of "enum_psm_indel_hsts_in_eff_indp_indel_blk": ($min_totct_indels_in_indel_blk, \@set_psm_hsts_in_indel_blk), where

$min_totct_indels_in_indel_blk is the total number of insertion/deletion events in the most parsimonious history(ies) **confined in the effectively independent indel block**;

@{$set_psm_hsts_in_indel_blk[$ph]} = (\%br2indels, \%node2blwise_gapstat) is for the $ph th parsimonious history **within the indel block**, where
%br2indels = ($branch_id => \@list_indels, ...) is the map of insertions/deletions onto branches, with
@{$list_indels[$id]} = ($stat_parent, $stat_node, \@indices_gpb) describing the $id th indel event at the branch,
with @indices_gpb listing the indices (in @blocks_in_cls) of the gap-pattern blocks involved in the indel event,
%node2blwise_gapstat = ($node_id => \@blwise_gapstats, ...) is the map of gap states onto the interior nodes,
If $fl_full = 0, %node2gapstat is given only at the top node.

[STEPs 1-4] all the processes will be put into a new integrated subroutine,
"enum_psm_indel_hsts_for_msa_finer", in "MyTreeMap_indels_spt_odr_finer.pm"
(created by substantially modifying "enum_psm_indel_scns_for_msa" in "MyTreeMap_indels_spt_odr.pm").

As an **INTERMEDIATE STEP**, create a new subroutine, **"enum_psm_indel_hsts_in_cls_blks_finer"**
(, by mimicking the philosophy (& function) of "enum_psm_indel_scns_in_cls_blks" in "MyTreeMap_indels_spt_odr.pm".)

Main Outputs of "enum_psm_indel_hsts_in_cls_blks_finer":
(\@min_totcts_indels_in_indel_blks, \@sets_psm_hsts_in_indel_blks, \@psm_hsts0_in_indel_blks), where

$min_totcts_indels_in_indel_blks[$eiidb] is the total number of insertion/deletion events in the most parsimonious history realizing the $eiidb th effectively independent indel block in the gapped segment in question.
@{$sets_psm_hsts_in_indel_blks[$eiidb]->[$ph]} = (\%br2indels, \%node2blwise_gapstat) is for the $ph th of the parsimonious histories realizing the $eiidb th effectively independent indel block in the gapped segment in question.
@{$psm_hsts0_in_indel_blks[$eiidb]} = (\%br2indels0, \%node2blwise_gapstat0) is for the initial candidate indel history realizing the $eiidb th effectively independent indel block in the gapped segment in question.

AND With By-Products: (\@list_gblocks, \@list_seqblocks, \@set_eff_indep_indel_blocks, \@indices_incl_root),
which are the outputs of the subroutines "partit_into_gblocks_seqblocks" (in STEP 3A) and "partit_into_eff_indep_indel_blocks" (in STEP 3B) performed on the gapped segment in question.

[INTERIM OUTPUT DATA (= Output of "enum_psm_indel_hsts_for_msa_finer")]

MAIN Outputs: (\@sets_min_totcts_indels_in_indel_blks, \@setssets_psm_hsts_in_indel_blks, \@sets_psm_hsts0_in_indel_blks), where

$sets_min_totcts_indels_in_indel_blks[$cl]->[$eiidb] is the total number of insertion/deletion events in the most parsimonious history(ies) realizing the $eiidb th effectively independent indel block in the $cl th gapped segment;

@{$setssets_psm_hsts_in_indel_blks[$cl]-->[$eiidb]->[$ph]} = (\%br2indels, \%node2blwise_gapstat) is for the $ph th of the parsimonious histories realizing the $eiidb th effectively independent indel block in the $cl th cluster, where
%br2indels = ($branch_id => \@list_indels, ...) is the map of insertions/deletions onto branches, with
@{$list_indels[$id]} = ($stat_parent, $stat_node, \@indices_gpb) describing the $id th indel event at the branch,
with @indices_gpb listing the indices (in @{$set_cls_gpblocks[$cl]}) of the gap-pattern blocks involved in the indel event,
%node2blwise_gapstat = ($node_id => \@blwise_gapstats, ...) is the map of gap states onto the interior nodes,
with $blwise_gapstats[$gpb] being $char_pres/abs for the $gpb th gap-pattern block (in the cluster) at the node $node_id;

@{$sets_psm_hsts0_in_indel_blks[$cl]->[$eiidb]} = (\%br2indels0, \%node2blwise_gstats0) is for the initial candidate indel history realizing the $eiidb th effectively indepenent indel block in the $cl th cluster.

By-Products 1: (\@set_gpatterns, \@set_node2dparsstat, \@set_br2change), which are the three components of @info_gpatterns, which in turn is a piece of the output of "cal_init_psm_cands_for_sgl_msa" (in STEP 2).

By-Products 2: (\@set_cls_gpblocks, \@info_init_psm_scn, \@set_lists_off_limits, \@set_lists_gblocks, \@set_lists_seqblocks, \@sets_eff_indep_indel_blocks, \@sets_indices_incl_root), where

@set_cls_gpblocks is a component of @info_init_psm_scn, which in turn is a piece of the output of "cal_init_psm_cands_for_sgl_msa" (in STEP 2);

@info_init_psm_scn is a piece of the output of "cal_init_psm_cands_for_sgl_msa" (in STEP 2);

@{$set_lists_off_limits[$cl]} is the output of "define_off_limits" (in STEP 3 A), performed on the $cl th gapped segment; ... PROBABLY UNNECESSARY. (Currently, @set_lists_off_limits is empty)

@{$set_lists_gblocks[$cl]} & @{$set_lists_seqblocks[$cl]} are the output of "partit_into_gblocks_seqblocks" (in STEP 3A), performed on the $cl th gapped segment;

@{$sets_eff_indep_indel_blocks[$cl]} is the output of "partit_into_eff_indep_indel_blocks" (in STEP 3B), performed on the $cl th gapped segment;

@{$sets_indices_incl_root[$cl]} is also the output of "partit_into_eff_indep_indel_blocks" (in STEP 3B), performed on the $cl th gapped segment.

[STEP 5] Using the parsimonious indel histories, ***calculate*** the first approximation
of the **multiplication factor** *contributed from* each **effectively independent indel block**.
Then, based on them, calculate the first-approximate log-likelihood of (the gap-pattern of) the input MSA.

..... create a **new subroutine**, **"apprx1_tot_lnlk_gpattern_msa2_finer"**, in "MyTreeMap_indels_ML_hs_finer.pm"
(by modifying "apprx1_tot_lnlk_gpattern_msa2" in "MyTreeMap_indels_ML_hs.pm").

[FINAL OUTPUT DATA (= Output of "apprx1_tot_lnlk_gpattern_msa2_finer")]

MAIN Outputs: ($tot_lnlk_gc_msa, $shared_fac1p, $shared_fac2p, \@set_clwise_tot_lnmfac_gc, \@set_clwise_lnmfac_gc_partit_nw, \@setsets_tot_lnmfac_gc_indel_blk), where

$tot_lnlk_gc_msa is the total log-likelihood of the gap-configuration of the input MSA in the 1st approximation
(= 'N/A' if any of the parsimonious histories contain such abnormalities that hamper the calculation of the likelihoods);

$shared_fac1p = - $tot_div_tree * ($totrate_del * $delta_del_5p + $totrate_ins);
$shared_fac2p = - ($totrate_ins+$totrate_del) * $tot_div_tree * $totlen_rootseq0;

$set_clwise_tot_lnmfac_gc[$i] is the total log-multiplication factor from the gap configuration in the ($i+1) th gapped segment,
which is $set_clwise_lnmfac_gc_partit_nw[$i] + Sum_{$eiidb} $setsets_tot_lnmfac_gc_indel_blk[$i]->[$eiidb]
(= 'N/A' if any of the parsimonious histories in the cluster contain such abnormalities that hamper the calculation of the multiplication factors);

$set_clwise_lnmfac_gc_partit_nw[$i] is the log-multiplication factor from the gap-configuration of the "partitioning" network (, which is determined exclusively from the Dollo parsiomonious ancestral gap-states,) of the ($i+1) th gapped segment;

$setsets_tot_lnmfac_gc_indel_blk[$i]->[$eiidb] is the total log-multiplication factor from the gap configuration in the $eiidb th effectively independent indel block of the ($i+1) th gapped segment,
which is log (Sum_{$hst} exp ($setsetsets_lnmfac_gc_indel_blk_hst[$i]->[$eiidb]->[$hst]))
(= 'N/A' if any of the parsimonious histories in the cluster contain such abnormalities that hamper the calculation of the multiplication factors);

AUXILIARY Outputs: ($tot_div_tree, $totlen_ng, $totlen_rootseq0, \@lens_rootseq0_in_cls, \@set_node2len_ancseq0_in_cls, \@setsetsets_lnmfac_gc_indel_blk_hst), where

$tot_div_tree is the total divergence of the tree, which is the summation of the branch lengths over branches in the entire tree;

$totlen_ng is the total number of gapless columns (i.e., the total length over the gapless segments);

$totlen_rootseq0 is the total length of the **"reference" root sequence** across the MSA.
As a "reference" root sequence, this subroutine uses the root sequence under the **Dollo parsimonious indel history**;

$lens_rootseq0_in_cls[$i] is the length of the "reference" root sequence in the ($i+1) th gapped segment;

%{$set_node2len_ancseq0_in_cls[$i]} = ($node_id => $len_ancseq0_in_cls, ...) gives the lengths of the **"reference" ancestral states** at node $node_id in the ($i+1) th gapped segment.
As a "reference" set of ancestral states, this subroutine uses the ancestral states in the **Dollo parsimonious indel history**;

$setsetsets_lnmfac_gc_indel_blk_hst[$i]->[$eiidb]->[$hst] is the log-multiplication factor from the $hst th indel history
that parsimoniously give the $eiidb th (effectively independent) indel block in the ($i+1) th gapped segment
(= 'N/A' if the parsimonious history contains such abnormalities that hamper the calculation of the likelihood).

NOTE1: Although the document may count the indel blocks as $eiidb = 1, 2, ... (1-based),
the program implementation here counts the indel blocks as $eiidb = 0, 1, 2, ... (0-based).

NOTE2: Two different series of indices, referred to as $cl and $i, are used to specify the gapped segments.
By examining the predecessor subroutine, "apprx1_tot_lnlk_gpattern_msa2" in "MyTreeMap_indels_ML_hs.pm",
I CONFIRMED that the "$cl th gapped segment" is actually identical to the "($i+1) th gapped segment" (when $cl = $i).
(That is, {$i +1} count the gapped segments in a 1-based manner, and {$cl} count them in a 0-based manner.)