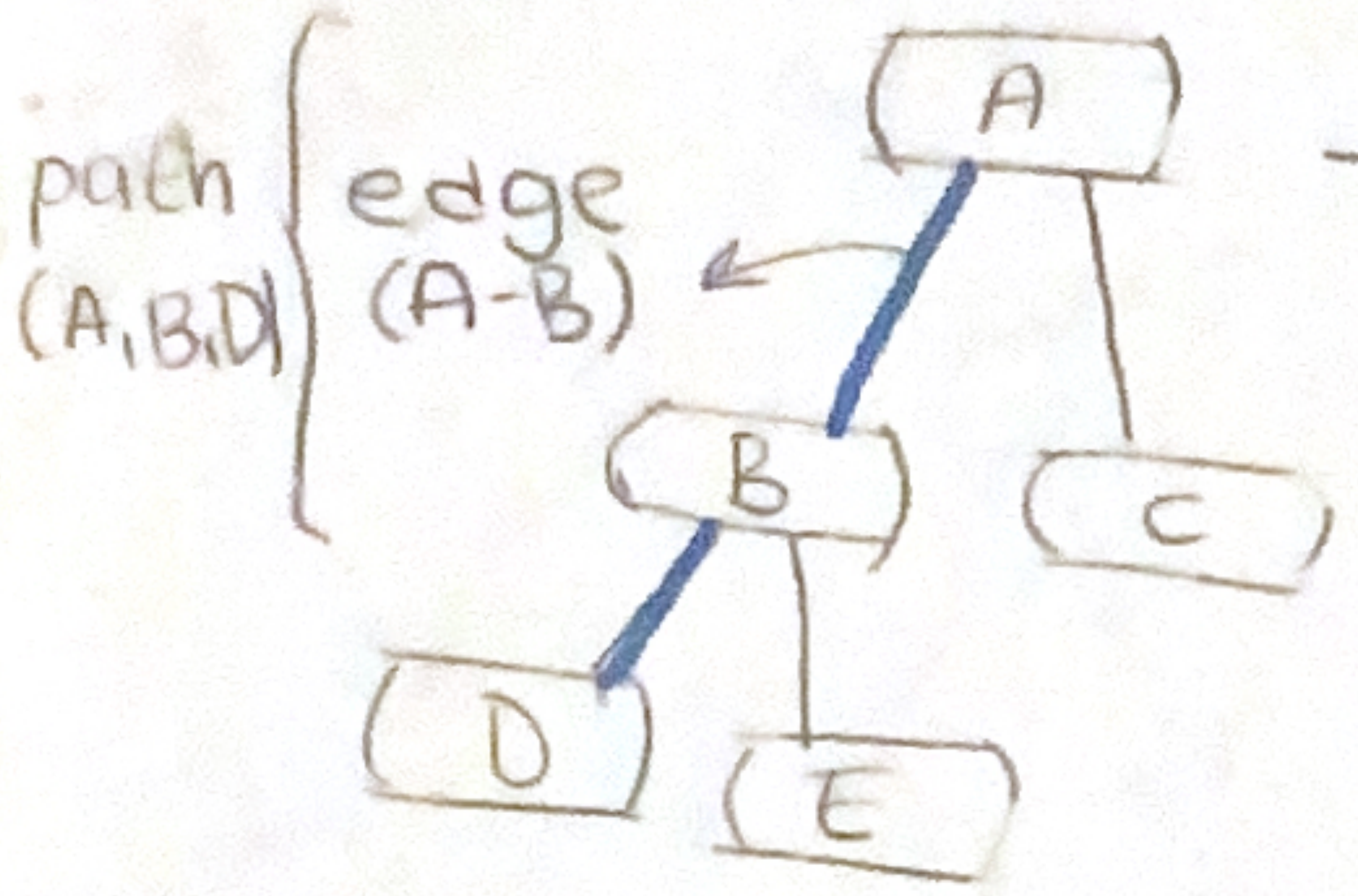
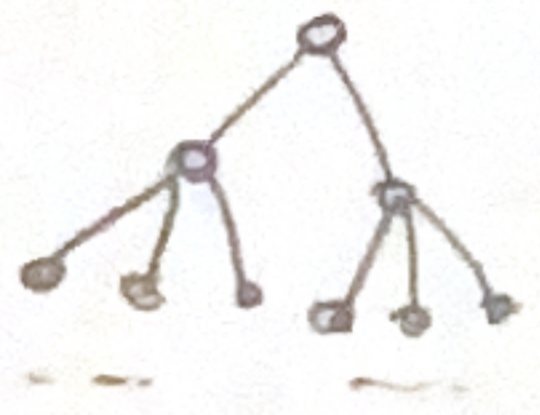
Trees



→ parent (ancestor)

→ children (descendant)



ordered tree (Çocuklarını 1. 2. 3. diye ayırt edebiliyorsak)

Abstract Data Type

p.element() → returns the element stored at position p.

T.root () → root of tree

T.is_root(p) → if p is the root

T. parent (p) → parent of p

T. children (p) → children of p

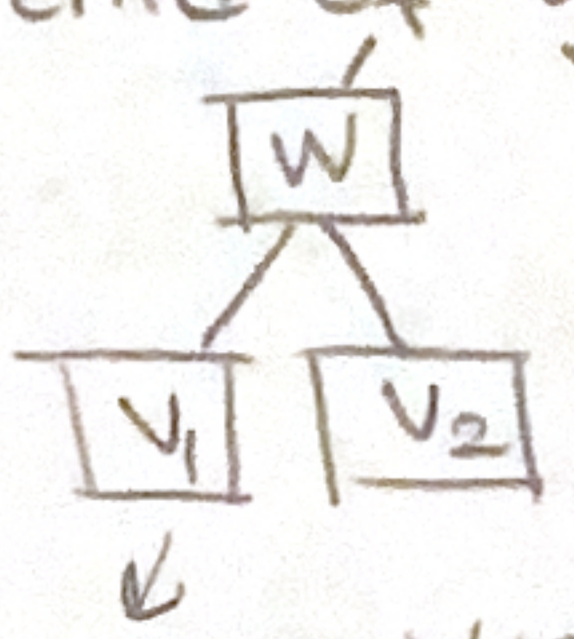
T. is_leaf (p) → True if p is leaf

→ bunlar Tree ABC'sinin abstract method'ları.

Yeni tree subclass'ı oluştururken override ediyoruz.

tree T: set of nodes storing elements s.t.

- nodes have parent-child relationships
- If T is nonempty, it has a root node (has no parent)
- Every node v of T different from the root has a unique parent w, every node with a parent w is a child of w.



} siblings

external / leaves (has no child)

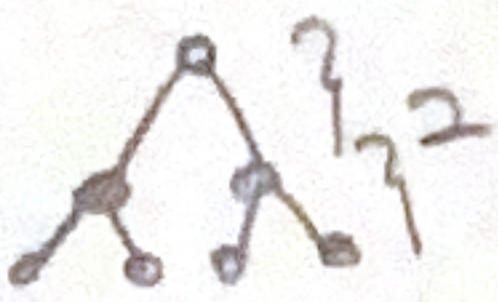
Depth of p: p'den önceki node'ların sayısı, p hariç. root'un depth'i 0.

$depth_p = 1 + depth_{\text{parent of } p}$

↓

$O(d_p + 1)$, performs constant time recursive step for each ancestor of p. (worst case'i $O(n)$

Height of p: maximum of depths of its leaf positions. (tau caht?)

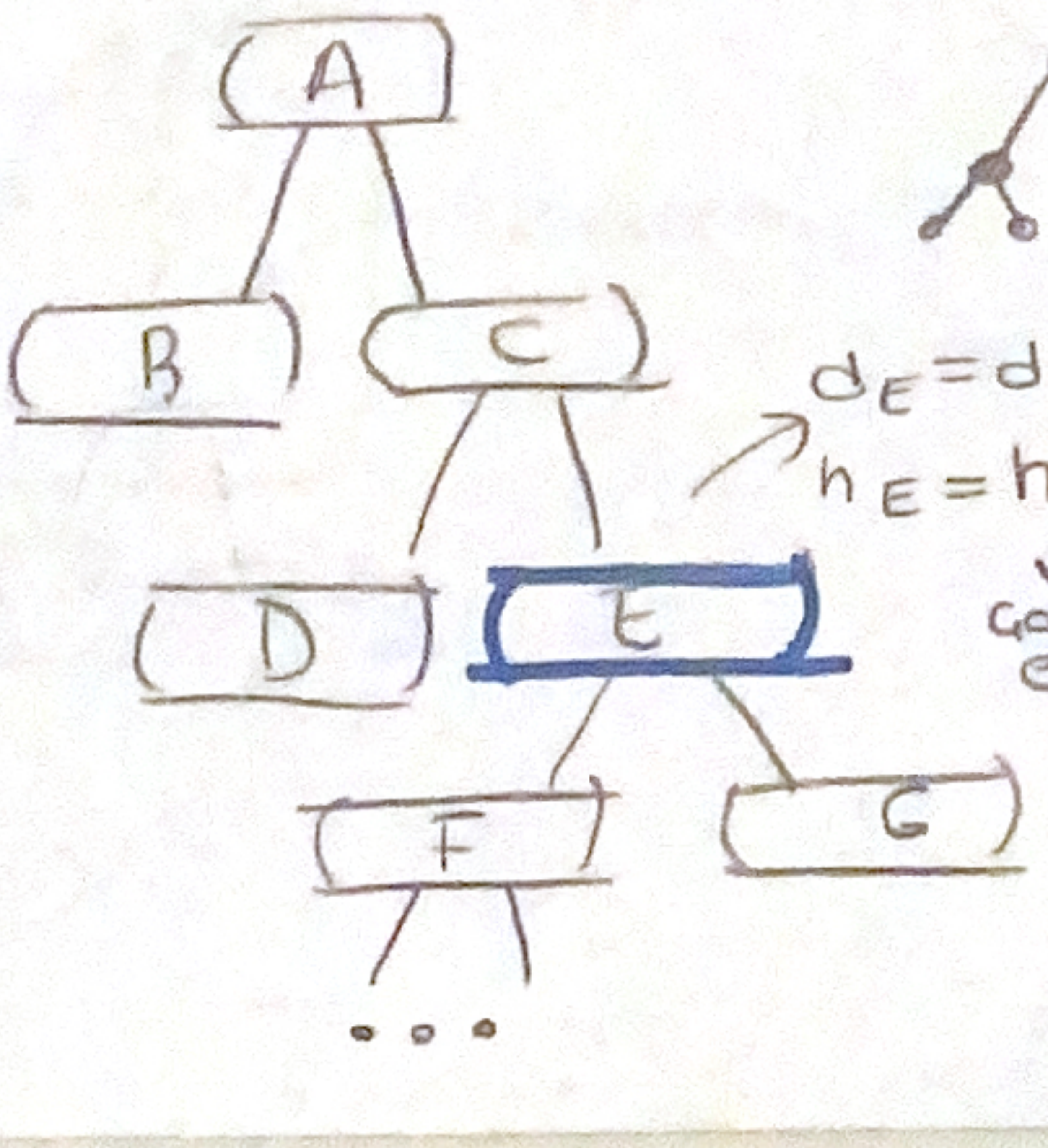


T ağacının height'ı root'un height'ına eşit.

→ p leaf node'sa height'ı 0

gocuklarından en yükseği → değilse ($h_{children} + 1$)

→ worst case $O(n)$ → recursive



$d_E = d_C + 1$

$h_E = h_F + 1$