

# Pseudo Code for “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

```
1: if  $P$  contains only one point then
2:   Create a leaf node  $v$  storing this point
3:    $v_{key} \leftarrow u^{1/D(v_{color})}$ , where  $u_i = \text{random}(0,1)$  and  $D(v_{color})$  is weight of
    $v_{color}$ 
4:    $v_{maxNode} \leftarrow v$ 
5: else
6:   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  $x > x_{mid}$ 
7:   Create a node  $v$  storing point at  $x_{mid}$ 
8:    $v_{left} \leftarrow \text{Build1DRangeTree}(P_{left})$ 
9:    $v_{right} \leftarrow \text{Build1DRangeTree}(P_{right})$ 
10:  if  $\text{key}(v_{left}) > \text{key}(v_{right})$  then
11:     $v_{key} \leftarrow \text{key}(v_{left})$ 
12:     $v_{maxNode} \leftarrow \text{maxNode}(v_{left})$ 
13:  else
14:     $v_{key} \leftarrow \text{key}(v_{right})$ 
15:     $v_{maxNode} \leftarrow \text{maxNode}(v_{right})$ 
16: 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:  $v \leftarrow \text{root}(T)$ 
2: while  $v$  is not a leaf and  $(x' \leq x_v \text{ or } x > x_v)$  do
3:   if  $x' \leq x_v$  then
4:      $v \leftarrow lc(v)$ 
5:   else
6:      $v \leftarrow rc(v)$ 
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:  $v_{split} \leftarrow \text{FindSplitNode}(T, x, x')$ 
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:    $v \leftarrow lc(v_{split})$ 
8:   while  $v$  is not a leaf do
9:     if  $x \leq x_v$  then
10:      Add  $rc(v)$  to  $C$ 
11:       $v \leftarrow lc(v)$ 
12:     else
13:       $v \leftarrow rc(v)$ 
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 at 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  $\text{maxNode}(c_{max})$ 
```

---