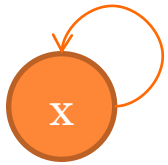
Implementacija z Gozdom

Za učinkovito implementacijo vozlišče potrebuje št. elementov poddrevesa:

```
public class DisjointSubset {
   Object value ;
   DisjointSubset parent ;
   int noNodes ; // moc podmnozice
} // class DisjointSubset
```

Operacija MAKESET iz enega elementa *x* tvori množico *{x}*



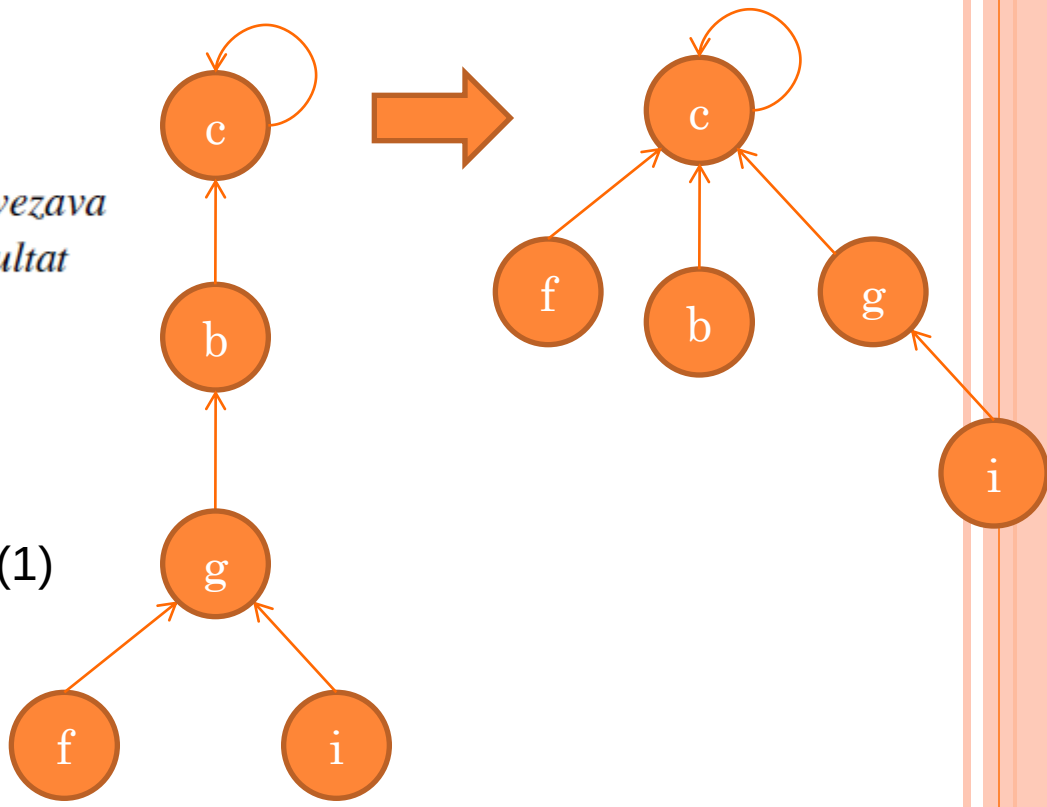

```
public DisjointSubset makeset(Object x) {
    DisjointSubset newEl = new DisjointSubset() ;
    newEl.value = x ;
    newEl.noNodes = 1 ;
    newEl.parent = newEl ;
    return newEl ;
}
```

Časovna zahtevnost: *O(1)*

Implementacija z Gozdom

FIND(x): vrne množico (t.j. koren drevesa), ki ji pripada element *x.*
Če je drevo izrojeno, je plezanje do korena lahko reda O(n).
Da se izrojenosti na dolgi rok izognemo,
vsa vozlišča na poti prevežemo na koren:

```
public DisjointSubset find(DisjointSubset x) {
  if (x == x.parent)
    return x ;
  else {
    x.parent = find(x.parent) ; // prevezava
    return x.parent ; // in hkrati rezultat
  }
} // find
```



FIND(f):
O(n) → O(1)

Implementacija z Gozdom

```
public void union(DisjointSubset a1, DisjointSubset a2) {
   DisjointSubset s1 = find(a1) ;
   DisjointSubset s2 = find(a2) ;
   if (s1.noNodes >= s2.noNodes) {
      s2.parent = s1 ;
      s1.noNodes += s2.noNodes ;
   }
   else {
      s1.parent = s2 ;
      s2.noNodes += s1.noNodes ;
   }
} // union
```

Časovna zahtevnost unije: *O(1)*