2. space: we get n nodes, so, it's n tree

for each edge $(i,j) \in G.E$, there is one node i in tree rooted by j and a node j in tree rooted by i. So in total, n root nodes $+ 2m$ non-root nodes

Space complexity is $\Theta(n+2m)$

Time:

Degree(G,i): go to tree rooted by i

find # non-root nodes.

do this by BFS starting from root,

whenever explore a new node, counter $+1$.

$\Theta(n)$ for worst case

AvgDegree(G): do Degree(G,i) for all n vertices

get total degree,

for worst case, assume any two i,j, $i \neq j$

there is $(i,j) \in G.E$

So, each tree has $n-1$ non-root nodes,

it's $\Theta(n^2)$

ContainsEdge(i, j) :

find tree rooted by i, $\Theta(1)$

find j in tree, by AVL search

Θ(heigh of tree)

in worst case, i is incident to all other nodes, so, is $\Theta(\log_2 n)$

time : $\Theta(\log_2 n)$

InsertEdge(G, i, j) :

find tree rooted by i, insert j.

by avl insertion

Θ(heigh of tree)

in worst case, i is incident to all other nodes, so, is $\Theta(\log_2 n)$

do the same for tree rooted by j

time : $\Theta(\log_2 n)$

L: time

Degree(G, i): find Linked List with head i
count node in Linked List
one by one.
In worst case, all nodes are incidents
with i. So, it's $\Theta(n)$

AvgDegree(G): count lenth of a linked list, one by one.
for worst case, assume any two $i, j, i \neq j$
there is $(i, j) \in G.E$, so, count $n(n-1)$ times
return $\frac{n(n-1)}{n}$, $\Theta(n^2)$

ContainsEdge(G, i, j): find Linked List with head i
find j in that Linked List
In worst case, all nodes are incident
with i. So, it's $\Theta(n)$

InsertEdge(G, i, j): find Linked List with head i
insert j in that Linked List
insert in front (right after head)
$\Theta(1)$

Compare	L	L'
Degree	$\Theta(n)$	$\Theta(n)$
AvgDegree	$\Theta(n^2)$	$\Theta(n^2)$
ContainsEdge	$\Theta(n)$	$\Theta(\log_2 n)$
InsertEdge	$\Theta(1)$	$\Theta(\log_2 n)$