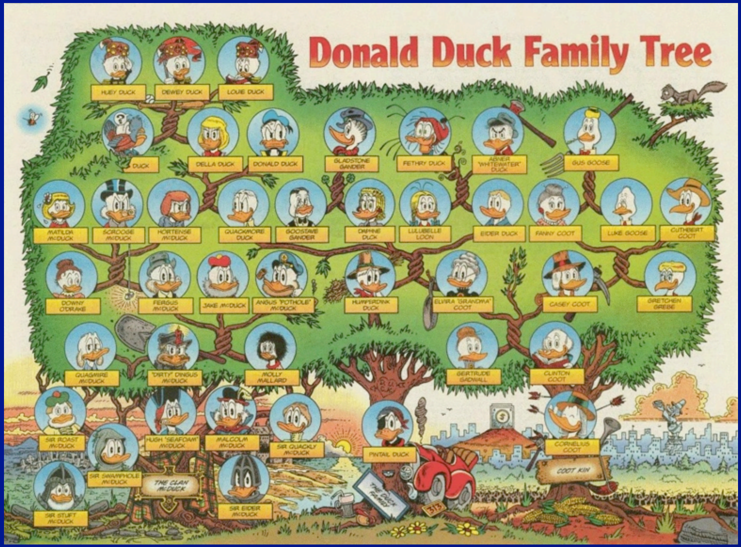
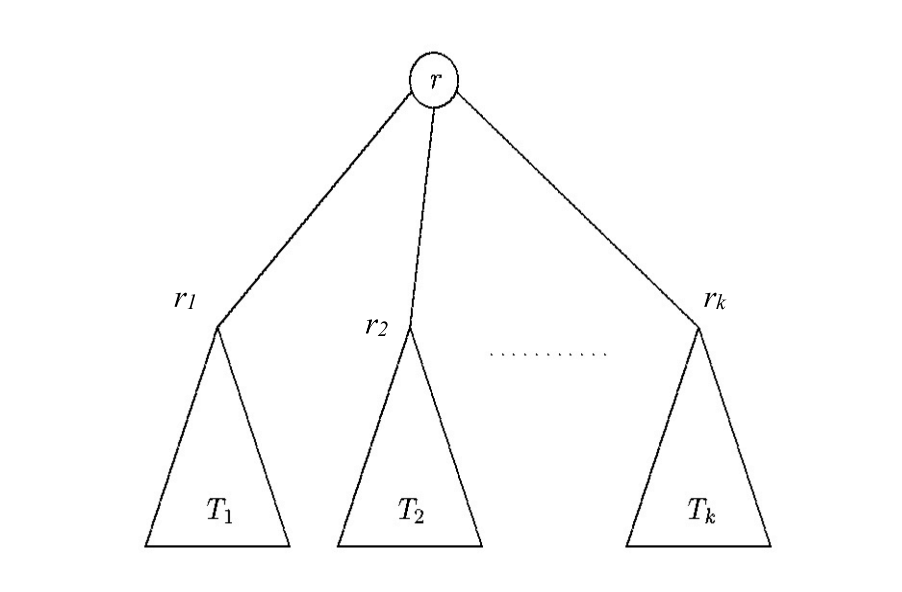

Stabla

Stablo

- Primjeri: obiteljsko stablo, sadržaj knjige
- Liste su bile linearno uređeni podaci (postoji predhodnik i sljedbenik, ali su svi na istoj razini)
- Stabla su hijerarhijski uređeni podaci (postoje prethodnik i sljedbenik, ali i više razina)
- Definicija: uređeno stablo T je objekt koja simulira hijerarhijsku strukturu skupom povezanih čvorova
- Postoji jedan istaknuti čvor r (korijen od stabla T)
- Svi čvorovi grade konačni niz (T_1,T_2, …,T_k) od 0 ili više disjunktnih manjih stabala

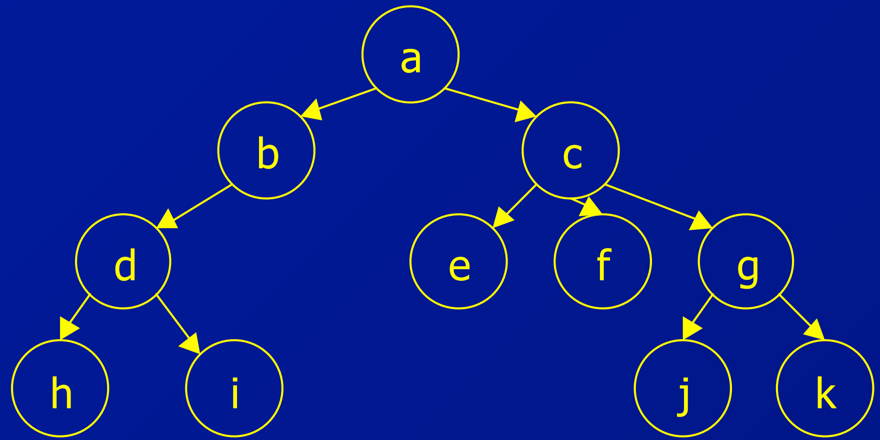


- manja stabla T_i se nazivaju podstabla korijena r
- korijeni r_i od T_i su djeca od r, a r je njihov roditelj
- korijen r nema roditelja
- svaki preostali član ima jednog roditelja
- djeca istog čvora su braća



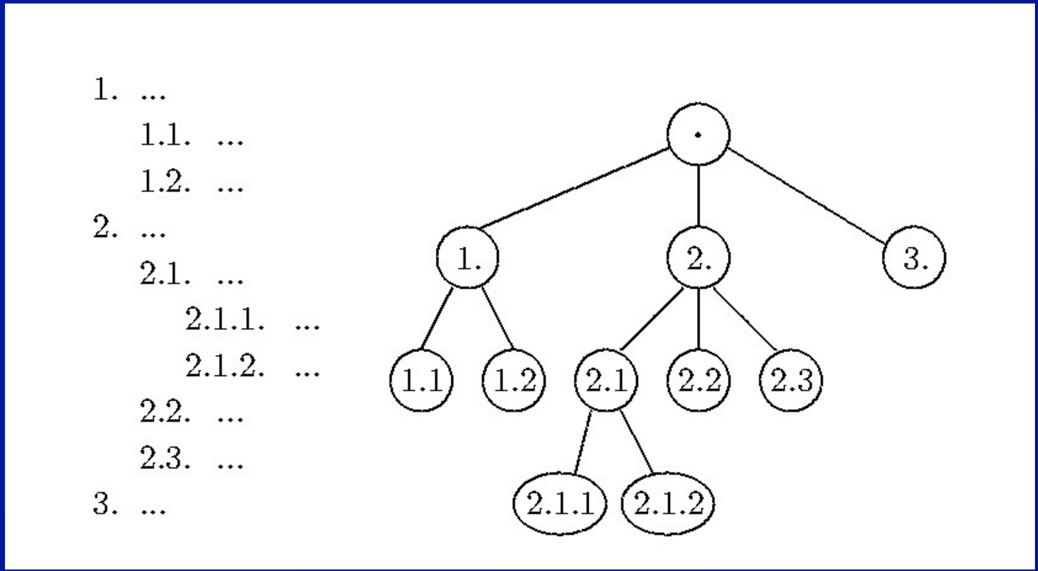
- Uređenost stabla se očituje u tome da među podstablima postoji linearno uređenje (određeno je koje je prvo, koje drugo, itd…)

- Primjer:



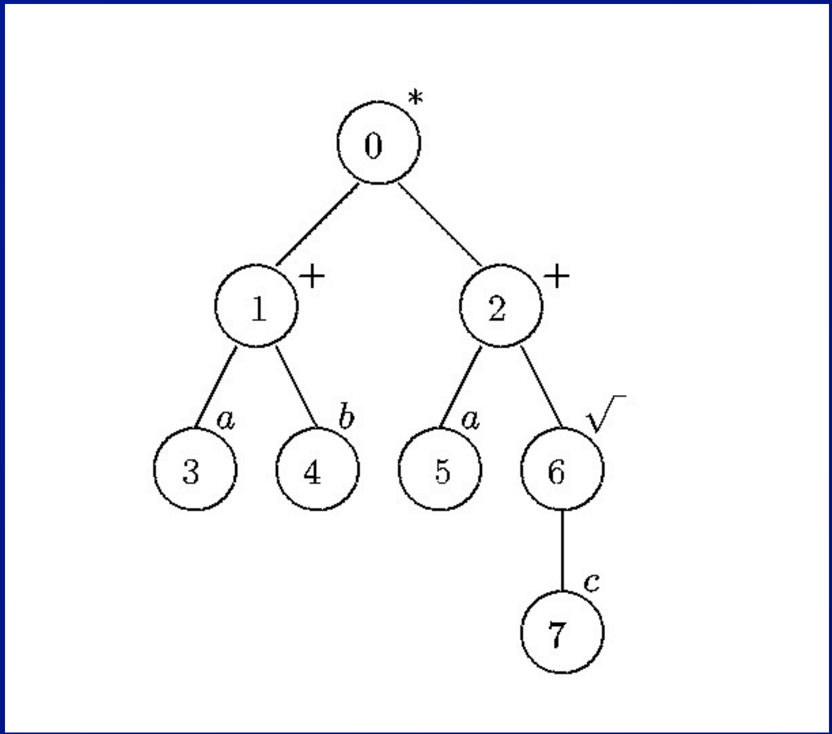
  - a je korijen stabla
  - Stupanj čvora je broj podstabala tog čvora (čvor a ima stupanj 2)
  - Skup $\{h,i,e,f,j,k\}$ je skup krajnjih čvorova – listova, koji nemaju djece
  - Unutrašnji čvor je čvor koji nije list (čvor d je unutrašnji čvor)
  - Stupanj stabla je maksimalni stupanj čvorova tog stabla (ovdje je to čvor c sa stupanjem 3)
  - Razina nekog čvora se određuje iz definicije da je korijen razine 1, a da su razine djece nekog čvora razine n jednaki n+1 (čvor f je razine 3)
  - Visina stabla je jednaka maksimalnoj popunjenoj razini čvorova u stablu (visina stabla je 4)
  - Visina se nekad zove i dubina (što je možda i logičnije, jer je ovim stablima korijen na vrhu)

- Primjeri za stabla:
  - sadržaj knjige
  - struktura državne uprave
  - obiteljsko stablo
  - datoteke u računalu



- Aritmetički izraz se može prikazati stablom: čvorovi bez djece predstavljaju operande, a ostali čvorovi računske operacije. Linearna uređenost stabla je važna ako su operacije nekomutativne.
- Npr. izraz

$$(a + b) * (a + \sqrt{c})$$



- Stablo iz zadnjeg primjera je označeno stablo: svakom čvoru je pridružen dodatni podatak – oznaka, različita od imena čvora
- Ime čvora služi kao identifikacija pa je jedinstveno (nema dva čvora s istim imenom)
- Oznaka čvora služi kao informaciju (dva čvora mogu imati istu oznaku)
- Ovdje navedeni pojmovi stablo, oznaka, čvor su analogni pojmovima lista, element, pozicija iz opisa liste
- Put od i_1 do i_m je niz čvorova $i_1, i_2, \ldots, i_m$ takvih da je i_p roditelj od i_{p+1} ($p=1, 2, \ldots, m-1$), duljina puta je jednaka $m-1$
- Za svaki čvor postoji jedinstveni put od korijena do njega
- Ako postoji put od čvora i do čvora j, onda je i predak od j, a j potomak od i
- Korijen je predak svih čvorova u stablu, a svi čvorovi su njegovi potomci
- Razina s je skup svih čvorova za koje put od korijena do tog čvora ima duljinu s
- Razina 0 je korijen

- Sve ovo su bila svojstva matematičkog objekta koji se naziva stablo, a nas će zanimati nešto praktičnija implementacija apstraktnog tipa podataka stablo

Apstraktni tip podataka TREE

node – bilo koji tip, ime čvora. Postoji poseban element koji služi kao ime nepostojećeg čvora, označavamo ga LAMBDA

labeltype – bilo koji tip, oznaka čvora (to su podaci koje pohranjujemo u stablo)

TREE – podatak ovog tipa je uređeno stablo čiji čvorovi su podaci tipa node, međusobno različiti i različiti od LAMBDA. Svakom čvoru se kao oznaka pridružuje podatak tipa labeltype

MAKE_ROOT(l,&T) – funkcija pretvara stablo T u stablo koje se sastoji samo od korijena s oznakom l. Vraća ime čvora koji služi kao korijen.

INSERT_CHILD(l,i,&T) – funkcija u stablo T ubacuje novi čvor s oznakom l, tako da on bude prvo dijete čvora i, vraća novi čvor. Nedefinirana ako i ne pripada T.

INSERT_SIBLING(l,i,&T) - funkcija u stablo T ubacuje novi čvor s oznakom l, tako da on bude idući po redu brat čvora i, vraća novi čvor. Nedefinirana ako je i korijen, ili ako nije u T

DELETE(i,&T) – funkcija izbacuje list i iz stabla T. Nedefinirana ako je i korijen, ako nije u T, ili ako ima djece (nije list)

ROOT(T) – funkcija vraća korijen stabla T

FIRST_CHILD(i,T) – funkcija vraća prvo po redu dijete čvora i u T. Ako je i list, vraća LAMBDA. Nedefinirana ako i nije u T.

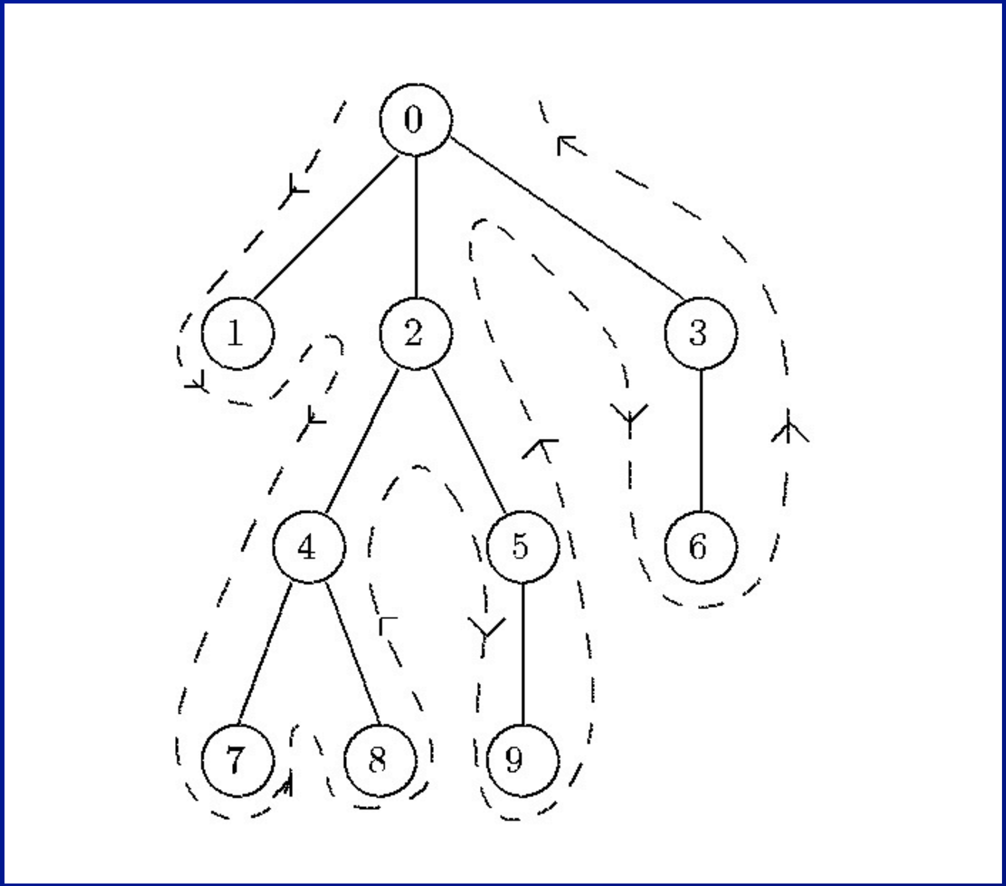
- NEXT_SIBLING(i,T) – funkcija vraća idućeg po redu brata čvora i u T. Ako je i zadnji brat vraća LAMBDA, nedefinirana ako i nije u T
- PARENT(i,T) – funkcija vraća roditelja čvora i u T. Ako je i korijen vraća LAMBDA. Nedefinirana ako i nije u T.
- LABEL(i,T) – funkcija vraća oznaku čvora i u T. Nedefinirana ako i nije u T.
- CHANGE_LABEL(l,i,&T) – funkcija mijenja oznaku čvora i iz T u l. Nedefinirana ako i nije u T.

Obilazak stabla

- Obilazak stabla je algoritam kojim posjećujemo čvorove stabla tako da svaki čvor posjetimo točno jednom
- Potrebno za obradu nad svim čvorovima
- Svaki obilazak uspostavlja linearno uređivanje među čvorovima
- Najčešći obilasci su: PREORDER(), INORDER(), POSTORDER()
- Zadaju se rekurzivno
- Ako je T stablo sastavljeno od korijena r i podstabala T_1, T_2,…,T_k tada:
  - PREORDER() … najprije posjećuje r, zatim obilazi T_1, pa T_2,…, na kraju obilazi T_k
  - INORDER() … najprije obilazi T_1, zatim posjećuje r, obilazi T_2,…, na kraju obilazi T_k
  - POSTORDER() … najprije obilazi T_1, zatim obilazi T_2,…, zatim obilazi T_k , na kraju posjećuje r

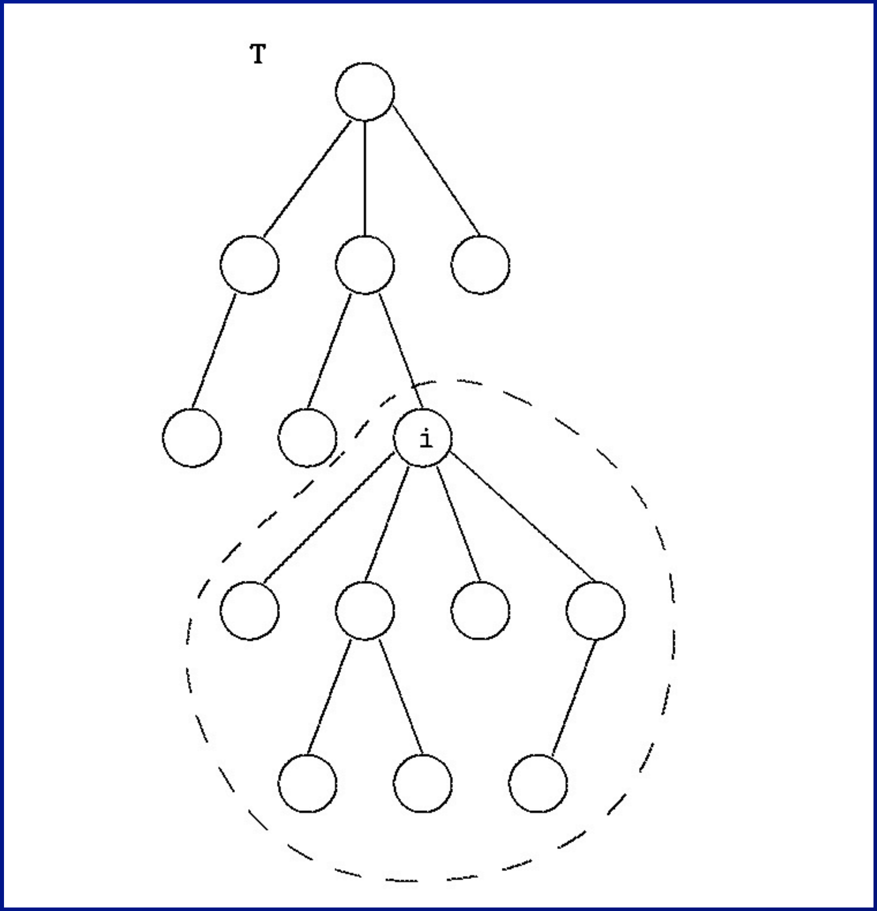
- Primjer: čvorove stabala sa slike algoritmi obilaze u sljedećem redoslijedu:

PREORDER(): 0, 1, 2, 4, 7, 8, 5, 9, 3, 6
INORDER(): 1, 0, 7, 4, 8, 2, 9, 5, 6, 3
POSTORDER(): 1, 7, 8, 4, 9, 5, 2, 6, 3, 0



- U apstraktnom tipu podataka TREE algoritmi obilaska se mogu pisati kao potprogrami. Primjer gdje je operacija posjećivanje čvora realizirana ispisom oznake čvora (obilazi se podstablo od T kojeg čini čvor i s potomcima):

```
void PREORDER(node i, TREE T) {
 node c; // Što treba pisati umjesto node?
 printf(format, LABEL(i,T)); // Što je format?
 c = FIRST_CHILD(i,T);
 while ( c != LAMBDA) { // Što je LAMBDA?
    PREORDER(c,T);
    c = NEXT_SIBLING(c,T);
   }
}
```



Implementacija stabla na osnovu veze čvor → roditelj

- Zasniva se na tome da svakom čvoru eksplicitno zapišemo njegovog roditelja
- Moguće razne varijante zbog raznih prikaza skupa čvorova
- Uzimamo za imena čvorova cijele brojeve 0, 1, 2, ..., n-1 gdje je n broj čvorova
- Stablo se prikazuje poljima, i-te ćelije polja opisuju i-ti čvor i u njima piše oznaka tog čvora, odnosno kursor na roditelja

```
#define MAXNODES ... // Umjesto točkica upišemo max broj čvorova. Kako se to zove?
#define LAMBDA -1
typedef int node;       // Čvorovi su integeri
typedef struct {
    node root;
    labeltype label[MAXNODES]; // Oznaka (labela) je zapravo sadržaj čvora
    node parent[MAXNODES];    // Čvorovi roditelja
} TREE;
```

Oznaka (željeni tip)

Oznaka (željeni tip)

Ime (integer)

Ime (integer)

root
0

	label	parent
0	a	-1
1	b	0
2	c	0
3	d	1
4	e	1
5	a	4
6	d	4
7	e	4
8	c	2
9	b	2

Kursor root pokazuje gdje je korijen stabla

Ako je MAXNODES veći od stvarnog broja čvorova, neke ćelije će ostati slobodne. One se mogu označiti tako da im se upiše nemoguća vrijednost (-1)

- Opisana struktura dobro podržava operacije PARENT() i LABEL() // Zašto?
- Ostale operacije zahtijevaju pretraživanje cijelog polja
- Mana je da se ne pamti redoslijed braće – ovakvo stablo je zapravo neuređeno
- Moguće uvesti dodatno pravilo da su braća poredana po svojim imenima (indeksima). Tada vrijedi (u polju tražimo ćeliju nakon i-te u kojoj je upisan isti roditelj):

```
node NEXT_SIBLING(node i, TREE T) {
 node j, p;
 p = T.parent[i];
 for (j = i + 1; j < MAXNODES; j++)
      if (T.parent[j] == p) return j;
return LAMBDA; /* ne postoji idući brat */
}
```

Ova implementacija je dobra ako nema mnogo ubacivanja/izbacivanja čvorova, nije potrebna uređenost stabla i pretežno se koriste operacije PARENT() i LABEL().

Implementacija stabla na osnovu veze čvor → (prvo dijete, idući brat)

- Svakom čvoru se eksplicitno zapiše njegovo prvo dijete te njegov idući brat
- Veza od čvora do djeteta ili brata se može realizirati pomoću pokazivača ili pomoću kursora
- Pogledajmo varijantu s kursorima gdje su imena čvorova cijeli brojevi

```
#define MAXNODES ...

typedef int node;
typedef struct {
   labeltype label;
   node first_child, next_sibling;
} node_struct;
```

memoriju zauzimamo globalnim poljem SPACE[] koje je zaliha ćelija od kojih se grade stabla

i-ta ćelija opisuje i-ti čvor

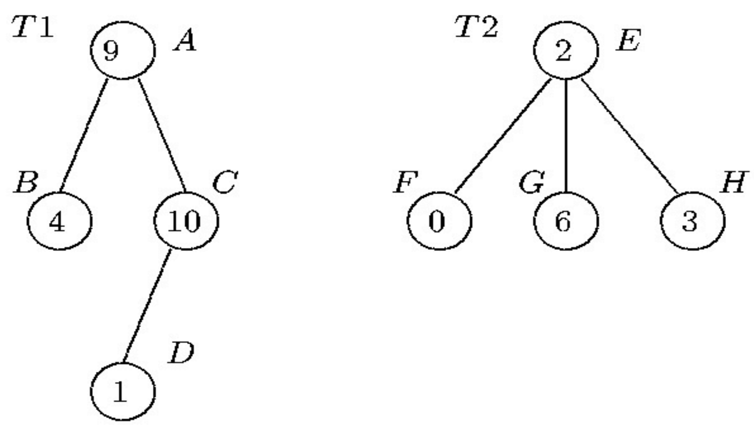
stablo prikazano kao vezana struktura ćelija

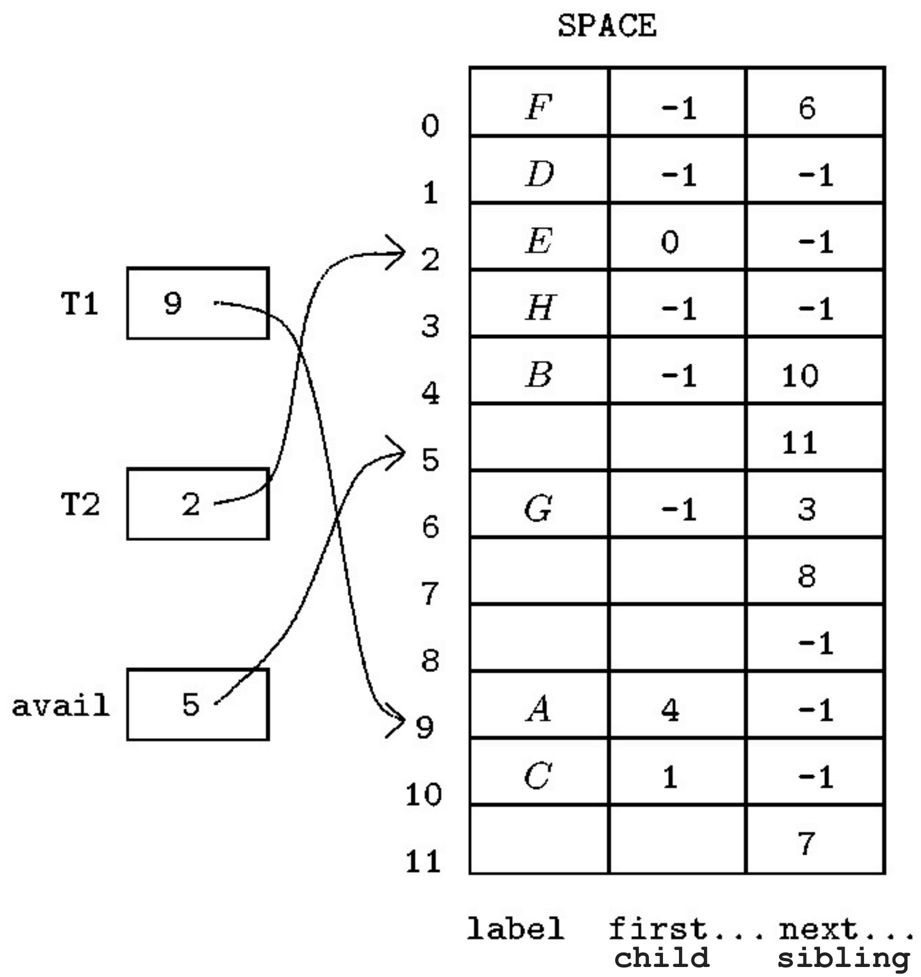
stablo se identificira s kursorom na korijen:

```
typedef int TREE;
```

```
node_struct SPACE[MAXNODES]; // MAXNODES je max broj čvorova u svim stablima
```

- Razna stabla koriste ćelije iz istog polja SPACE[]. Sve slobodne ćelije povezane su u vezanu listu čiji poredak pokazuje globalni kursor avail. Slobodne ćelije se vežu kursorima smještenima npr. u komponenti next_sibling. Sve operacije osim PARENT() se mogu efikasno implementirati.





	label	first... child	next... sibling
0	F	-1	6
1	D	-1	-1
2	E	0	-1
3	H	-1	-1
4	B	-1	10
5			11
6	G	-1	3
7			8
8			-1
9	A	4	-1
10	C	1	-1
11			7

Ukoliko je potrebna i operacija PARENT(), u ćeliju polja SPACE[] može se dodati i kursor prema roditelju.

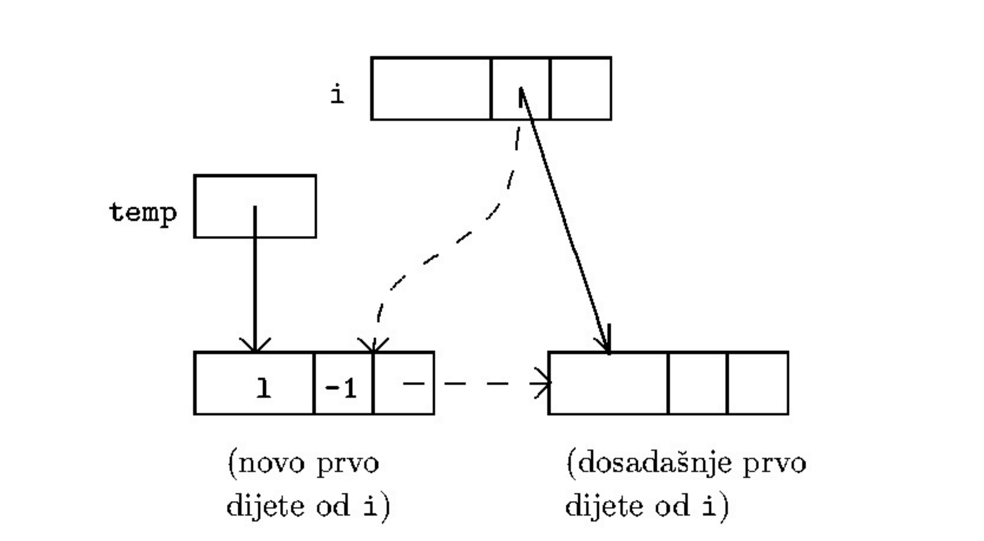
Pitanja:

1) Mogu li biti dva elementa s imenom B?

2) Mogu li biti dva elementa s čvorom 10?

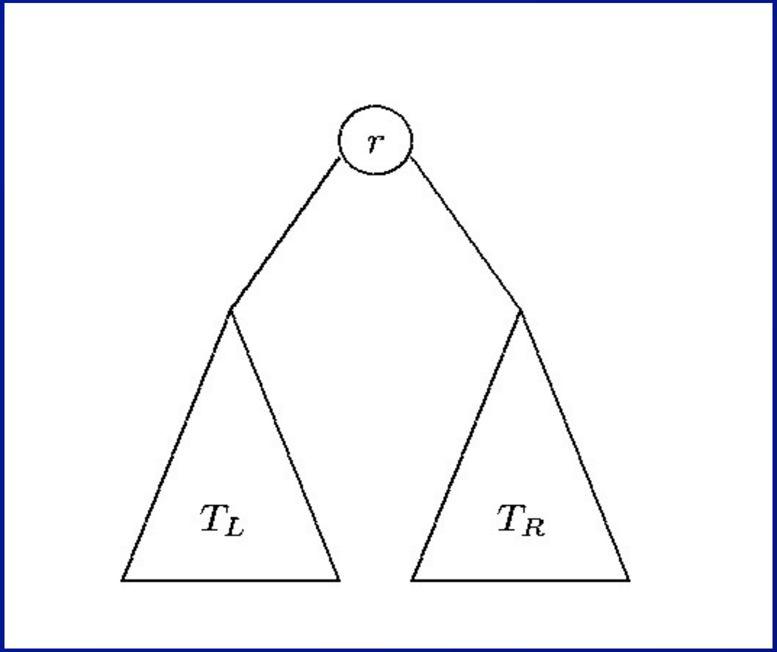
```
node INSERT_CHILD(labeltype l, node i) {
node temp;
if (avail == -1)
     printf("Nema više slobodnih mjesta"); //avail je globalna varijabla
else {
     temp = avail;// Privremeni čvor je postavljen na prvo prazno mjesto (avail)
     avail = SPACE[avail].next_sibling; // Pomaknemo avail jedno mjesto dalje
     SPACE[temp].label = l; // Postavljamo oznaku novog čvora
     SPACE[temp].first_child = -1; // Novi čvor (još) nema djece
     SPACE[temp].next_sibling = SPACE[i].first_child; /* Ranije prvo dijete od
  čvora i postaje prvi brat novog djeteta */
     SPACE[i].first_child = temp; // Novo dijete postaje prvi brat novog djeteta
     return temp;
   }
}
```

Ova implementacija pogodna kada ima puno ubacivanja/izbacivanja čvorova ili kad se više stabala spaja u veće ili za intenzivnu uporabu veza roditelj → dijete



Binarno stablo

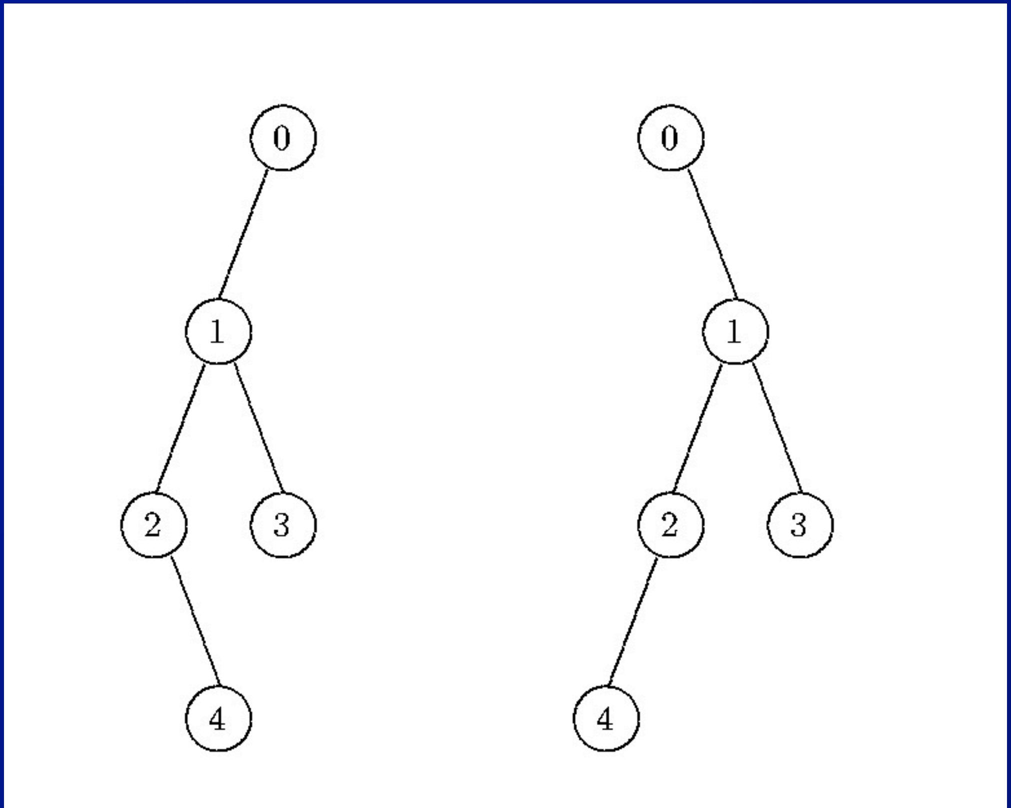
- Primjer: kao dioba stanice (jedan roditelj, dva dijeteta)
- Stabla kakva smo razmatrali do sada češća su u matematici, dok se u računarstvu više koriste binarna stabla
- To je nešto pravilniji objekt kojeg je jednostavno prikazati računalom
- Definicija: binarno stablo T je konačan skup podataka istog tipa koje zovemo čvorovi, pri čemu T može biti prazan skup (prazno stablo) ili ima istaknut čvor r koji se zove korijen od T, dok ostali čvorovi grade uređeni par (T_L,T_R) disjunktnih manjih binarnih stabala



- Ako T sadrži korijen r, tada se binarna stabla T_L i T_R zovu lijevo i desno podstablo
- Korijen od T_L (ako postoji) je lijevo dijete od r (T_R desno dijete), a r je njihov roditelj
- Ostala terminologija ista kao kod stabala, primjenjuju se i isti algoritmi obilaska
- Važno je da binarno stablo nije specijalan slučaj uređenog stabla, jer binarno stablo može biti prazno, te ako čvor ima samo jedno dijete kod binarnog stabla je važno da li je ono lijevo ili desno

Primjer: 2 različita binarna stabla

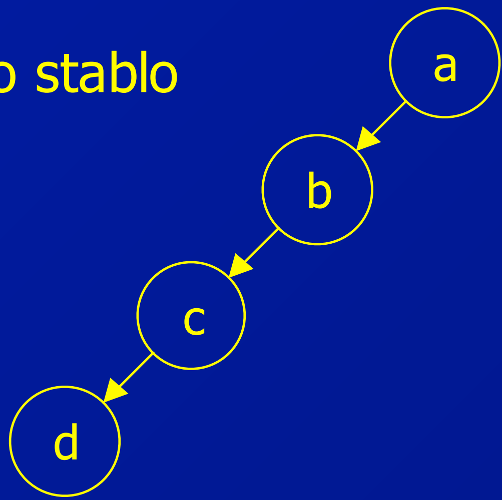
Da su ovo "obična" uređena stabla, bila bi identična.



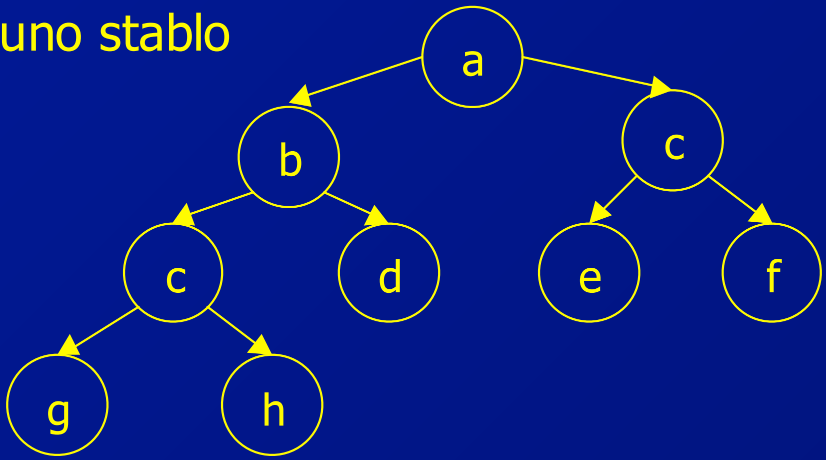
- Dakle, binarno stablo je stablo koje se sastoji od nijednog, jednog ili više čvorova drugog stupnja. Kod binarnog stabla razlikujemo lijevo i desno podstablo svakog čvora.

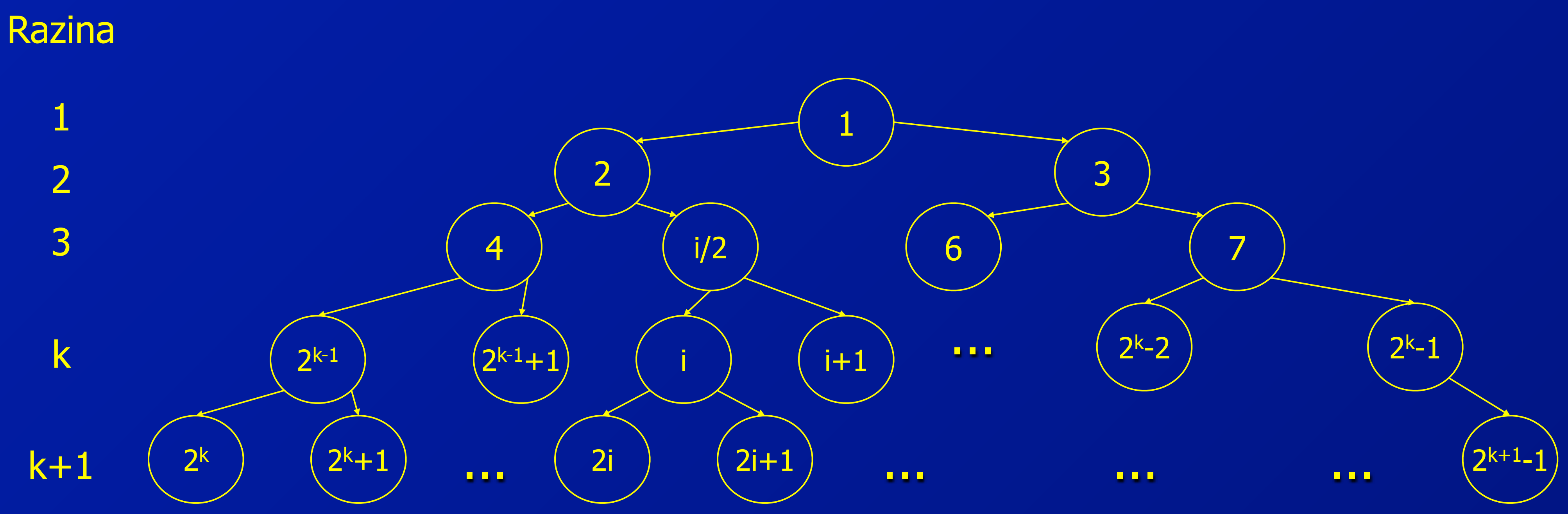
- Primjeri: binarnih stabala:

Koso stablo



Potpuno stablo

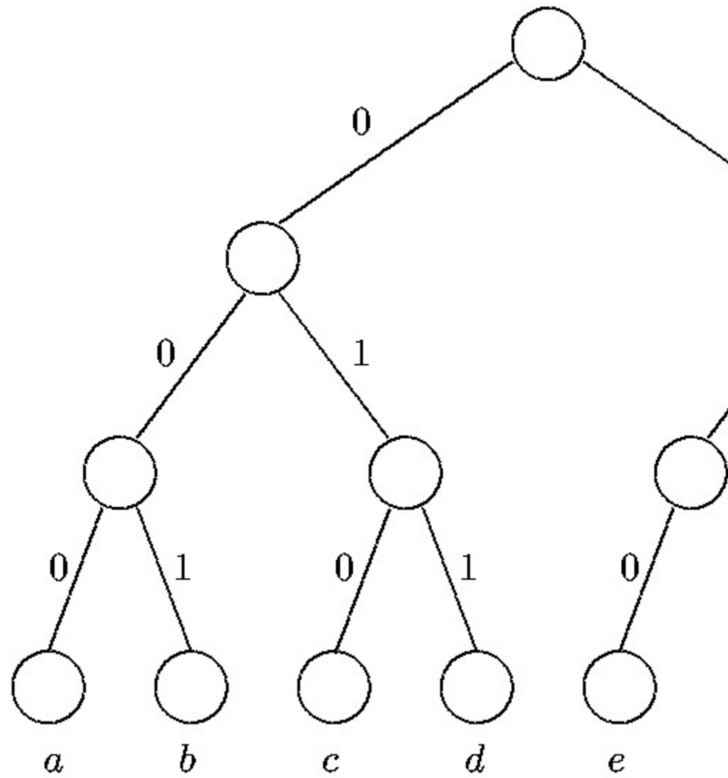




- Iz definicije binarnog stabla slijedi da je:
  - maksimalni broj čvorova na k-toj razini jednak je 2^{k-1}
  - maksimalni broj čvorova binarnog stabla visine k jednak je 2^k -1 za $k>0$
  - Stablo koje je visine k i ima 2^k -1 elemenata naziva se **puno binarno stablo**. Binarno stablo s n čvorova dubine k je **potpuno** ako i samo ako njegovi čvorovi odgovaraju čvorovima punog binarnog stabla dubine k koji su numerirani od 1 do n.
  - Posljedica je u tome da je razlika razina krajnjih čvorova potpunog stabla najviše jedan.

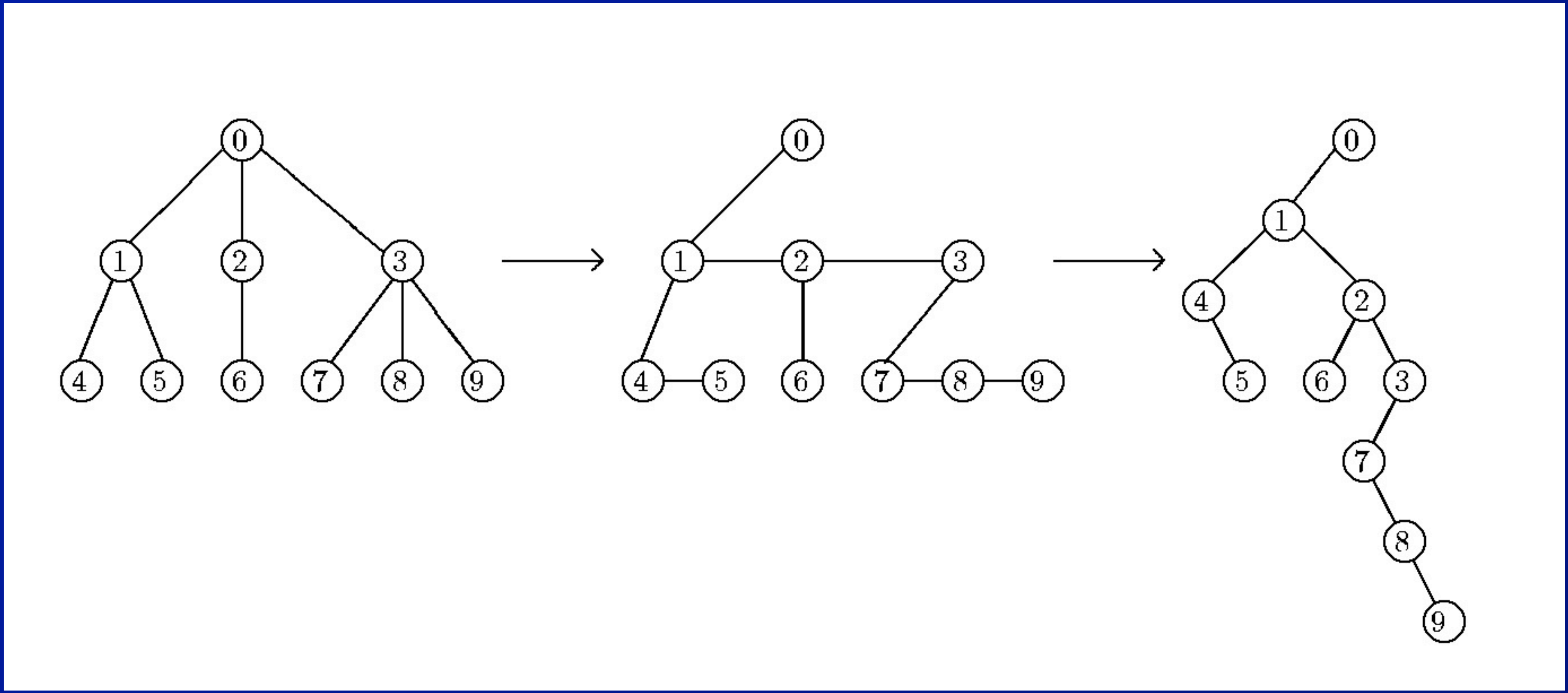
- Primjeri binarnih stabala:
- Ako je aritmetički izraz sastavljen od binarnih operacija tada se njegova građa može prikazati binarnim stablom
- Znakovi su kodirani nizom bitova; postupak dekodiranja se može prikazati binarnim stablom:

znak	kod
a	000
b	001
c	010
d	011
e	100



Je li ovo potpuno binarno stablo?

- Bilo koje uređeno stablo se može interpretirati kao binarno stablo pomoću veza čvor→prvo_dijete, te čvor→idući_brat



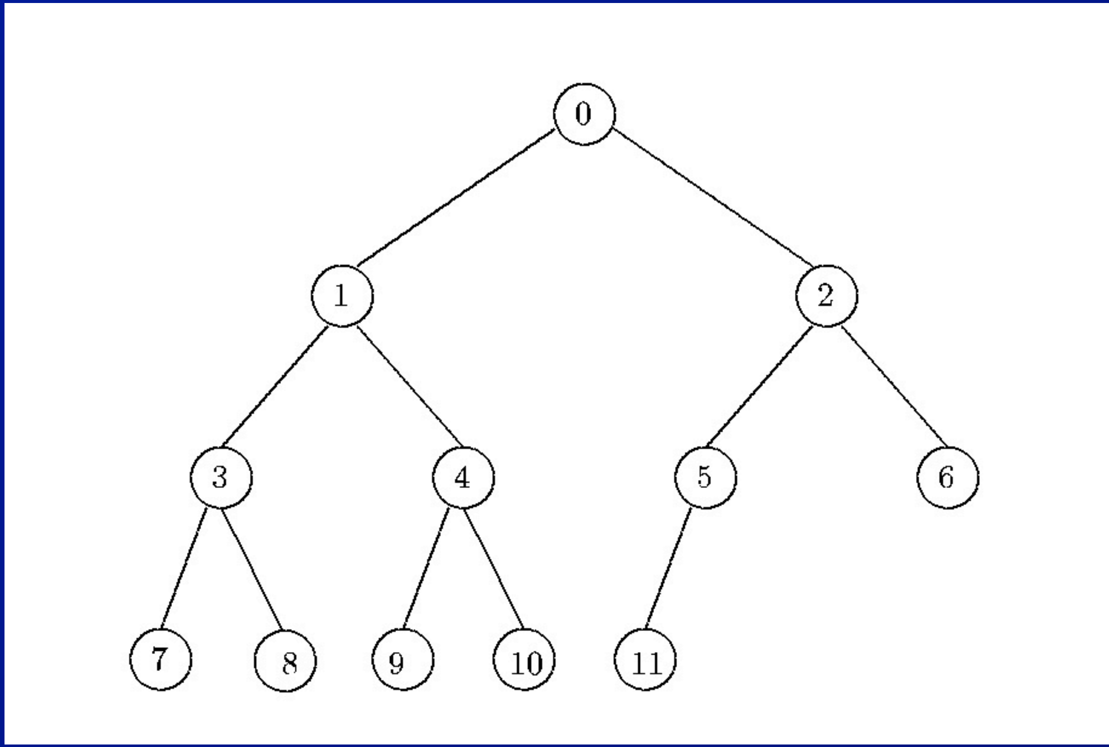
- Ova pretvorba je već korištena kod ranije implementacije: čvor→(prvo_dijete, idući_brat)

Potpuno binarno stablo

- Potpuno binarno stablo je građeno od n čvorova, s imenima 0, 1, 2, …, n-1. Pritom vrijedi:
  - lijevo dijete čvora i je čvor 2i+1 (ako je 2i+1 > n-1 tada čvor nema lijevo dijete)
  - desno dijete čvora i je čvor 2i+2 (ako je 2i+2 > n-1 tada i nema desno dijete)

Primjer: potpuno binarno stablo s n = 12 izgleda ovako

Na svim nivoima osim zadnjeg postoje svi mogući čvorovi
Čvorovi na zadnjem nivou kreću s lijeve strane.
Numeriranje ide od razine 0 na razinu 1, razinu 2, itd. s lijeva na desno.
Ovo je objekt sa statičkom građom - ne stvaraju se novi čvorovi i podstabla jer rezultat više ne bi bio potpuno binarno stablo
(samo na desnom kraju zadnjeg nivoa moguće ubacivanje/ izbacivanje čvorova)



Apstraktni tip podataka BTREE

- Također se može definirati na razne načine kao ATP
- Operacije na nivou čvorova i nivou podstabala
- Ovdje jedan primjer s opširnim popisom operacija

node … bilo koji tip, imena čvorova. Postoji poseban element LAMBDA koji označava nepostojeći čvor

labeltype … bilo koji tip, oznaka čvora

BTREE … podatak ovog tipa je binarno stablo čiji čvorovi su podaci tipa node, međusobno različiti i različiti od LAMBDA. Svakom čvoru je kao oznaka pridružen podatak tipa labeltype

MAKE_NULL(&T) … funkcija pretvara T u prazno binarno stablo

EMPTY(T) … funkcija vraća istinu ako je T prazno binarno stablo

CREATE(l,TL,TR,&T) … funkcija stvara novo binarno stablo T kojem je lijevo podstablo TL, a desno TR. TL i TR moraju biti disjunktni. Korijen od T dobiva oznaku l.

LEFT_SUBTREE(T,&TL), RIGHT_SUBTREE(T,&TR) … funkcija preko parametra TL (TR) vraća lijevo(desno) podstablo binarnog stabla T. Nedefinirana za prazan T.

INSERT_LEFT_CHILD(l,i,&T), INSERT_RIGHT_CHILD(l,i,&T) … funkcija u binarno stablo T ubacuje novi čvor s oznakom l, tako da on bude lijevo (desno) dijete čvora i. Vraća novi čvor. Nedefinirana ako i nije u T, ili ako i već ima to dijete.

DELETE(i,&T) … funkcija izbacuje list i iz binarnog stabla T. Nedefinirana ako i nije u T ili ako i ima dijete.

ROOT(T) … funkcija vraća korijen od T. Za prazan T, vraća LAMBDA.

LEFT_CHILD(i,T), RIGHT_CHILD(i,T) … funkcija vraća lijevo (desno) dijete čvora i iz T. Ako dijete ne postoji vraća LAMBDA. Nedefinirana ako i nije u T.

PARENT(i,T) … funkcija vraća roditelja čvora i iz T. Ako je i korijen vraća LAMBDA. Nedefinirana za i koji nije u T.

LABEL(i,T) … funkcija vraća oznaku čvora i u binarnom stablu T. Nedefinirana ako i nije u T.

CHANGE_LABEL(l,i,&T) … funkcija mijenja oznaku čvora i iz T tako da ona postane l. Nedefinirana ako i nije u T.

Prikaz binarnog stabla statičkom strukturom polje

- Potpuno se binarno stablo jednostavno prikazuje jednodimenzionalnim poljem, bez podataka za povezivanje, i koristi se pravilima za određivanje odnosa u stablu. Ako korištenje polja počinje od člana s indeksom 1, veze za i-ti čvor su:
  - roditelj(i) = $\lfloor i/2 \rfloor$ za $i \neq 1$; kada je $i = 1$, čvor i je korijen pa nema roditelja
  - lijevo_dijete(i) = $2*i$ ako je $2*i \le n$; kad je $2*i > n$ čvor i nema lijevog djeteta
  - desno_dijete(i) = $2*i + 1$ ako je $2*i+1 \le n$; kad je $2*i+1 > n$ čvor i nema desnog djeteta
  - Ovako se mogu prikazati sva binarna stabla, ali se tada memorija ne koristi efikasno. Najgori slučaj su kosa stabla koja koriste smo k lokacija od 2^k -1 lokacija predviđenih za to stablo.

Koso stablo	a	b		c				d								e
Potpuno stablo	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p

- Problem kod prikaza stabla statičkom strukturom polje je i teško umetanje i brisanje čvorova jer ti zahtjevi mogu tražiti pomicanje puno elemenata.

Implementacija potpunog binarnog stabla pomoću polja

- i-ta ćelija polja sadrži oznaku čvora i, postoji kursor koji pokazuje na zadnji čvor n-1

```
#define MAXNODE …
typedef int node;
typedef struct {
    int last;
    labeltype labels[MAXNODE];
} BTREE;
```

Manevriranje po stablu (drugi način izvedbe): korijen predstavljen 0-tom ćelijom, lijevo i desno dijete čvora iz i-te ćelije se nalaze u ćelijama 2i+1 i 2i+2 (ako one postoje). Roditelj čvora iz i-te ćelije je u (i-1)/2 ćeliji

Prikaz binarnog stabla dinamičkom strukturom – implementacija pomoću pokazivača

- Svakom čvoru se eksplicitno zapiše njegovo lijevo i desno dijete, a po potrebi se može dodati i pokazivač na roditelja

```
typedef struct cell_tag {
        labeltype label;
        struct cell_tag *leftchild;
        struct cell_tag *rightchild;
        struct cell_type *parent;   /* kada je potrebna i veza s roditeljom */
  } celltype;

typedef celltype *node;
typedef celltype *BTREE;
```

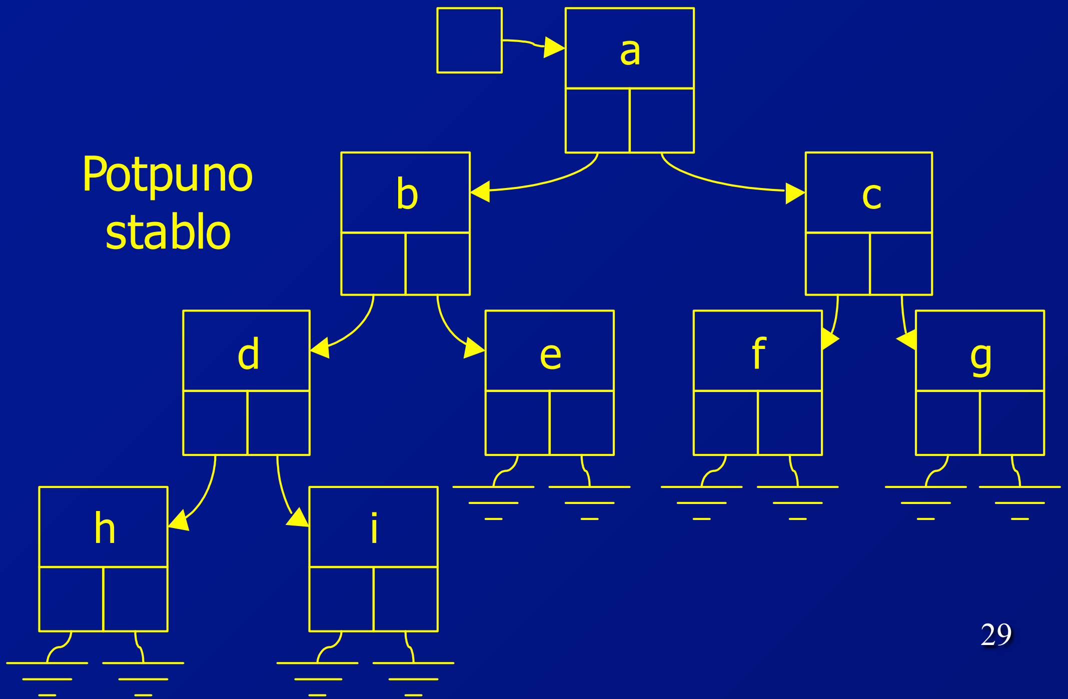
Svaki se čvor prikaže jednom ćelijom, pa je čvor jednoznačno određen pokazivačem na tu ćeliju. Binarno stablo se poistovjećuje s pokazivačem na korijen. Prazno stablo se prikazuje pokazivačem NULL.

- Sve operacije iz ATP BTREE se mogu efikasno izvesti uz konstantno vrijeme izvršavanja

```
void CREATE (labeltype l, BTREE TL, BTREE TR, BTREE *Tptr) {
       *Tptr = (celltype*) malloc(sizeof(celltype));// Što je celltype?
      (*Tptr)->label = l;
      (*Tptr)->leftchild = TL;
      (*Tptr)->rightchild = TR;
}
```

Koso stablo

Potpuno stablo



K-stabla, stablo traženja (sortirano stablo)

- Prirodna generalizacija binarnih stabala su k-stabla, gdje k predstavlja stupanj stabla, $k>2$, sa istim mogućnostima prikazivanja. Općenita stabla, s raznim stupnjevima, se mogu transformirati u binarna stabla što rezultira manjim i efikasnijim algoritmima, te manjom potrošnjom memorije.

- Može se oblikovati stablo za traženje (sortirano, uređeno stablo) po nekom od podataka (ključu) koji se upisuju u pojedini čvor. Upis novog čvora počinje pretragom od korijena stabla. Uspoređuje se već upisani podatak u čvorovima s novim podatkom:
  - ako je ključ novog čvora manji od ključa upisanog čvora usporedbe, nastavlja se usporedba u lijevom podstablu.
  - ako je ključ novog čvora veći ili jednak od ključa upisanog čvora usporedbe, nastavlja se usporedba u desnom podstablu.
  - ako upisani čvor nema podstablo u traženom smjeru, novi čvor postaje dijete upisanog čvora.

Domaća zadaća

- (1) Kolika je vjerojatnost da iz tri bacanja kocke dobijemo barem jednu šesticu? Jeste li sigurni?
- (2) Napišite program koji empirijski provjerava (za n=10, 100 i 1000 bacanja) vjerojatnosti svih mogućih ishoda bacanja dvije kocke te ispisuje očekivane (teorijske) i dobivene rezultate.
- Vidi vježbe za više detalja...