Pseudo Code for "Dependent Weighted Random Sampling with Colors using 1-D Range Tree"

Alexander Madrigal, Nathaniel Madrigal

18 October, 2023

Algorithm 1 $Build1DRangeTree(P)$

Input. A set of P colored points on a line, A dictionary of D color keys paired with weights

Output. The root of a 1-dimensional range tree

if P contains only one point **then**
  Create a leaf node v storing this point
  $v_{key} \leftarrow u^{1/D(v_{color})}$, where $u_i = random(0, 1)$ and $D(v_{color})$ is weight of v_{color}
  $v_{maxNode} \leftarrow v$
else
  Split P into two subsets; one subset P_{left} contains $x \leq x_{mid}$, the median x-coordinate, and the other subset P_{right} contains $x > x_{mid}$
  Create a node v storing point at x_{mid}
  $v_{left} \leftarrow Build1DRangeTree(P_{left})$
  $v_{right} \leftarrow Build1DRangeTree(P_{right})$
  if $key(v_{left}) > key(v_{right})$ **then**
    $v_{key} \leftarrow key(v_{left})$
    $v_{maxNode} \leftarrow maxNode(v_{left})$
  else
    $v_{key} \leftarrow key(v_{right})$
    $v_{maxNode} \leftarrow maxNode(v_{right})$
return v

Algorithm 2 $FindSplitNode(T, x, x')$

Input. A 1-dimensional range tree T, x, and x' where $x \leq x'$
Output. The node v where the paths to x and x' split, or the leaf where both paths end

```
1: v ← root(T)
2: while v is not a leaf and (x' ≤ x_v or x > x_v) do
3:     if x' ≤ x_v then
4:         v ← lc(v)
5:     else
6:         v ← rc(v)
7: return v
```

Algorithm 3 $FindCanonicalSet(T, [x : x'])$

Input. A 1-dimensional range tree T and a range $[x : x']$
Output. A set of all canonical nodes in T that lie in the range

```
1: Create an empty set C
2: v_split ← FindSplitNode(T, x, x')
3: if v_split is a leaf then
4:     Add point v_split to C if point is in range
5: else
6:     (* Follow the path to x and add points to the right of the path to C *)
7:     v ← lc(v_split)
8:     while v is not a leaf do
9:         if x ≤ x_v then
10:            Add rc(v) to C
11:            v ← lc(v)
12:        else
13:            v ← rc(v)
14:    Add the point stored at leaf v to C if point is in range
15:    Similarly, follow the path to x', add points to the left of path to C, and
       check if point stored a the leaf where the path ends is in range and must
       be added to C
16: return C
```

Algorithm 4 $WeightedRandomColorNode(C)$

Input. A set of canonical nodes C
Output. A weighted random node stored within canonical nodes C

```
1: Store the canonical node with the greatest key in c_max
2: return maxNode(c_max)
```