$O(n^2)$ dynamic programming algorithms for

1. Allowed timing incompatibilities (as CoRe-PA does)
2. Uniform host-switch cost
3. Unlimited host-switch distance

For this writing, by replacing e for edges with v, we refer to the node at the lower (later) endpoint of the edge e. For example, v_P is the tip of e_P that happens later.

Also, for edge e, we refer to its children edges (whose higher tip is v) as e_1 and e_2. We refer to its parent edge (whose lower tip is the higher tip of e) as e_p, and its sibling (the other edge that share the higher tip with e) as e_s.

Case 1 host-switch allowed between any host edges

We will modify the optimized version of the sweep-time algorithm with assumptions 2 and 3. Firstly, we will remove the timing information from our dynamic programming. Let us define two new tables as following:

> $S(e_P, e_H)$ stores the minimum cost of associating e_P right after the time v_H occurs, including the cost of associating the subtrees of e_P and e_H. This will not include the host speciation that creates e_H. This table is comparable to A table in the sweep-time algorithm.
>
> $E(e_P, e_H)$ stores the minimum cost of associating e_P right before the time v_H occurs, including the cost of associating the subtrees of e_P and e_H. This must include the host speciation v_H. This table is comparable to B table in the sweep-time algorithm.

With the same reasoning as the sweep-time algorithm, this table can be filled out in the post-order of the parasite edges, and for each parasite edge, in any order of the host edges. Again, we will keep the table D defined as $D(e_P) = \min_{e_H} S(e_P, e_H)$ so that we can compute the cost for host-switch case in $O(1)$. $D(e_P)$ is updated if a new $S(e_P, e_H)$ is better than the value stored in $D(e_P)$, so it does not increase the asymptotic running time. The overall algorithm can be summarized as shown below:

```
For each parasite edge e_P in post-order
        D(e_P) ← ∞
        For each host edge e_H in any order
                Compute E(e_P, e_H)
        For each host edge e_H in any order
                Compute S(e_P, e_H)
                D(e_P) ← min{D(e_P), S(e_P, e_H)}
```

Our goal is to compute $E(e_P, e_H)$ and $S(e_P, e_H)$ in $O(1)$, so that the overall running time is $O(n^2)$.

Computation of entries of $E(e_P, e_H)$

If an endpoint of e_P is a tip, then $E(e_P, e_H) = 0$ if one end of e_H is a tip associated at a tip of e_P. Otherwise, $E(e_P, e_H) = \min\{\text{COSPEC}(e_P, e_H), \text{LOSS}(e_P, e_H)\}$, where:

- If both endpoints of both e_P and e_H are not tips, then
  $\mathrm{COSPEC}(e_P, e_H) = \min\{S(e_{P_1}, e_{H_1}) + S(e_{P_2}, e_{H_2}), S(e_{P_1}, e_{H_2}) + S(e_{P_2}, e_{H_1})\} + \mathrm{cost}_{\mathrm{COSPEC}}$.
  Otherwise, $\mathrm{COSPEC}(e_P, e_H) = \infty$.
- If both endpoints of e_H are not tips, then
  $\mathrm{LOSS}(e_P, e_H) = \min\{S(e_P, e_{H_1}), S(e_P, e_{H_2})\} + \mathrm{cost}_{\mathrm{LOSS}}$.
  Otherwise, $\mathrm{LOSS}(e_P, e_H) = \infty$.

Computation of entries of $\boldsymbol{S(e_P, e_H)}$

$S(e_P, e_H) = \min\{E(e_P, e_H), \mathrm{DUP}(e_P, e_H), \mathrm{HS}(e_P, e_H)\}$, where

- If both endpoints of e_P are not tips, then $\mathrm{DUP}(e_P, e_H) = S(e_{P_1}, e_H) + S(e_{P_2}, e_H) + \mathrm{cost}_{\mathrm{DUP}}$.
  Otherwise, $\mathrm{DUP}(e_P, e_H) = \infty$.
- If both endpoints of e_P are not tips, then
  $\mathrm{HS}(e_P, e_H) = \min\{S(e_{P_1}, e_H) + D(e_{P_2}), S(e_{P_2}, e_H) + D(e_{P_1})\} + \mathrm{cost}_{\mathrm{HS}}$.
  Otherwise, $\mathrm{HS}(e_P, e_H) = \infty$.

Case 2 host-switch not allowed for same take-off and landing host edges

To solve this, we can simply modify the algorithm in case 1 to keep best two host-switch costs, along with the site to switch to for the best one. To compute $\mathrm{HS}(e_P, e_H)$, if e_H is not the best site to switch to, then we can safely switch to e_H. Otherwise, we can simply choose the second best instead. To update $D(e_P)$ when $\mathrm{HS}(e_P, e_H)$ is computed, we need to avoid storing the switches to the same edge by, in case e_H is as better than the second best solution, make sure not to replace the second solution if the best solution is also has e_H as its landing site.

Case 3 host-switch not allowed between same or ancestor-descendant host edges

Firstly, notice that we cannot simply keep some constant number of landing sites because we need as much as $O(n)$ sites if they are all incident to tips. So in this case, we will prepare a table D defined as follows:

$$D(e_P, e_H) = \min_{e_{H'} \text{ is } \textbf{not } e_H, \text{its descendant,or its ancestor}} S(e_P, e_{H'})$$

This definition means that $D(e_P, e_H)$ will keep the cost of the optimal host-switch allowed for take-off site e_H. This allows us compute

$$\mathrm{HS}(e_P, e_H) = \min\{S(e_{P_1}, e_H) + D(e_{P_2}, e_H), S(e_{P_2}, e_H) + D(e_{P_1}, e_H)\} + \mathrm{cost}_{\mathrm{HS}}.$$

in constant time. The other formulas from case 1 are still applicable.

Next, notice that since we compute the table in post-order of e_P, then we do not need to update the D table right away. We can compute $S(e_P, e_H)$ for all e_H, before we compute $D(e_P, e_H)$ for all e_H. As long as this computation takes $O(n)$ for all e_H for each e_P, then the total running time will still be $O(n^2)$. However, to compute D table efficiently, we introduce the C table:

$$C(e_P, e_H) = \min_{e_{H'} \text{ is } e_H \text{ or its descendant}} S(e_P, e_{H'})$$

Please think of this table C is a mathematical function – it does not have any interpretation for the biological problem instance. We can define $C(e_P, e_H)$ recursively as follows:

- If v_H is a tip, then $C(e_P, e_H) = S(e_P, e_H)$.
- Otherwise, $C(e_P, e_H) = \min\{S(e_P, e_H), C(e_P, e_{H_1}), C(e_P, e_{H_2})\}$.

Therefore, we can fill out the table C for all e_H for each e_P in post-order, which has the running time of $O(n)$.

Now, we can define D recursively as follows:

- If e_H is the dummy root edge, then $D(e_P, e_H) = \infty$.
- Otherwise, $D(e_P, e_H) = \min\left\{C(e_P, e_{H_s}), D\left(e_P, e_{H_p}\right)\right\}$.

The correctness of both recursive formulas for the computation of C and D can be proved inductively.

As a result, we can also fill out table C for all e_H for each e_P in pre-order in $O(n)$. Therefore, the running time of C and D for each e_P is $O(n)$, and the total running time is still $O(n^2)$. The overview of this modified algorithm is shown below.

```
For each parasite edge e_P in post-order
    For each host edge e_H in any order
        Compute E(e_P, e_H)
    For each host edge e_H in any order
        Compute S(e_P, e_H)
    For each host edge e_H in post-order
        Compute C(e_P, e_H)
    For each host edge e_H in post-order
        Compute D(e_P, e_H)
```