

Obraz zawierający tekst, zrzut ekranu, Czcionka, numer

Opis wygenerowany automatycznie



function JoinRightAVL(l, v, r) (l>r)

(l', v', r') = expose(l) (mamy pewnosc ze l' jest rozne max o 1 od r')

if (Height(r') <= Height(r)+1)

T1 = Node(r', v, r)

if (Height(T1) <= Height(l')+1) then return Node(l', v', T1)

//(mozemy po prostu utworzyc nowe)

else return rotateLeft(Node(l', v', rotateRight(T1)))

//(rR bo r' moze byc jeden wieksze od r, czyli rL by zrobiło r'=+1 i r=-1 czyli zepsulo avl)

else

T1 = JoinRightAVL(r', v, r)

T2 = Node(l', v', T1)

if (Height(T1) <= Height(l')+1) return T2

else return rotateLeft(T2)

function JoinLeft(l, v, r) (l<r)

(l', v', r') = expose(r)

if (Height(l') <= Height(l)+1)

T1 = Node(l', v, l)

if (Height(T1) <= Height(r')+1) then return Node(T1, v', r')

else return rotateRight(Node(rotLeft(T1), v', r'))

else

T1 = JoinLeft(l', v, l)

T2 = Node(T1, v', r')

if (Height(T1) <= Height(r')+1) return T2

else return rotateRight(T2)

Zad 3.

size = n + 1

cur\_size = 0

tree = [[] for \_ in range(size)]

iscentroid = [False] \* size

# ctree = [[] for \_ in range(size)]

def dfs(src, visited, subtree):

visited[src] = True

subtree[src] = 1

cur\_size += 1

for adj in tree[src]:

if not visited[adj] and not iscentroid[adj]:

dfs(adj, visited, subtree)

subtree[src] += subtree[adj]

def findCentroid(src, visited, subtree):

iscentroid = True

visited[src] = True

heavy\_node = 0

for adj in tree[src]:

if not visited[adj] and not iscentroid[adj]:

if subtree[adj] > cur\_size//2:

iscentroid = False

if heavy\_node == 0 or subtree[adj] > subtree[heavy\_node]:

heavy\_node = adj

if iscentroid and cur\_size - subtree[src] <= cur\_size//2:

return src

else:

return findCentroid(heavy\_node, visited, subtree)

def findCentroidUtil(src):

visited = [False] \* size

subtree = [0] \* size

cur\_size = 0

dfs(src, visited, subtree)

for i in range(size):

visited[i] = False

centroid = findCentroid(src, visited, subtree)

iscentroid[centroid] = True

return centroid