Pseudo Code for "Uniform Random Sampling using 1-D Range Tree"

Nathaniel Madrigal, Alexander Madrigal

8 October, 2023

Algorithm 1 $Build1DRangeTree(P)$

Input. A set of P points on a line
Output. The root of a 1-dimensional range tree

if P contains only one point **then**
    Create a leaf node v storing this point and store 1 as the weight of v
else
    Split P into two subsets; one subset P_{left} contains x-coordinate less than or equal to x_{mid}, the median x-coordinate, and the other subset P_{right} contains points with x-coordinate larger than x_{mid}
    $v_{left} \leftarrow Build1DRangeTree(P_{left})$
    $v_{right} \leftarrow Build1DRangeTree(P_{right})$
    Create a node v storing x_{mid}, make v_{left} the left child of v, make v_{right} the right child of v, and make the sum of $weight(v_{left})$ and $weight(v_{right})$ the weight of v
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

$v \leftarrow root(T)$
while v is not a leaf **and** ($x' \leq x_v$ or $x > x_v$) **do**
    if $x' \leq x_v$ **then**
        $v \leftarrow lc(v)$
    else
        $v \leftarrow rc(v)$
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

Create an empty set C
$v_{split} \leftarrow FindSplitNode(T, x, x')$
if v_{split} is a leaf **then**
    Add point v_{split} to C if point is in range
else
    (* Follow the path to x and add points to the right of the path to C *)
    $v \leftarrow lc(v_{split})$
    while v is not a leaf **do**
        if $x \leq x_v$ **then**
            Add $rc(v)$ to C
            $v \leftarrow lc(v)$
        else
            $v \leftarrow rc(v)$
    Add the point stored at leaf v to C if point is in range
    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
return C

Algorithm 4 $UniformRandomNode(C)$

Input. A set of canonical nodes C with size n
Output. A uniform random node within the subleafs of C

(* Select a weighted random canonical node*)
for each canonical node $c_i \in C$ **do**
    Calculate a key $k_i = u_i^{1/w_i}$, where $u_i = random(0, 1)$ and w_i is the weight of node c_i
Store the canonical node with the greatest key in c_{max}
(* Traverse down c_{max} following the path with greatest keys *)
$v \leftarrow c_{max}$
while v is not a leaf **do**
    Calculate the key of left child $k_{lc} = u^{1/w_{lc}}$, where $u = random(0, 1)$ and w_{lc} is the weight of $lc(v)$
    Calculate the key of right child $k_{rc} = u^{1/w_{rc}}$, where $u = random(0, 1)$ and w_{rc} is the weight of $rc(v)$
    if $k_{rc} \leq k_{lc}$ **then**
        $v \leftarrow lc(v)$
    else
        $v \leftarrow rc(v)$
return v