Skip Quadtree Pseudocode

Luo Qian

September 4, 2014

Point

```
1  float32_t x, y;
```

SkipQuadtreeNode

```
1  SkipQuadtreeNode parent;
2  SkipQuadtreeNode up, down;
3  SkipQuadtreeNode[4] children;
4  double length; // side length of square, if is_square
5  Point center; // center of the square; length/2 distance from each edge
6  boolean is_square;
```

SkipQuadtree_search(SkipQuadtreeNode node, Point p)

```
 1  // call starts at the root of the highest-order tree
 2  // children[i] is quadrant i+1, relative to node
 3  int quadrant = getQuadrant(node, p);
 4  if node.children[quadrant].is_square
 5        return SkipQuadtree_search(node.children[quadrant],p)
 6  elseif node.children[quadrant] == p
 7        return true
 8  elseif node.children[quadrant].down is not null // not lowest-level tree
 9        return SkipQuadtree_search(node.children[quadrant].down,p)
10  return false
```

```
SkipQuadtree_add(SkipQuadtreeNode node, Point p)
1  if p is not in node
2       return false
3  while random() < 0.5 // still winning the coin toss
4       if node.up is null
5            create node.up and link properly // adding more levels
6       node = node.up
7  SkipQuadtree_add_helper(node, p)
8  return true

SkipQuadtree_add_helper(SkipQuadtreeNode node, Point p)
 1  // call starts at the root of the highest-order tree
 2  if node.down is not null
 3       downNode = SkipQuadtree_add(node.down, p)
 4  // children[i] is quadrant i+1, relative to node
 5  int quadrant = getQuadrant(node, p);
 6  SkipQuadtreeNode newNode = new SkipQuadtreeNode(x = p.x, y = p.y, is_square = false)
 7  newNode.down = downNode
 8  downNode.up = newNode
 9  newNode.parent = node
10  if node.children[quadrant] is null
11       node.children[quadrant] = newNode
12       return newNode
13  elseif node.children[quadrant].is_square
14       return SkipQuadtree_add(node.children[quadrant], p)
15  else // node.children[quadrant] is preexisting point
16       // square is the smallest square containing both node.children[quadrant] and newNode
17       SkipQuadtreeNode square = new SkipQuadtreeNode(x, y = center of square)
18       square's children are now node.children[quadrant] and newNode
19       while the existing child and p are in the same quadrant of square
20            square = smaller square representing the coinciding quadrant
21       replace node.children[quadrant] with square
22       return newNode
23  return null
```

SkipQuadtree_remove(SkipQuadtreeNode node, Point p)

```
1  // call starts at the root of the highest-order tree
2  // children[i] is quadrant i+1, relative to node
3  int quadrant = getQuadrant(node, p)
4  if node.is_square
5       return SkipQuadtree_remove(node.children[quadrant],p)
6  elseif node.children[quadrant] == p
7       if node has only one child, including the match // is a root node for that level
8            // note that this happens only at the highest-level root node each time
9            remove the matching child
10           remove the level by removing the root node
11      elseif node has only two children, including the match
12           replace this square with the other child in this level's tree
13      else
14           remove the matching child
15      if node.down is not null
16           SkipQuadtree_remove(node.down,p)
17      return true
18 elseif node.children[quadrant].down is not null // not lowest-level tree
19      return SkipQuadtree_remove(node.children[quadrant].down,p)
20 return false
```