KD-Trees

1 Setup

Algorithm 1 Setup $(dataset, depth, dim, root)$

$cd = [depth \mod dim]$
$dataset$ = sort dataset according to cd
$median\_points$ = at max 2 medians from the dataset
$median$ = first median point
$root.value = median$
$left\_array$ = dataset values to the left of the median
$right\_array$ = dataset values to the right of the median
if $left\_array$ is not empty **then**
    $root.left$ = Setup(left_array, depth+1, dim, root.left)
end if
if $right\_array$ is not empty **then**
    $root.right$ = Setup(right_array, depth+1, dim, root.right)
end if
return root

2 Searches

Algorithm 2 Search $(query, depth, dim, k, node, maxHeap, section\_reached)$

```
cd = [depth mod dim]
if section_reached is false then
    if node = leaf then
        section_reached = true
    end if
    if maxHeap.size < k then
        maxHeap.insert(node.value)
    else if node.value is closer than maxHeap.max then
        maxHeap.removeMax
        maxHeap.insert(node.value)
    end if
    if query[cd] < node[cd] and node is not leaf then
        node.leftExplored = true
        Search(query, depth+1, dim, k, node.left, maxHeap, section_reached)
    else if node is not leaf then
        node.rightExplored = true
        Search(query, depth+1, dim, k, node.right, maxHeap, section_reached)
    end if
else
    if maxHeap.size < k then
        maxHeap.insert(node.value)
    else if node.value is closer than maxHeap.max then
        maxHeap.removeMax
        maxHeap.insert(node.value)
    end if
    if node is leaf then
        Search(query, depth−1, dim, k, node.parent, maxHeap, section_reached)
    else
        if both children of node explored or node.value > maxHeap.max then
            Search(query, depth−1, dim, k, node.parent, maxHeap, section_reached)
        else if node.leftExplored and node.value < maxHeap.max then
            Search(query, depth+1, dim, k, node.right, maxHeap, section_reached)
        else if node.rightExplored and node.value < maxHeap.max then
            Search(query, depth+1, dim, k, node.left, maxHeap, section_reached)
        else if node.value < maxHeap.max then
            Search(query, depth+1, dim, k, node.left, maxHeap, section_reached)
            Search(query, depth+1, dim, k, node.right, maxHeap, section_reached)
        end if
    end if
end if
```

Algorithm 3 ModifiedSearch ($query, depth, dim, k, node, maxHeap, section\_reached, nodeCount, c$)

```
if nodeCount >= clogN then
   if maxHeap.length < k then
      fill maxHeap with random values upto k
   end if
   exit
end if
cd = [depth mod dim]
if section_reached is false then
   if node = leaf then
      section_reached = true
   end if
   if maxHeap.size < k then
      maxHeap.insert(node.value)
   else if node.value is closer than maxHeap.max then
      maxHeap.removeMax
      maxHeap.insert(node.value)
   end if
   if query[cd] < node[cd] and node is not leaf then
      node.leftExplored = true
      nodeCount+ = 1
      ModifiedSearch(query, depth + 1, dim, k, node.left, maxHeap, section_reached)
   else if node is not leaf then
      node.rightExplored = true
      nodeCount+ = 1
      ModifiedSearch(query, depth + 1, dim, k, node.right, maxHeap, section_reached)
   end if
else
   if maxHeap.size < k then
      maxHeap.insert(node.value)
   else if node.value is closer than maxHeap.max then
      maxHeap.removeMax
      maxHeap.insert(node.value)
   end if
   if node is leaf then
      nodeCount+ = 1
      ModifiedSearch(query, depth − 1, dim, k, node.parent, maxHeap, section_reached)
   else
      if both children of node explored or node.value > maxHeap.max then
         nodeCount+ = 1
         ModifiedSearch(query, depth−1, dim, k, node.parent, maxHeap, section_reached)
      else if node.leftExplored and node.value < maxHeap.max then
         nodeCount+ = 1
         ModifiedSearch(query, depth+1, dim, k, node.right, maxHeap, section_reached)
      else if node.rightExplored and node.value < maxHeap.max then
         nodeCount+ = 1
         ModifiedSearch(query, depth+1, dim, k, node.left, maxHeap, section_reached)
      else if node.value < maxHeap.max then
         nodeCount+ = 1
         ModifiedSearch(query, depth+1, dim, k, node.left, maxHeap, section_reached)
         nodeCount+ = 1
         ModifiedSearch(query, depth+1, dim, k, node.right, maxHeap, section_reached)
      end if
   end if
end if
```

Algorithm 4 ObliviousSearch $(query, depth, dim, k, node, maxHeap, section\_reached, nodeCount, c)$

```
if nodeCount >= clogN then
    exit
    if maxHeap.length < k then
        fill maxHeap with random values upto k
    end if
end if
cd = [depth mod dim]
if section_reached is false then
    if node = leaf then
        section_reached = true
    end if
    if maxHeap.size < k then
        maxHeap.insert(node.value)
    else if node.value is closer than maxHeap.max then
        maxHeap.removeMax
        maxHeap.insert(node.value)
    end if
    if query[cd] < node[cd] and node is not leaf then
        node.leftExplored = true
        nodeCount+ = 1
        node.left = OMAP.retrieve(node.leftLabel)
        ObliviousSearch(query, depth + 1, dim, k, node.left, maxHeap, section_reached)
    else if node is not leaf then
        node.rightExplored = true
        nodeCount+ = 1
        node.right = OMAP.retrieve(node.rightLabel)
        ObliviousSearch(query, depth + 1, dim, k, node.right, maxHeap, section_reached)
    end if
else
    if maxHeap.size < k then
        maxHeap.insert(node.value)
    else if node.value is closer than maxHeap.max then
        maxHeap.removeMax
        maxHeap.insert(node.value)
    end if
    if node is leaf then
        nodeCount+ = 1
        node.parent = OMAP.retrieve(node.parentLabel)
        ObliviousSearch(query, depth − 1, dim, k, node.parent, maxHeap, section_reached)
    else
        if both children of node explored or node.value > maxHeap.max then
            nodeCount+ = 1
            node.parent = OMAP.retrieve(node.parentLabel)
            ObliviousSearch(query, depth−1, dim, k, node.parent, maxHeap, section_reached)
        else if node.leftExplored and node.value < maxHeap.max then
            nodeCount+ = 1
            node.right = OMAP.retrieve(node.rightLabel)
            ObliviousSearch(query, depth+1, dim, k, node.right, maxHeap, section_reached)
        else if node.rightExplored and node.value < maxHeap.max then
            nodeCount+ = 1
            node.left = OMAP.retrieve(node.leftLabel)
            ObliviousSearch(query, depth+1, dim, k, node.left, maxHeap, section_reached)
    else if node.value < maxHeap.max then
        nodeCount+ = 1
        node.left = OMAP.retrieve(node.leftLabel)
        ObliviousSearch(query, depth + 1, dim, k, node.left, maxHeap, section_reached)
        nodeCount+ = 1
        node.right = OMAP.retrieve(node.rightLabel)
        ObliviousSearch(query, depth + 1, dim, k, node.right, maxHeap, section_reached)
    end if
end if
end if
```