```
Rebuild(x):
    A=empty array
    inorder(x,A)
    return buildBalanced(A,1,size_of(A))

inorder(x,A):
    if x = NULL return
    inorder(x.left,A)
    push(A,(x.key,x.val))
    inorder(x.right,A)
```

```
buildBalanced(A,from,to):
   mid = (from+to)/2
   root = new node(A[mid].key,A[mid].val)
   root.size = 1
   if (from<mid)
     // build left subtree
     left = buildBalanced(A,from,mid-1)
     root.left = left
     left.parent = root
     root.size += left.size
   if (to>mid)
     // build right subtree
     right = buildBalanced(A,mid+1,to)
     root.right = right
     right.parent = root
     root.size += right.size
   return root
```

```
Insert(key,val,root):
  if root = NULL
    x = new node(key,val)
**  x.size = 1
**  return (x,x,0)

  if key < root.key
**  (root.left,x,depth) = Insert(key,val,root.left)
    root.left.parent = root
**  root.size += 1
  else
**  (root.right,x,depth) = Insert(key,val,root.right)
    root.right.parent = root
**  root.size += 1

**return (root,x,depth+1)
```

```
Scapegoat_Insert(key,val,root):
  (root,x,depth) = Insert(key,val,root)
  if depth>maxDepth(root.size)
     scapegoat = findScapegoat(x)
     parent = scapegoat.parent
     y = Rebuild(scapegoat)
     if (parent = NULL)
        return y
     else
       if (parent.left = scapegoat)
          parent.left = y
       else
          parent.right = y
       y.parent = parent
   return root
```

```
findScapegoat(x):
  if (x.left and x.left.size>(2/3)*x.size)
    return x
  if (x.right and x.right.size>(2/3)*x.size)
    return x
  return findScapegoat(x.parent)
```

Slovníky: Zhrnutie

Metóda	Insert	Search	Delete	Analýza
Spájaný zoznam	$\Theta(1)$	$\Theta(n)$	$\Theta(n)$	worst-case
Utriedené pole	$\Theta(n)$	$\Theta(\log n)$	$\Theta(n)$	worst-case
Hašovanie s reťazením	$\Theta(1)$	$\Theta(n)$	$\Theta(n)$	worst-case
	$\Theta(1)$	$\Theta(1+\alpha)$	$\Theta(1+\alpha)$	expected
— s otvorenou adresáciou	$\Theta(n)$	$\Theta(n)$	N/A	worst-case
	$\Theta(\frac{1}{1-\alpha})$	$\Theta(\frac{1}{\alpha}\ln\frac{1}{1-\alpha})$ or $\Theta(\frac{1}{1-\alpha})$	N/A	expected
Binárne vyhľ. stromy	$\Theta(n)$	$\Theta(n)$	$\Theta(n)$	worst-case
	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$	average
AVL stromy	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$	worst-case
Scapegoat stromy	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log n)$	amortized
Tries (string of length m)	$\Theta(m)$	$\Theta(m)$	$\Theta(m)$	worst-case