we are not using find-with-compression, so rank is just depth

```
node    A B C D E F G      A B C D E F G
parent  A B C D E F G      0 0 0 0 0 0 0 rank   ... add edge BC
  rx <- find(B) = B    links followed: 1
  ry <- find(C) = C    links followed: 1
  union(B,C), both rank 0, so parent[B] <- C
parent  A C C D E F G    0 0 1 0 0 0 0  ... add edge BF  links 3
parent  A C C D E C G    0 0 1 0 0 0 0  ... add edge DE  links 2
parent  A C C E E C G    0 0 1 0 1 0 0  ... add edge CD  links 3
parent  A C E E E C G    0 0 1 0 2 0 0  ... add edge AF  links 4
parent  E C E E E C G    0 0 1 0 2 0 0  ... add edge FG  links 4
parent  E C E E E C C    0 0 1 0 2 0 0
```

for Prim's, we do not need a UF structure, since we are always adding a link from a vertex not in the tree, to the tree

```
        A B C D E F G
intree  - - - - - - *      now add edge   fg
intree  - - - - - * *      now add edge   bf
intree  - * - - - * *      now add edge   bc
intree  - * * - - * *      now add edge   cd
intree  - * * * - * *      now add edge   de
intree  - * * * * * *      now add edge   af
intree  * * * * * * *
```

if the trees being combined have different depth, then the root of the shorter is changed so that its parent is the root of the taller, and so the depth of the new tree is the same as the max depth of the two previous trees

if the trees being combined have different depth, then the new tree has depth 1 greater than the depth of the previous trees

so depth increases if and only if two tree with the same depth are combined, in which case the depth increases by one. so then two trees, each with depth d, and so each with at least 2^d nodes, form a tree with depth $d+1$, and at least $2^d + 2^d = 2^{d+1}$ nodes