Packing Coloring on full 3-degree tree

Algorithm 1: Packing Coloring on Full Binary Tree

```
Color Every odd layer with color 1.
for Every Layer $l \in \{2, 4, 6, \dots\}$ do
    if $l = 2$ then
        Color left vertex with 2;
        Color right vertex with 3;
    else if $l = 4$ then
        if Grand Parent Color $= 2$ then
            Color children with $3, 4, 3, 5$;
        end
        else if Grand Parent Color $= 3$ then
            Color children with $2, 4, 2, 5$;
        end
    end
    /* For other layers lower than 4 */
    if Grand Parent Color $\in \{4, 5, 6, 7\}$ then
        Color children with $2, 3, 2, 3$;
        GOTO next node;
    end
    if Grand Parent Color is 3 or 2 then
        $x \leftarrow$ grandparent node;
        if color$(x) = 3$ then
            Color each left grand-childern with 2;
        else if color$(x) = 2$ then
            Color each left grand-childern with 3;
        end
        $(u_1, u_2, u_3), x, (v_1, v_2, v_3)$ is a triplet $u_j, v_j$ are left and right siblings
         of $x$;
        $u_j, v_j \leftarrow \phi$ if there is none exists, if either is $\phi$ ignore;
        $C \leftarrow$ visit each $u_i, v_i$ upto distance 7 to find an un-used color;
        Color node with $C$;
    end
end
Output Coloring;
```

Two Pass Strategy

Algorithm 2: Two Pass Strategy

```
Activated ← $\phi$;
for Every event $(k, v_k)$ do
    $v(k) = v_k$;
    for $\forall j \in$ fanout of $k$ do
        Update the value of j;
        Activated ← Activated $\cup \{j\}$;
    end
end
for $j \in$ Activated do
    begin
        $v' =$ evaluate$(j)$;
        if $v' \neq lsv(j)$ then
            add event $(j, v')$ to the event list at time $t + d(j)$;
            $lsv(j) = v'$;
        end
    end
end
```

One Pass Strategy

Algorithm 3: One Pass Strategy

```
for Every event $(k, v_k)$ do
    $v(k) = v_k$;
    for $\forall j \in$ fanout of $k$ do
        Update the value of j;
        Instead of putting $j$ to activated set, process it here;
        begin
            $v' =$ evaluate$(j)$;
            if $v' \neq lsv(j)$ then
                add event $(j, v')$ to the event list at time $t + d(j)$;
                $lsv(j) = v'$;
            end
        end
    end
end
```

One Pass Strategy With Zero-Width Spikes

Algorithm 4: One Pass Strategy With Zero-Width Spikes

```
for Every event $(k, v_k)$ do
    $v(k) = v_k$;
    for $\forall j \in$ fanout of $k$ do
        Update the value of j;
        Instead of putting $j$ to activated set, process it here;
        begin
            $v' = \text{evaluate}(j)$;
            if $v' \neq lsv(j)$ then
                $t' = t + d(j)$;
                if $t' = lst(j)$ then
                    cancel event $(j, lsv(j))$ at time $t'$;
                end
                add event $(j, v')$ to the event list at time $t + d(j)$;
                $lsv(j) = v'$;
                $lst(j) = t'$;
            end
        end
    end
end
```