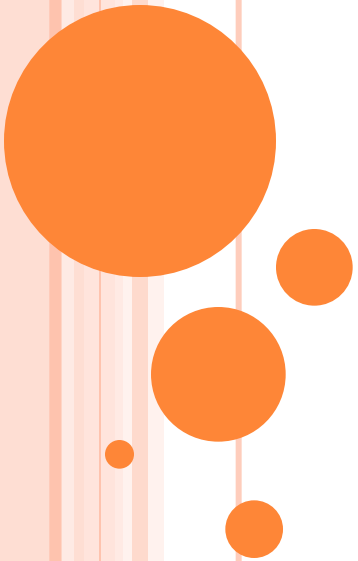


ADT DREVO

ADT TREE: IMPLEMENTACIJA

- PARENT(n, T) – vrne očeta vozlišča *n* v drevesu *T*
- LEFTMOST_CHILD(n, T) – vrne najbolj levega sina vozlišča *n*
- RIGHT_SIBLING(n, T) – vrne desnega brata vozlišča *n*
- LABEL(n, T) – vrne oznako vozlišča *n*
- ROOT(T) – vrne koren drevesa *T*
- MAKENULL(T) – naredi prazno drevo *T*
- CREATE(r, v, T1,…, Ti) – generira drevo s korenom *r* z oznako *v* ter s stopnjo *i* s poddrevesi *T1*, …, *Ti*

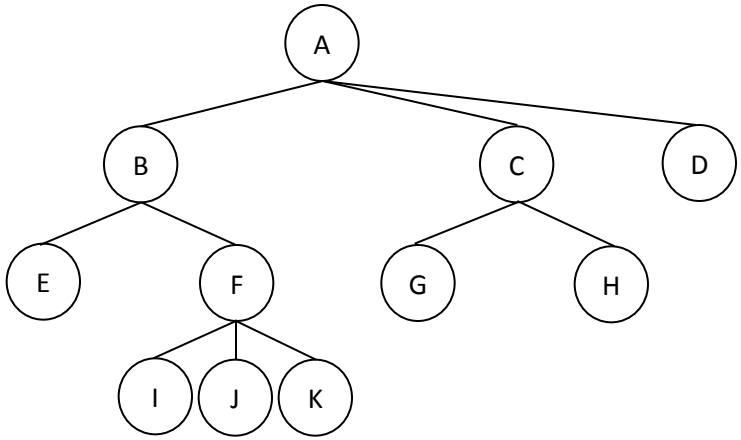
IMPLEMENTACIJA:

- *s poljem*
- *s kazalci*

IMPLEMENTACIJA DREVESA S POLJEM

Drevo je shranjeno v polje na naslednji način:

- vsako vozlišče hrani število sinov, celo drevo pa število vozlišč *n*
- koren drevesa je prvi element polja
- sledijo vsa vozlišča prvega (najbolj levega) poddrevesa korena, nato vsa vozlišča drugega poddrevesa in tako naprej
- vsako poddrevo je shranjeno po istem pravilu



A	3	koren
B	2	prvo poddrevo
E	0	
F	3	
I	0	
J	0	
K	0	
C	2	drugo poddrevo
G	0	
H	0	
D	0	tretje poddrevo

oznaka št. sinov

Implementacija s poljem

Ker polje ni dinamična podatkovna struktura, se redko uporablja za implementacijo dreves.

Problematično je spreminjanje dreves

- `LEFTMOST_CHILD(n, T)` – `O(1)`
- `LABEL(n, T)` – `O(1)`
- `ROOT(T)` – `O(1)`
- `MAKENULL(T)` – `O(1)`
- `CREATE(r, v, T1,…, Ti)` – `O(`*n*`)`
- `PARENT(n, T)` – `če eksplicitno skranjen: O(1)`
- `RIGHT_SIBLING(n, T)` – *O(k)*, *k* je število vozlišč poddrevesa

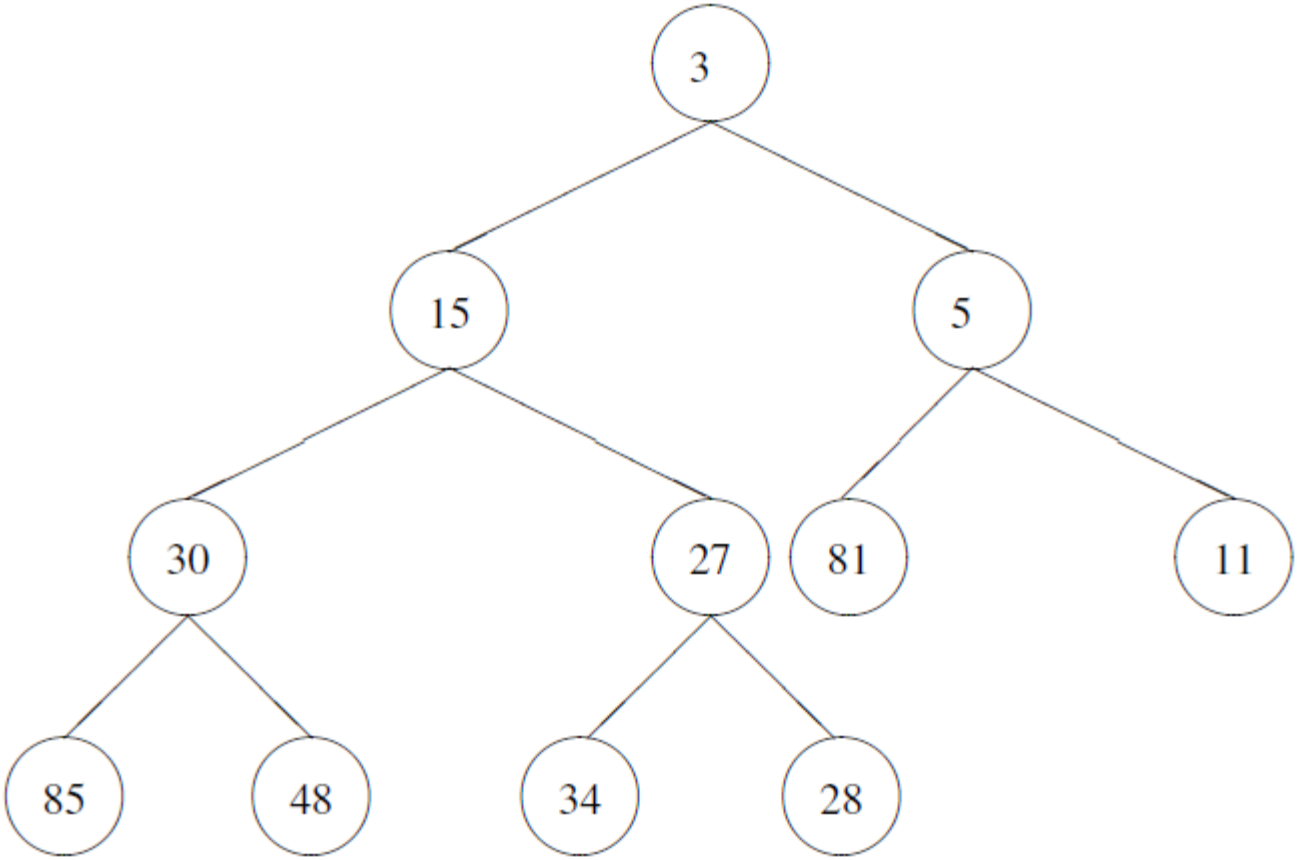
Implementacija s poljem

Za levo poravnana drevesa s konstantno stopnjo so operacije sprehajanja po drevesu učinkovite:

- `LEFTMOST_CHILD(n, T)` – $O(1)$
- `LABEL(n, T)` – $O(1)$
- `ROOT(T)` – $O(1)$
- `MAKENULL(T)` – $O(1)$
- `CREATE(r, v, T1,…, Ti)` – $O(n)$
- `PARENT(n, T)` – $O(1)$
- `RIGHT_SIBLING(n, T)` – $O(1)$

Implementacija s poljem

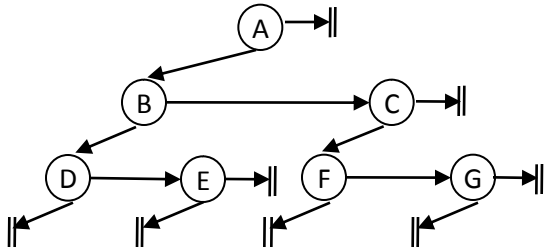
KOPICE: levo (ali desno) poravnana drevesa s konstantno stopnjo



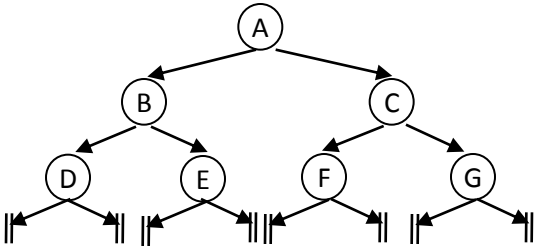
IMPLEMENTACIJA DREVESA S KAZALCI

Osnovni obliki predstavitve drevesa:

- vsako vozlišče vsebuje kazalca na levega otroka in desnega brata



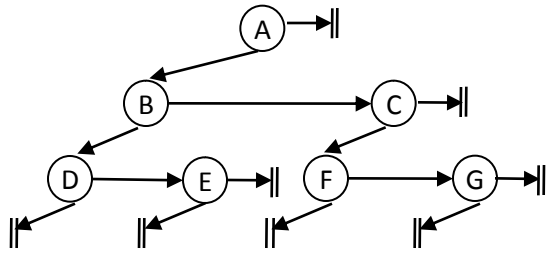
- vsako vozlišče vsebuje kazalce na vse otroke (običajno drevesa z navzgor omejeno stopnjo, npr. binarna drevesa)



V obeh primerih lahko vozlišče vsebuje tudi kazalec na očeta.

IMPLEMENTACIJA DREVESA S KAZALCI

- vsako vozlišče vsebuje kazalca na levega otroka in desnega brata



- drevo je podano s kazalcem na koren drevesa
- eno vozlišče vsebuje tri kazalce: oče, levi sin, desni brat:

```
public class TreeLSRSnode extends TreeNode {
   public TreeLSRSnode parent, leftSon, rightSibling ;
}
```

IMPLEMENTACIJA DREVESA S KAZALCI

Operacije so učinkovite – tudi učinkovito spreminjanje drevesa:

- `LEFTMOST_CHILD(n, T) – O(1)`
- `LABEL(n, T) – O(1)`
- `ROOT(T) – O(1)`
- `MAKENULL(T) – O(1)`
- **`CREATE(r, v, T1,…, Ti) – O(1)`**
- `PARENT(n, T) – O(1)`
- `RIGHT_SIBLING(n, T) – O(1)`

BINARNA DREVESA

BINARNA drevesa:

- vsa vozlišča s stopnjo manjšo ali enako 2
- vozlišče ima lahko tudi samo desnega sina

Lastnosti binarnih dreves:

- binarno drevo višine v ima največ $2^v - 1$ vozlišč
- višina binarnega drevesa z n vozlišči: $n \geq v \geq \lceil \log_2(n+1) \rceil$
- v binarnem drevesu z n vozlišči je $n + 1$ praznih poddreves
- Na koliko načinov lahko izrodimo binarno drevo z n vozlišči?