

# Baze podataka

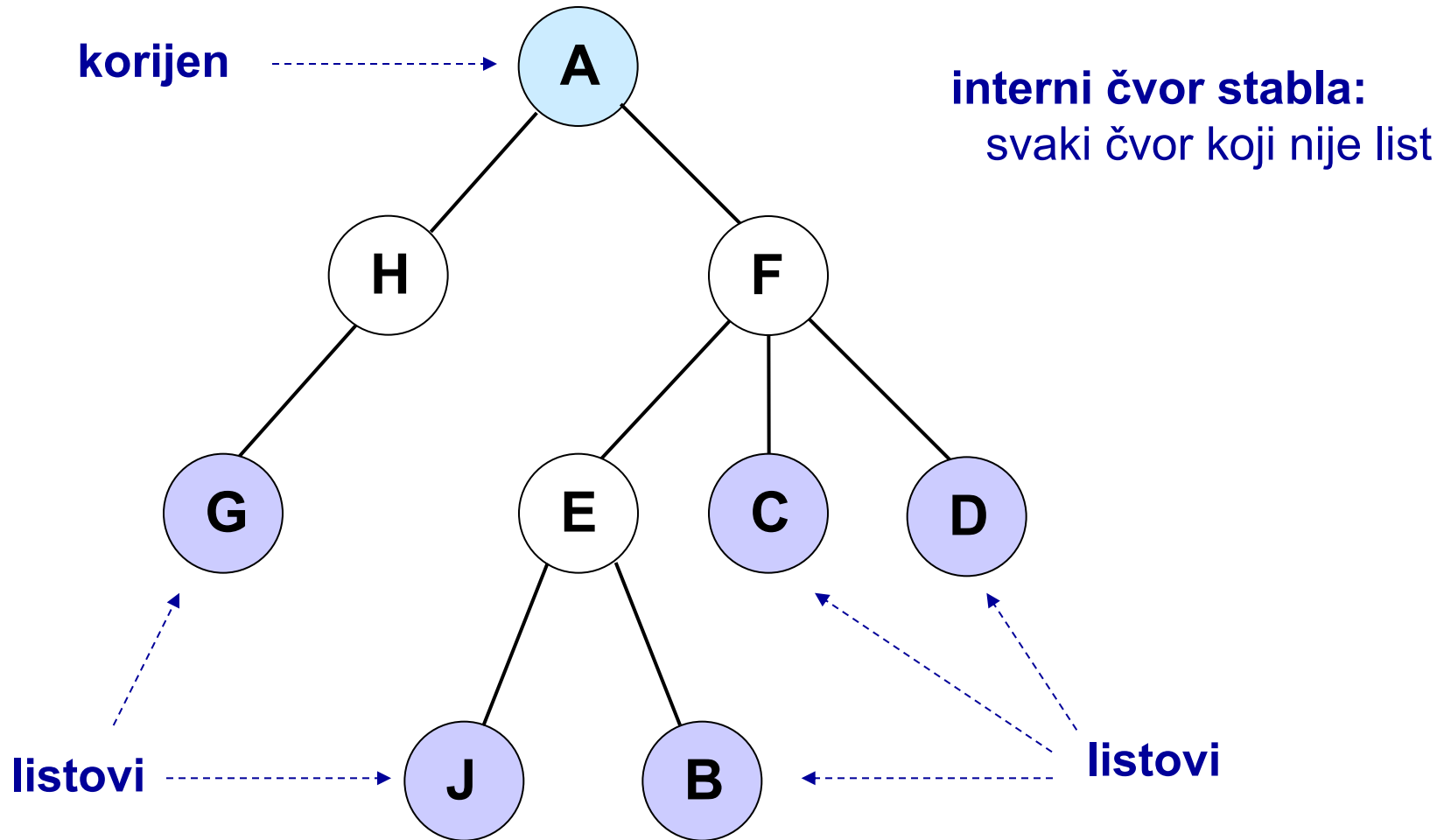
## Dopunski materijali (za one koji žele znati više)

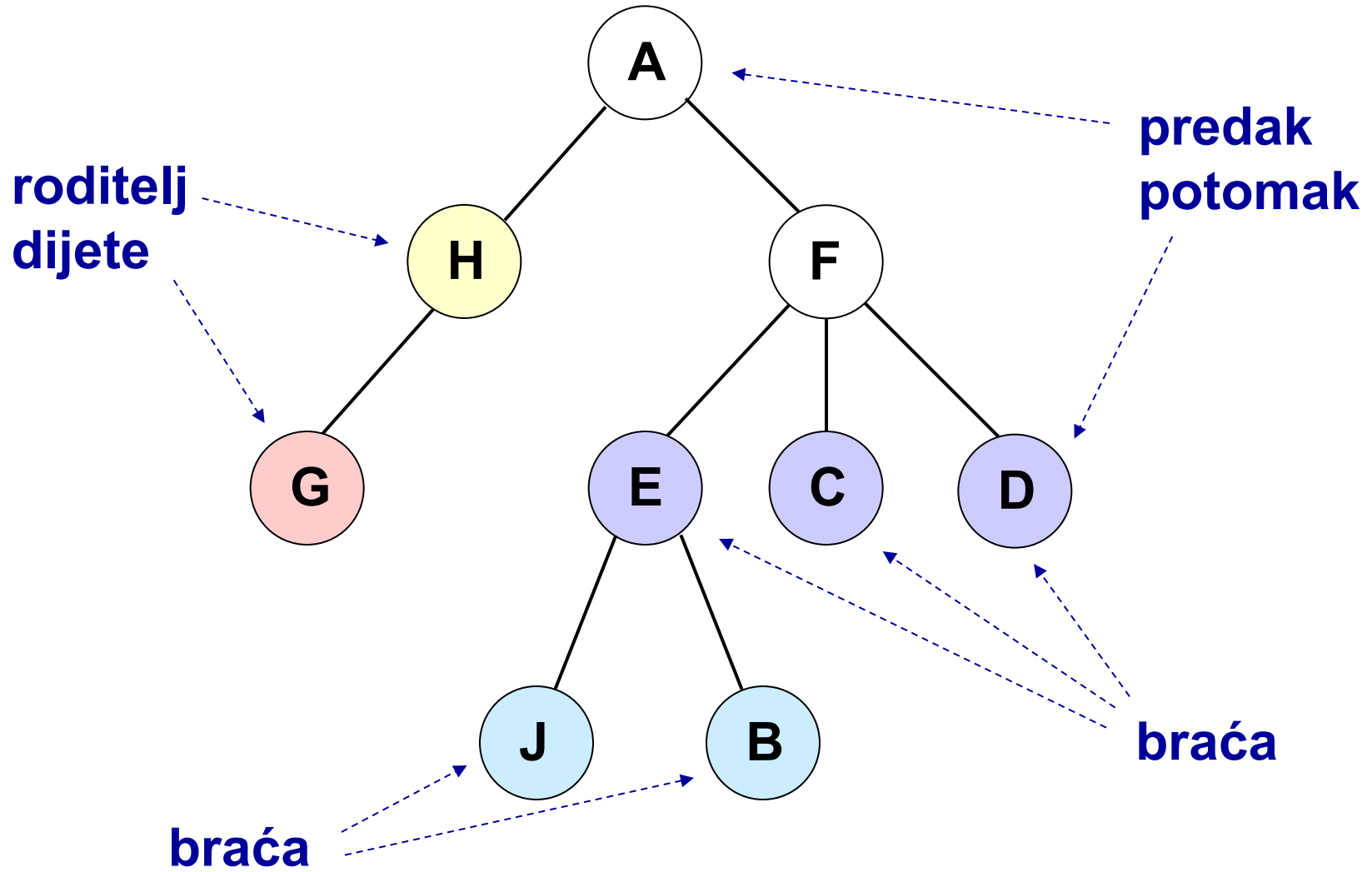
## B - stabla

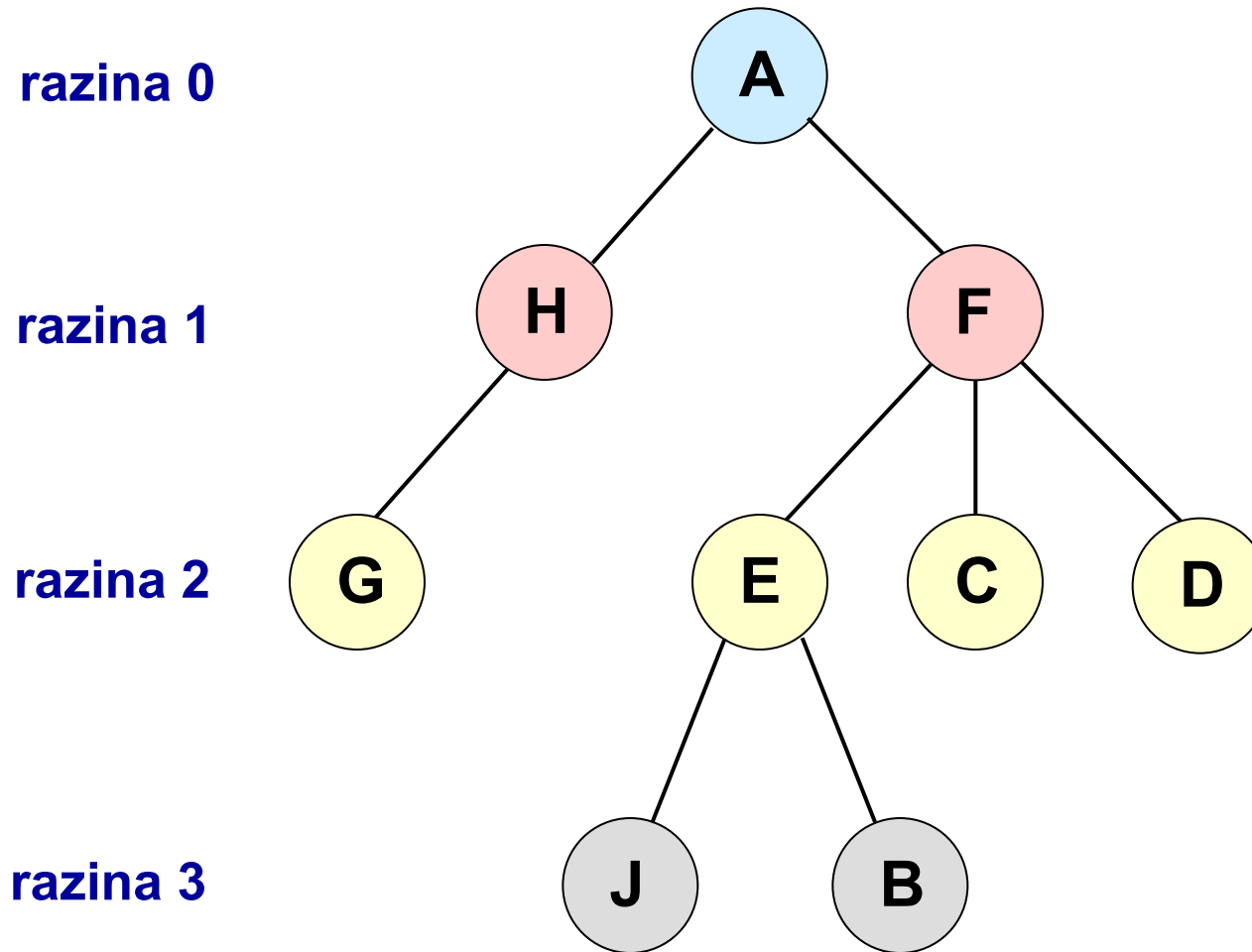
# Travanj, 2021.



# 1. Stablo kao struktura podataka







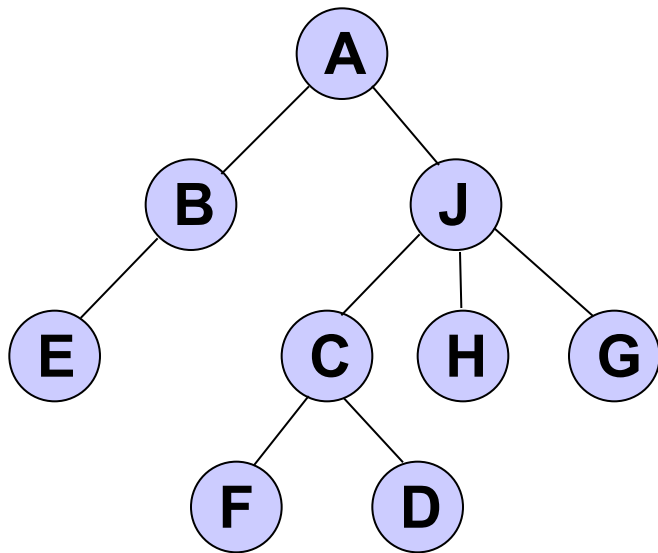
**razina čvora (*level*):** duljina puta od korijena do čvora

**dubina stabla (*depth*):** najveća duljina puta od korijena do lista

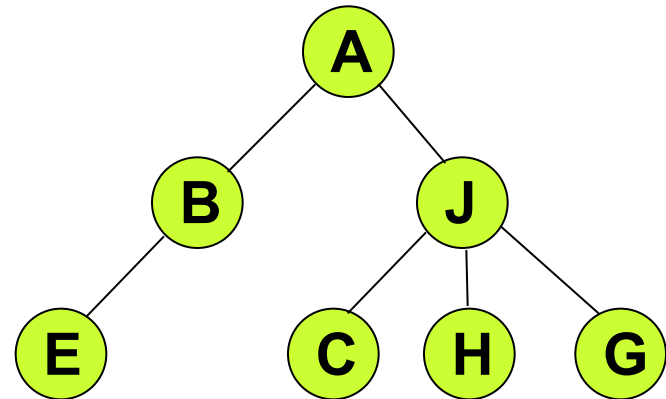
**red stabla (*order*):** najveći broj djece koje čvor može imati

Stablo je **balansirano** (*balanced*) ukoliko je duljina puta od korijena do lista jednaka za svaki list u stablu

stablo nije balansirano



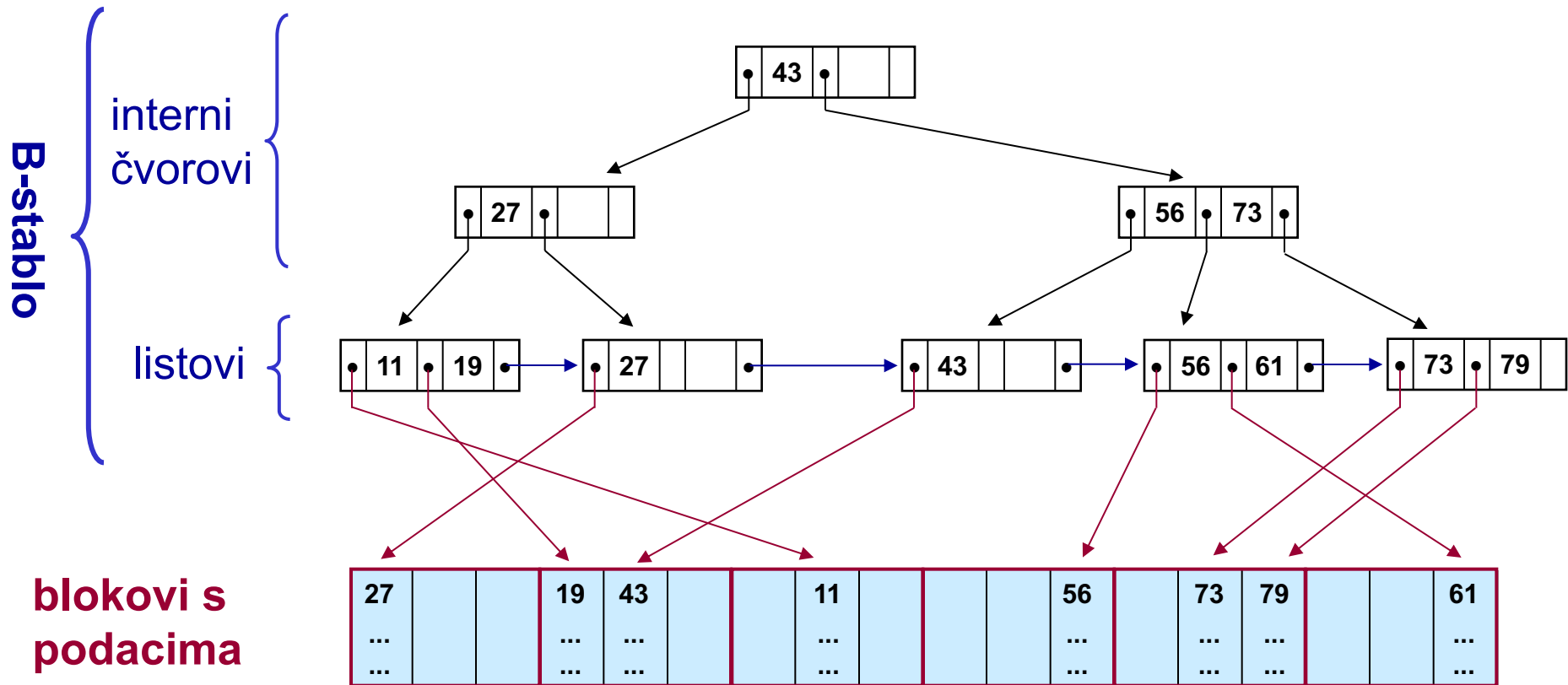
stablo je balansirano



Oznaka **B** u B-stablo znači "**balansirano**"!

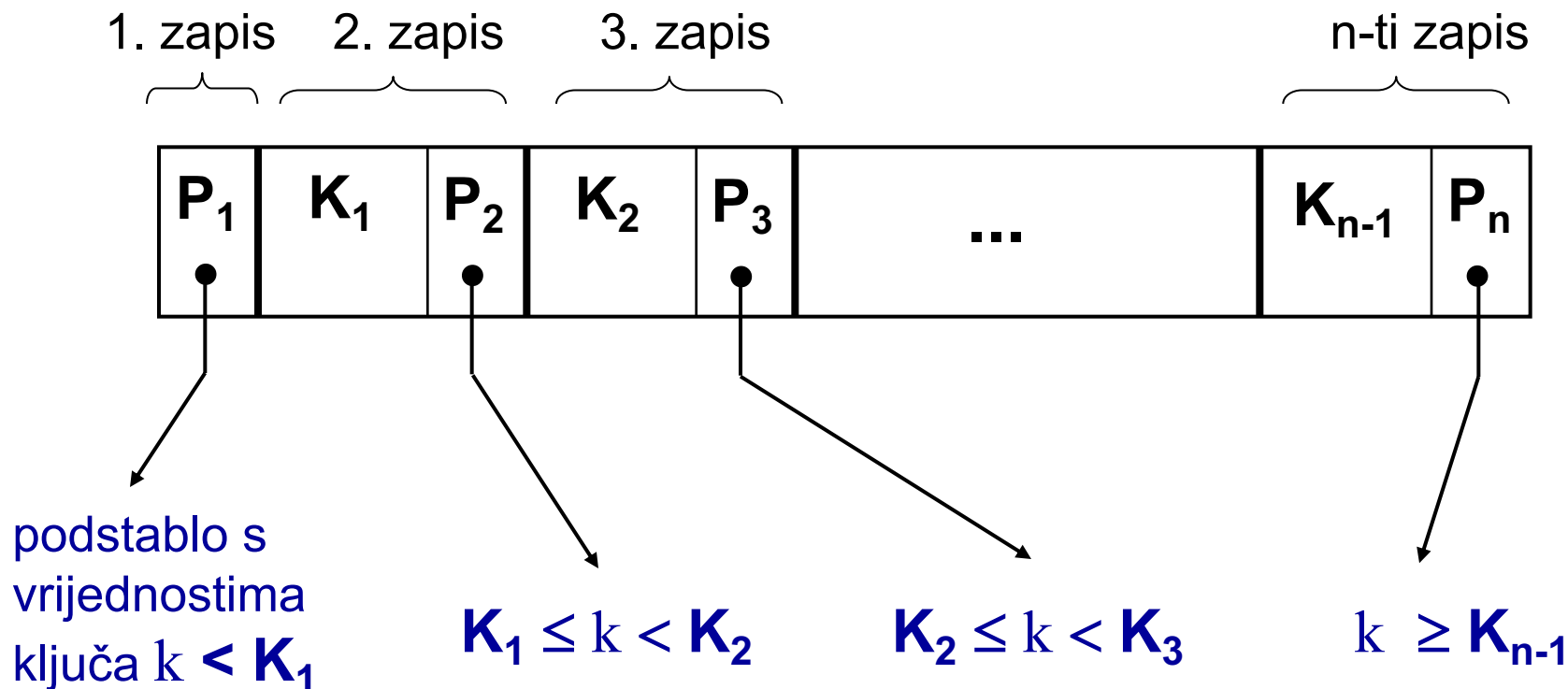
## 2. Struktura B-stabla

Opisujemo varijantu B-stabla koja se naziva **B<sup>+</sup>-stablo**. Opisi ostalih varijanti B-stabala (B\*-stablo, B-stablo, ...) mogu se pronaći u literaturi.



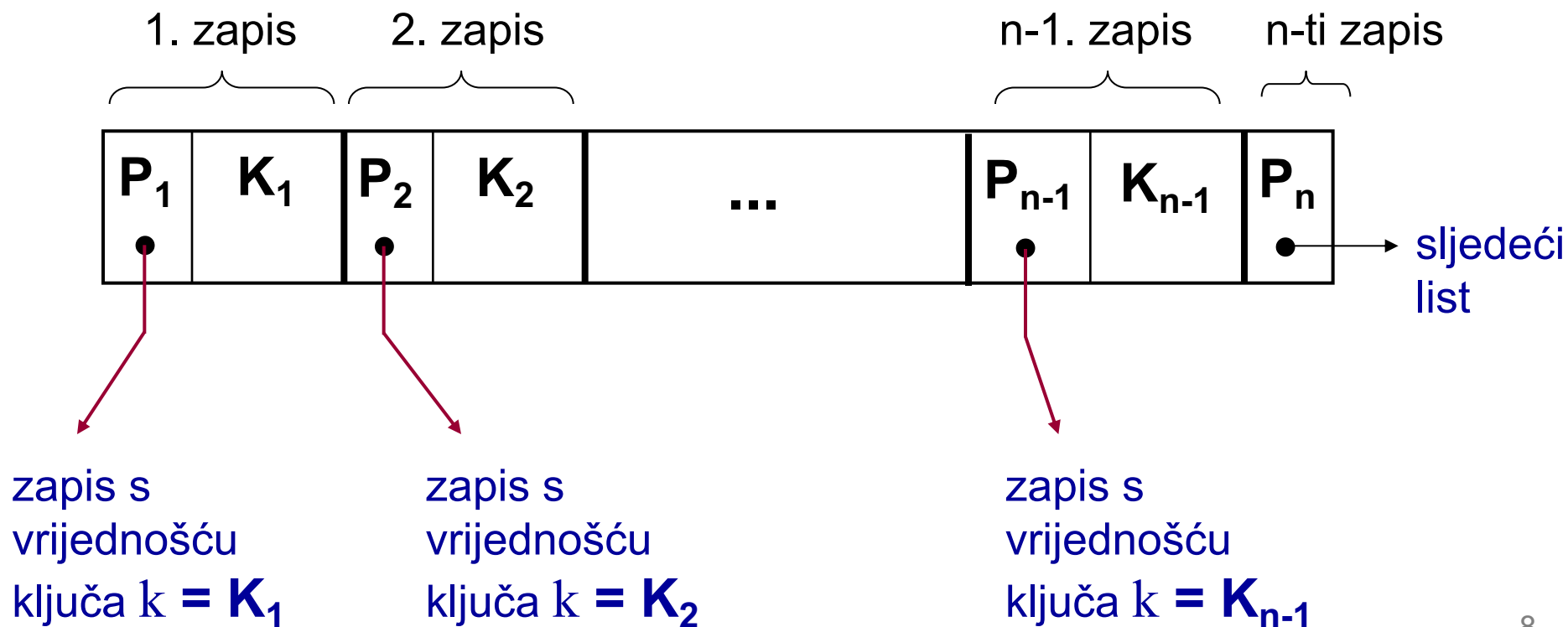
## Struktura internog čvora

- U B<sup>+</sup>-stablu reda **n**, interni čvor sadrži
  - najviše **n** kazaljki
  - najmanje  $\lceil n/2 \rceil$  kazaljki  $\rightarrow \lceil a \rceil$  je najmanji cijeli broj  $\geq a$ 
    - ovo ograničenje ne vrijedi za korijen (2 .. n kazaljki)
- uz **p** kazaljki u čvoru, broj pripadnih vrijednosti  $K_i$  u čvoru je **p-1**



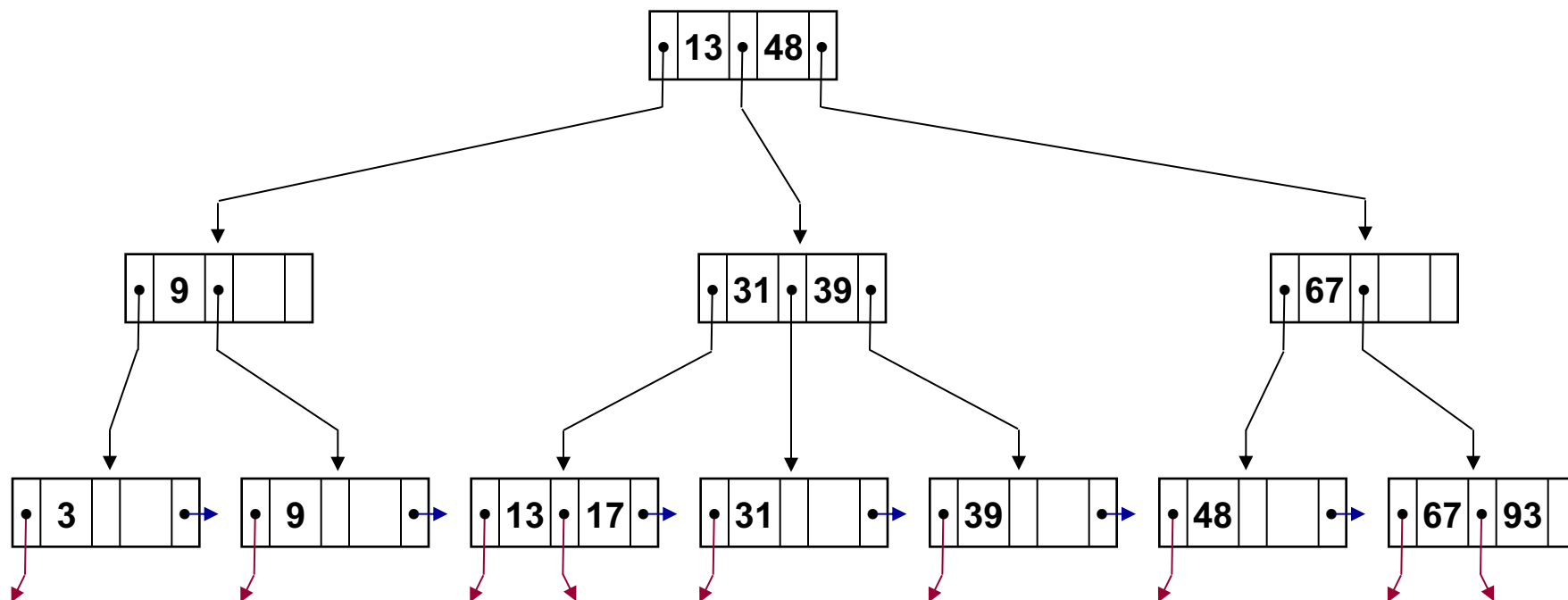
## Struktura lista

- U B<sup>+</sup>-stablu reda **n**, list sadrži
  - najviše **n-1** vrijednosti  $K_i$  i pripadnih kazaljki na zapise
  - najmanje  $\lceil (n-1)/2 \rceil$  vrijednosti  $K_i$  i pripadnih kazaljki na zapise
  - svi listovi, osim krajnje desnog, sadrže kazaljkicu na sljedeći list
    - omogućuju se upiti tipa od-do





## Primjer B<sup>+</sup>-stabla reda 3



kazaljke na zapise u blokovima s podacima

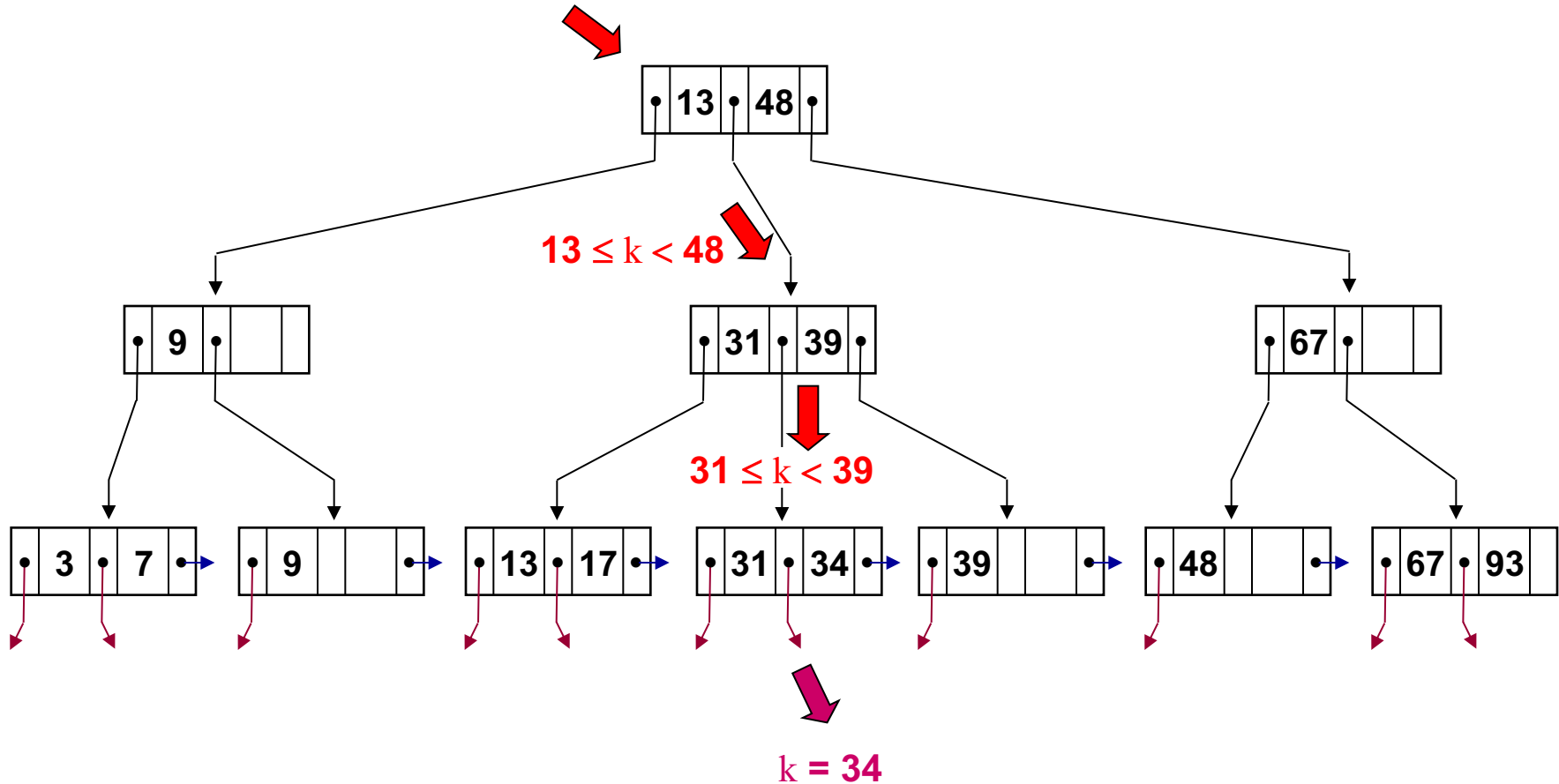
Odabire se red stabla čija primjena rezultira veličinom čvora koja odgovara veličini fizičkog bloka. Ovisno o veličini ključa, red stabla je uglavnom veličine 10 - 200.

### **3. Algoritmi za B-stablo**

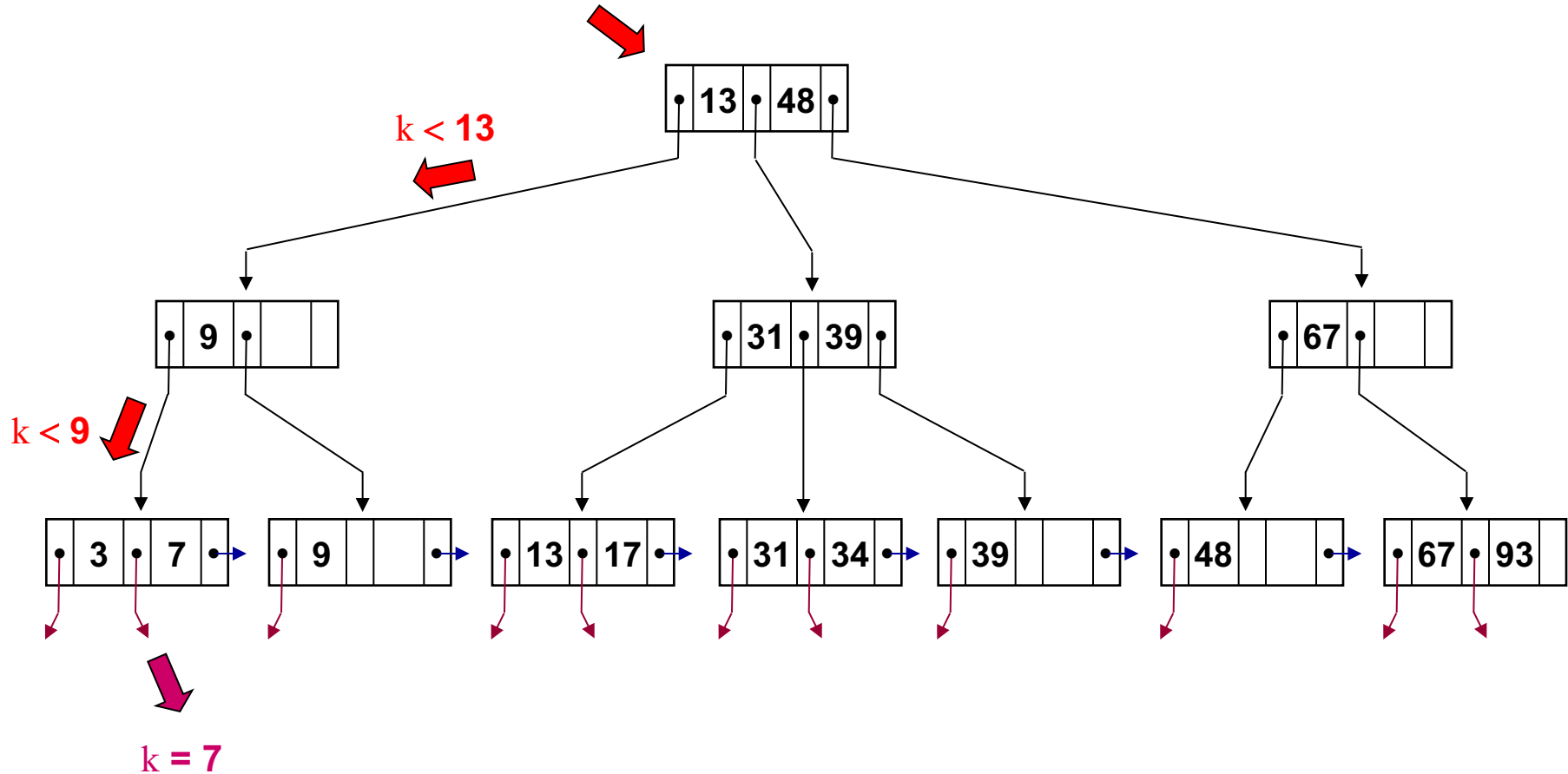
## Algoritam za pretragu B<sup>+</sup>-stabla

- algoritam za traženje zapisa s ključem vrijednosti  $k$  je rekurzivan
  - cilj je u svakom koraku rekurzije (pretraga  $i$ -te razine) pronaći čvor na nižoj,  $(i+1)$ -voj razini, koji će voditi prema listu u kojem se nalazi ključ čija je vrijednost  $k$
- traženje zapisa započinje od korijena (0-te razine)
- u čvoru  $i$ -te razine potrebno je pronaći najveću vrijednost ključa koja je manja ili jednaka traženoj vrijednosti  $k$ 
  - za prvu kazaljku internog čvora nije navedena vrijednost ključa, pa ona "pokriva" sve vrijednosti ključeva manje od prve vrijednosti ključa ( $K_1$ ) navedene u čvoru
- nakon pronalaženja odgovarajuće vrijednosti ključa, slijedi se pripadna kazaljka i time se obavlja pozicioniranje na  $(i+1)$ -vu razinu
- postupak se ponavlja rekurzivno sve dok se ne dođe do lista. U njemu se mora nalaziti, ukoliko postoji, ključ čija je vrijednost  $k$ , te pripadna kazaljka prema traženom zapisu u datoteci

traži se zapis s ključem **34**



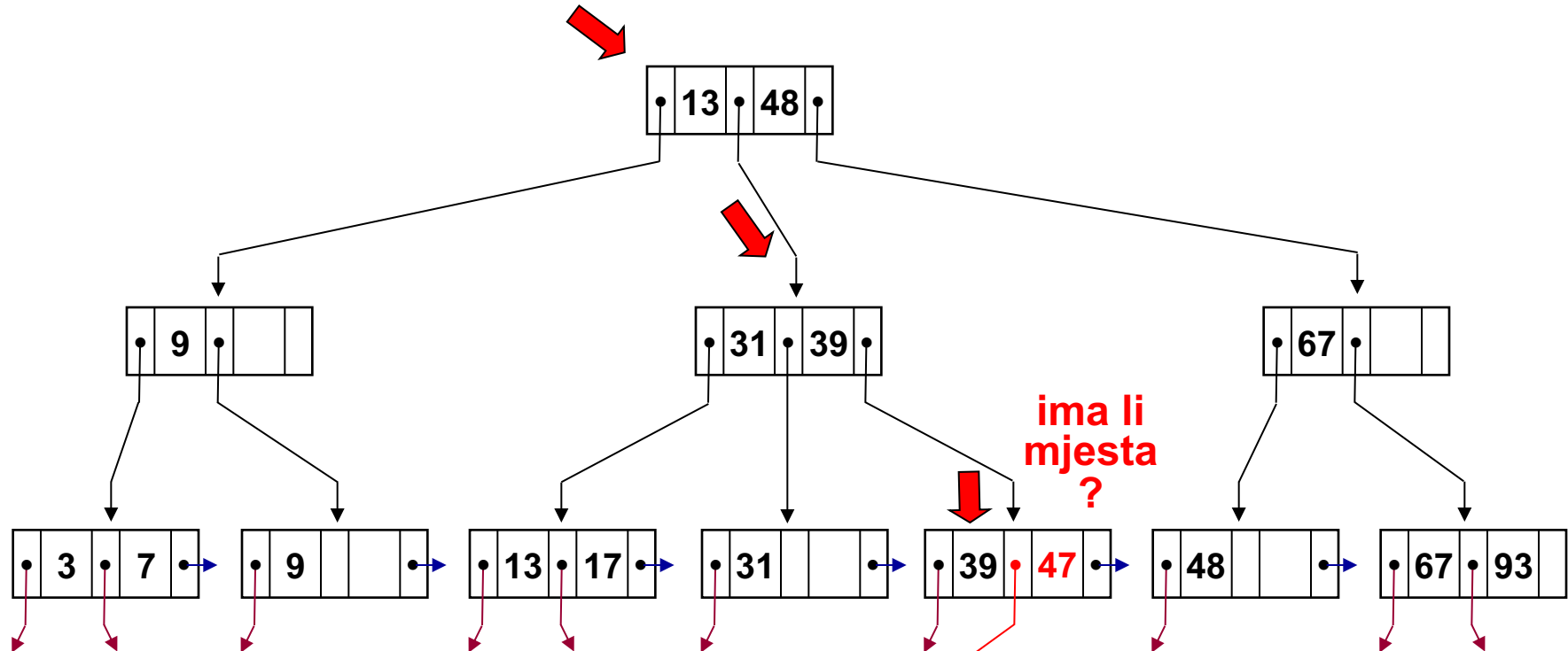
traži se zapis s ključem 7



# Algoritam za dodavanje zapisa u B<sup>+</sup>-stablo

- zapis (podaci) se upisuje u jedan od slobodnih blokova s podacima
- obavlja se algoritam za pronalaženje lista L u koji pripada vrijednost ključa k
- ako čvor L nije popunjen, u čvor se dodaje zapis s ključem k i kazaljkom na pripadni zapis u datoteci, uz očuvanje poretka vrijednosti ključeva
  - nova vrijednost nikad nije prva u čvoru, osim u slučaju krajnjeg lijevog čvora
- ako je čvor L popunjen
  - stvara se novi čvor i zapisi među njima se podijele, pri čemu svakom od čvorova pripadne polovica zapisa
  - budući da je dodan novi čvor, u nadređeni čvor potrebno je dodati zapis s kazaljkom i najmanjom vrijednošću ključa u novom čvoru
  - za dodavanje novog zapisa u nadređeni čvor koristi se ista procedura kao za dodavanje zapisa u čvor na nižoj razini
  - postupak je rekurzivan i mora se obaviti za svaku nadređenu razinu (sve dok se ne dođe do korijena ili se na nekoj od razina nađe dovoljno mjesta za upis vrijednosti ključa i kazaljke, bez dodavanja novih čvorova).
  - ako se dođe do korijena, može se desiti da u korijenu nema mjesta za novi zapis. Tada se dodaje novi čvor i formira se novi korijen na višoj razini

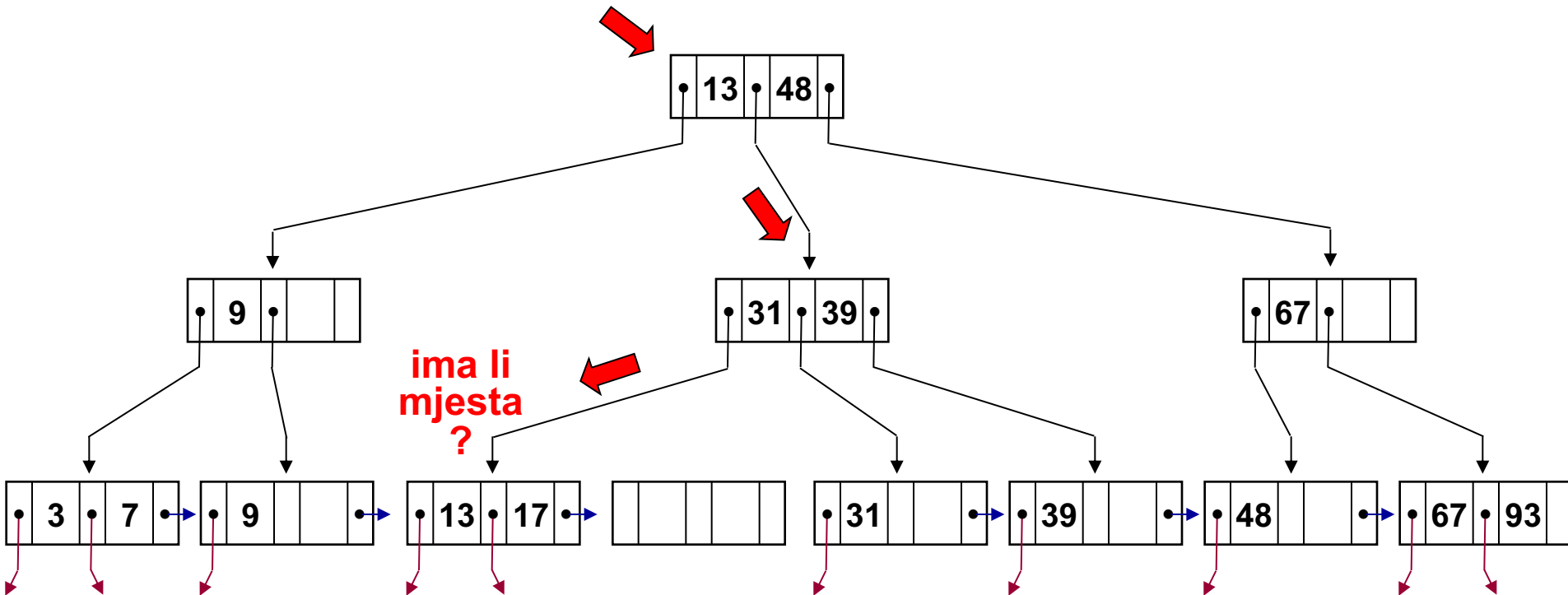
dodavanje zapisa s ključem 47



dodavanje zapisa u blok s podacima

9		93	13	39	67	34	7	47			3		17	48			31
...		...	...	...	...	...	...	...			...		...	...			...
...		...	...	...	...	...	...	...			...		...	...			...

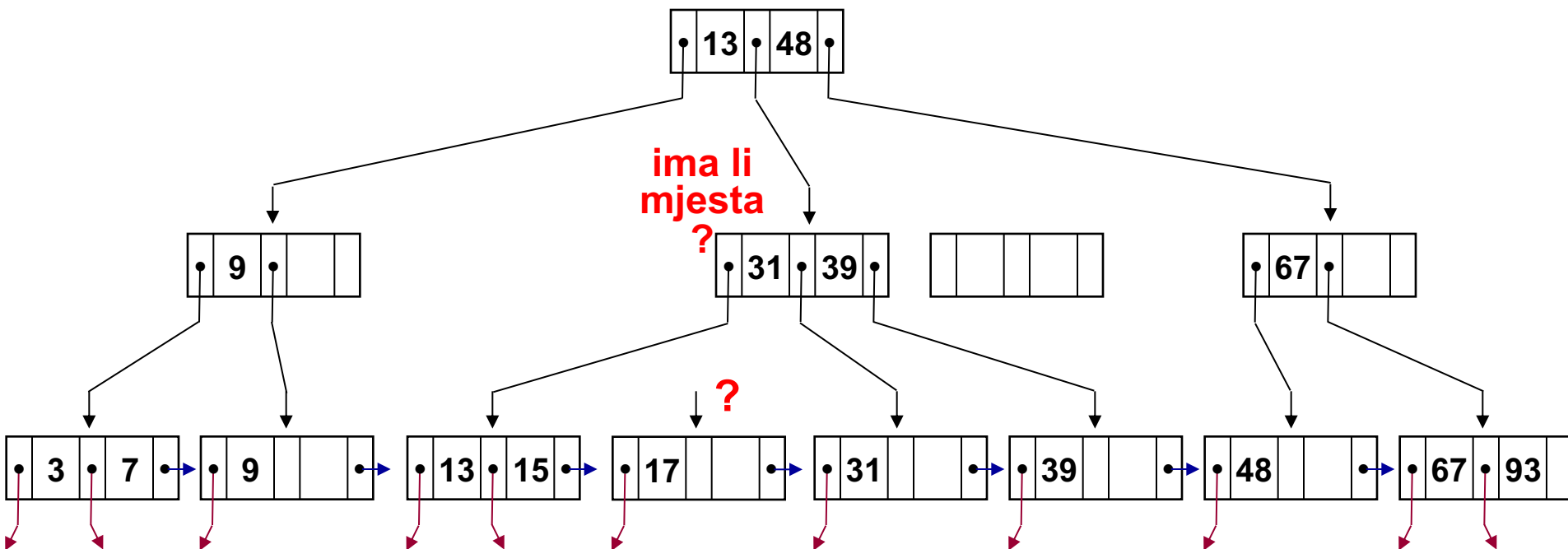
dodavanje zapisa s ključem **15**



dodati novi čvor i podijeliti zapise  
13, 15, 17 među čvorovima

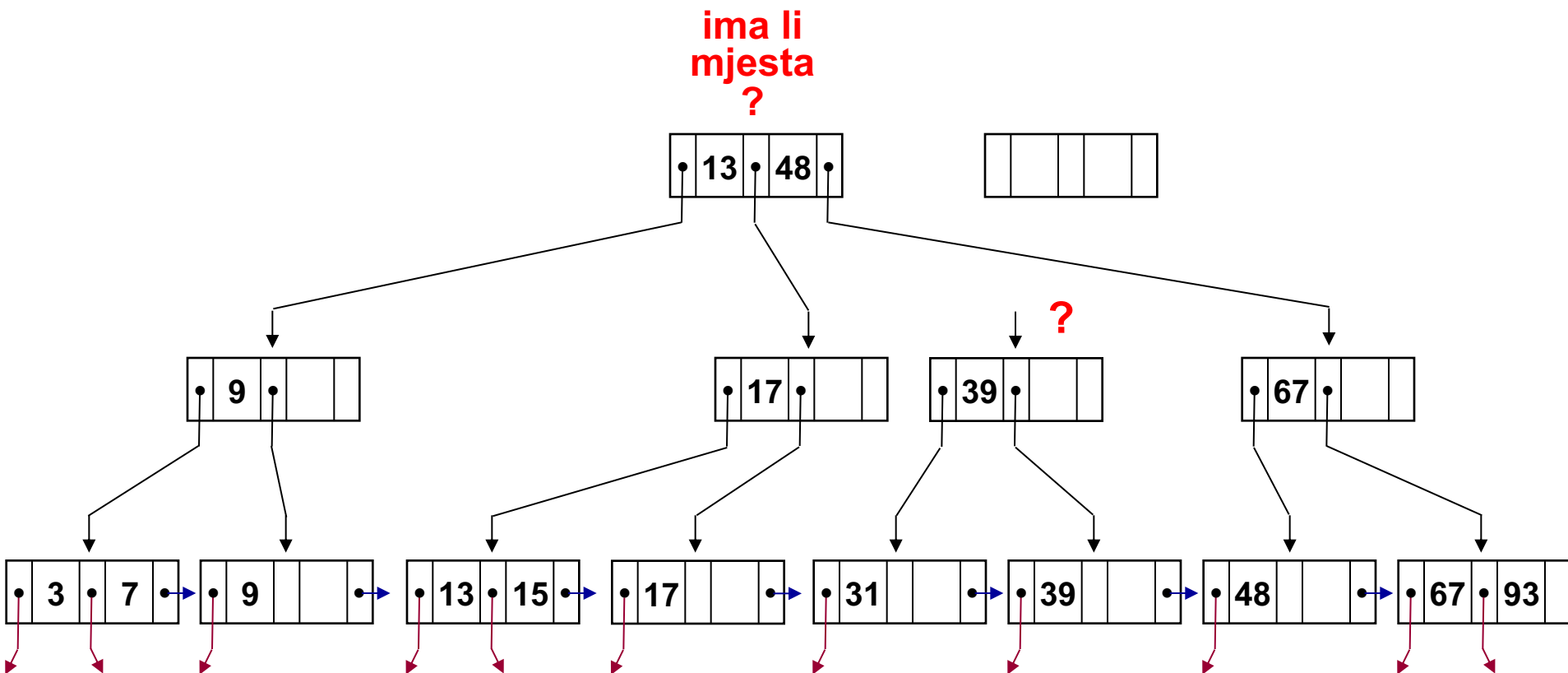


dodavanje zapisa s ključem **15**



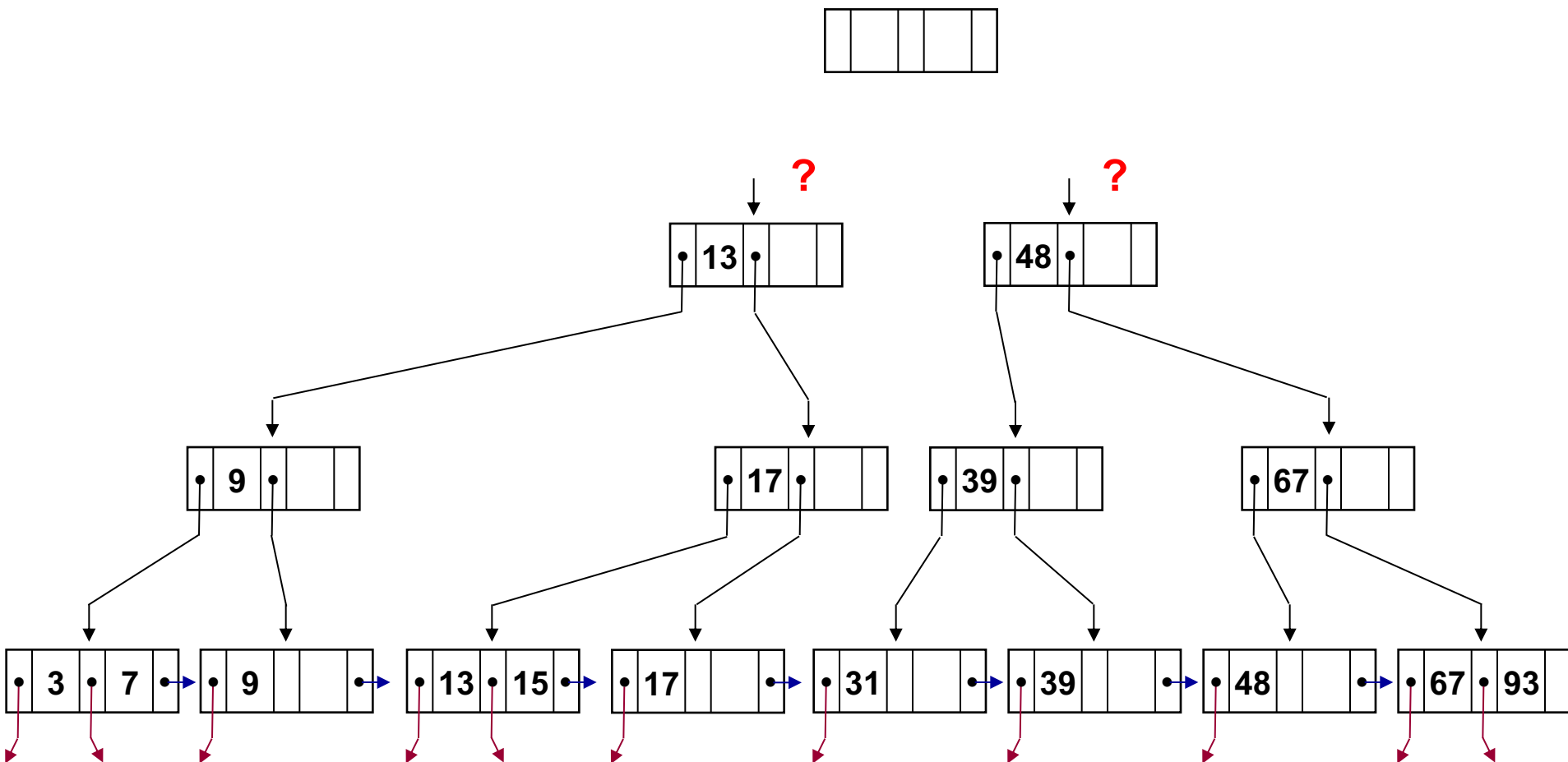
dodati novi čvor i podijeliti zapise  
13, 17, 31, 39 među čvorovima

dodavanje zapisa s ključem **15**



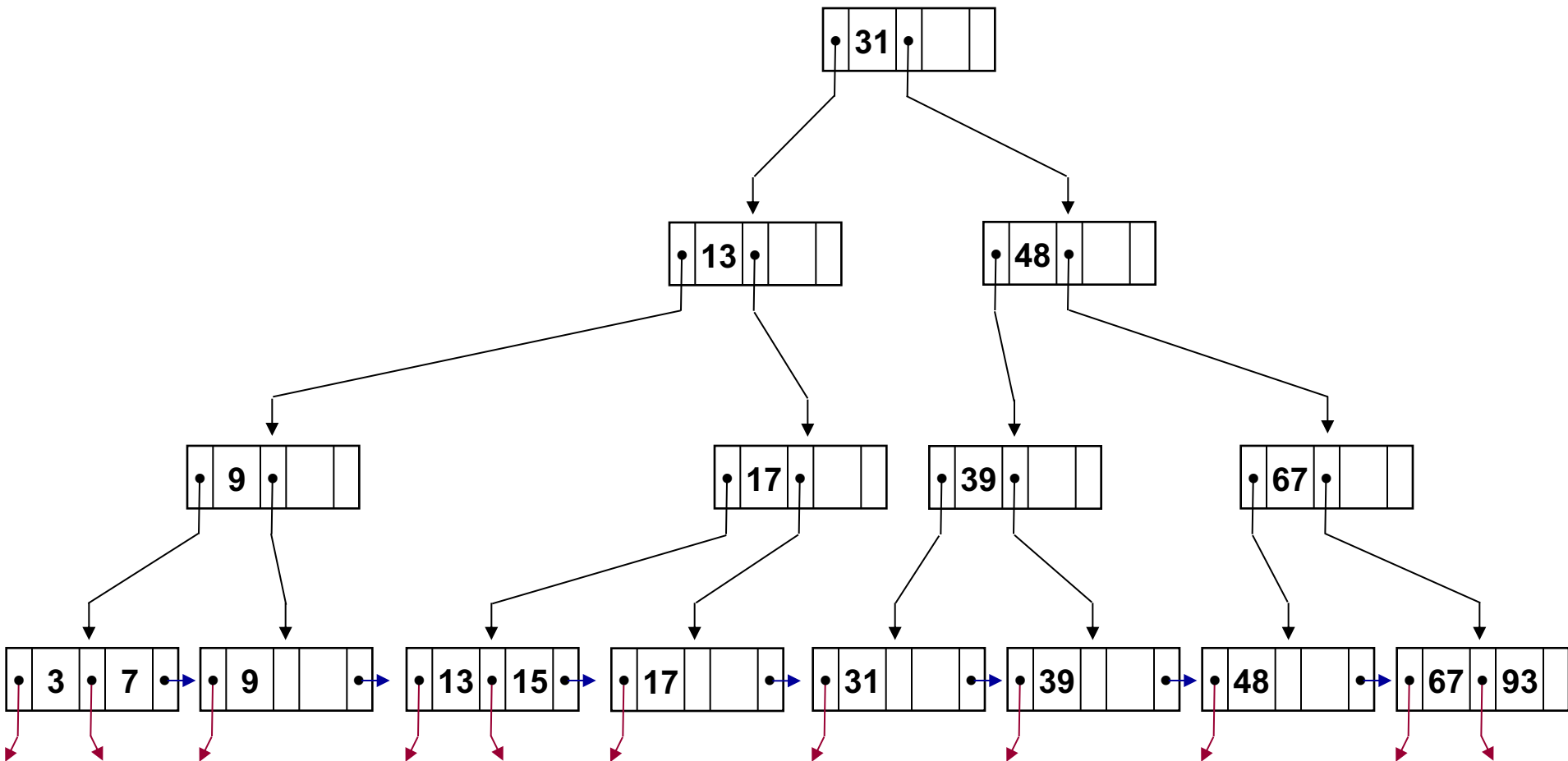
dodati novi čvor i podijeliti zapise  
3, 13, 31, 48 među čvorovima

dodavanje zapisa s ključem **15**



dodati novi korijen i upisati zapise 3, 31

dodavanje zapisa s ključem **15**

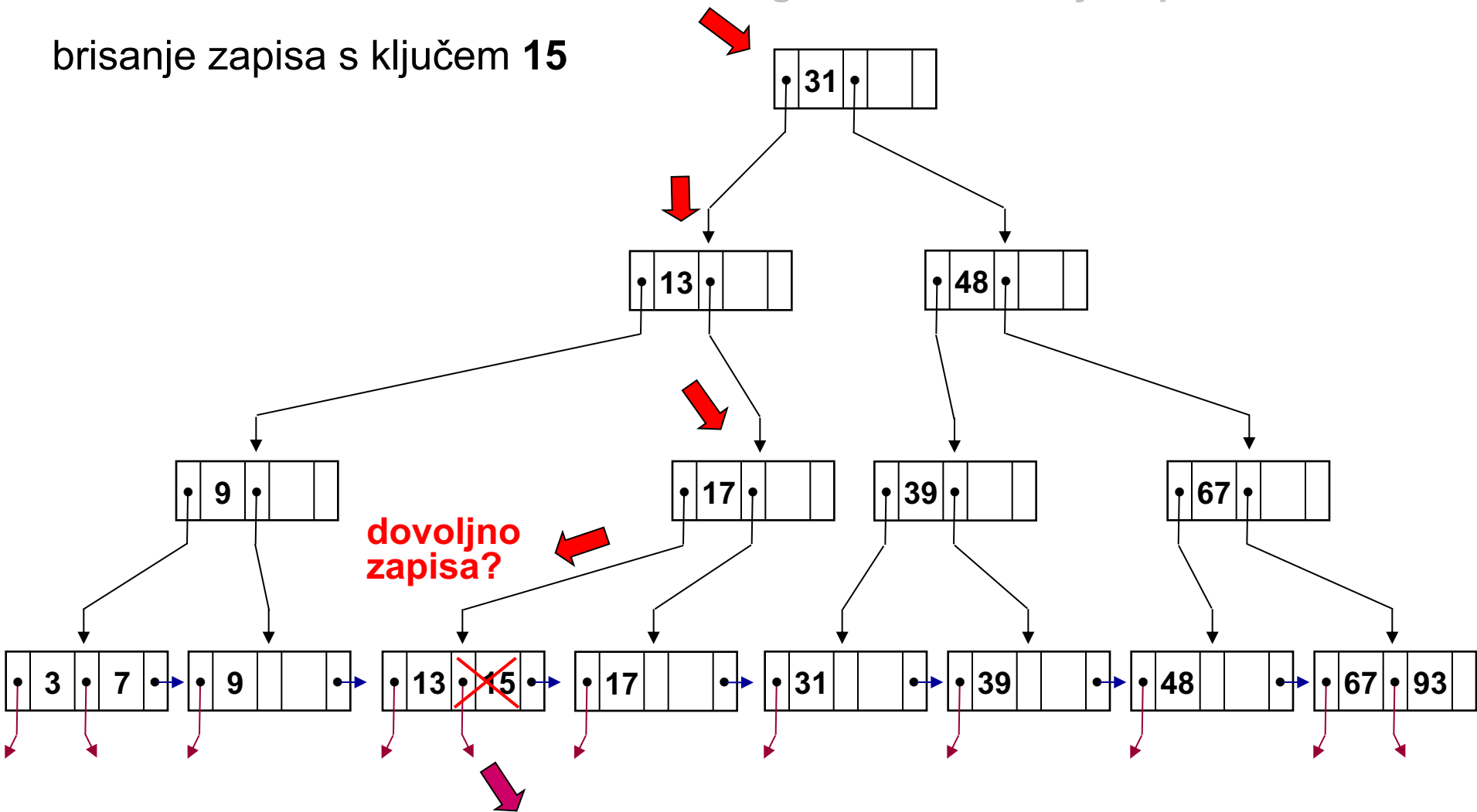


**rezultat je balansirano stablo veće dubine**

# Algoritam za brisanje zapisa iz B<sup>+</sup>-stabla

- obavlja se algoritam za pronalaženje lista L u kojem se nalazi ključ k
- zapis se briše iz bloka s podacima, a vrijednost ključa i kazaljka iz čvora L
- ako čvor L sadrži dovoljan broj zapisa
  - ukoliko je potrebno (onda kada se iz lista obriše prvi zapis, osim u krajnje lijevom listu), mijenja se vrijednost ključa u nekom od predaka
- ako čvor L nakon brisanja nema dovoljan broj zapisa, traži se čvor-brat  $L_x$  koji se nalazi neposredno s lijeva ili s desna čvoru L i ima više od dovoljnog broja zapisa. Ako se takav ne pronađe, traži se bilo koji čvor-brat  $L_x$  koji se nalazi neposredno s lijeva ili s desna čvoru L
  - ako odabrani brat  $L_x$  ima više od dovoljnog broja zapisa, tada se zapisi podijele između čvorova L i  $L_x$ , uz zadržavanje poretka zapisa. U pretcima čvorova L i  $L_x$  obavljaju se potrebne izmjene vrijednosti ključeva
  - inače (odabrani brat  $L_x$  ima upravo dovoljan broj zapisa), čvorovi L i  $L_x$  se spajaju u jedan čvor. U nadređenom čvoru briše se zapis za L i eventualno mijenja vrijednost ključa u nekom od predaka. Brisanje zapisa u nadređenom čvoru svodi se na rekurzivno izvođenje procedure za brisanje. Ako se putem prema korijenu dođe u situaciju da treba spojiti jedina dva čvora-djeteta korijena, tada se oni spajaju, postaju novi korijen stabla, a stari se korijen briše

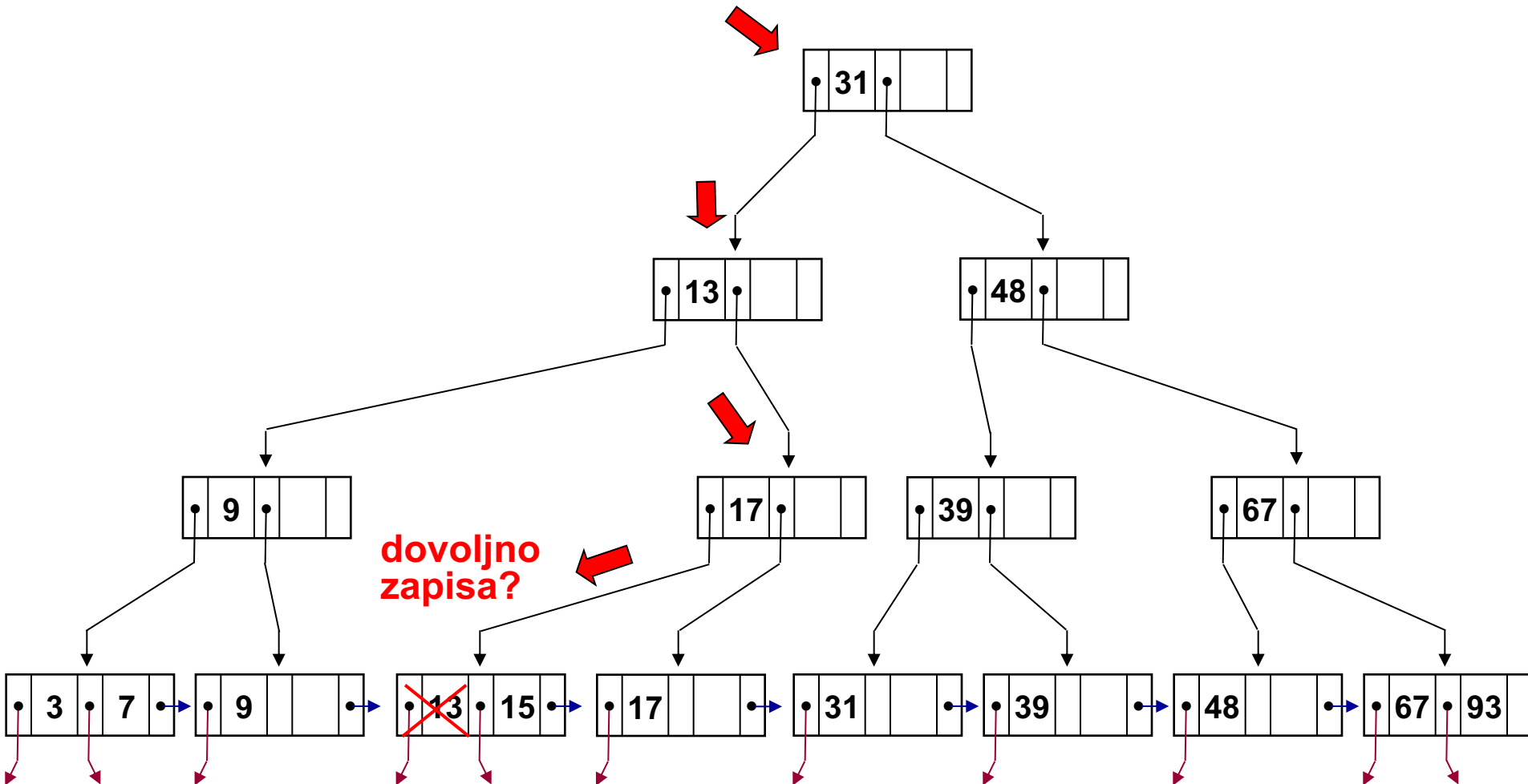
brisanje zapisa s ključem 15



brisanje zapisa iz bloka s podacima

39		9	31	13	7	48	93			3		<del>15</del>	67			17
...		...	...	...	...	...	...			...		...	...			...
...		...	...	...	...	...	...			...		...	...			...

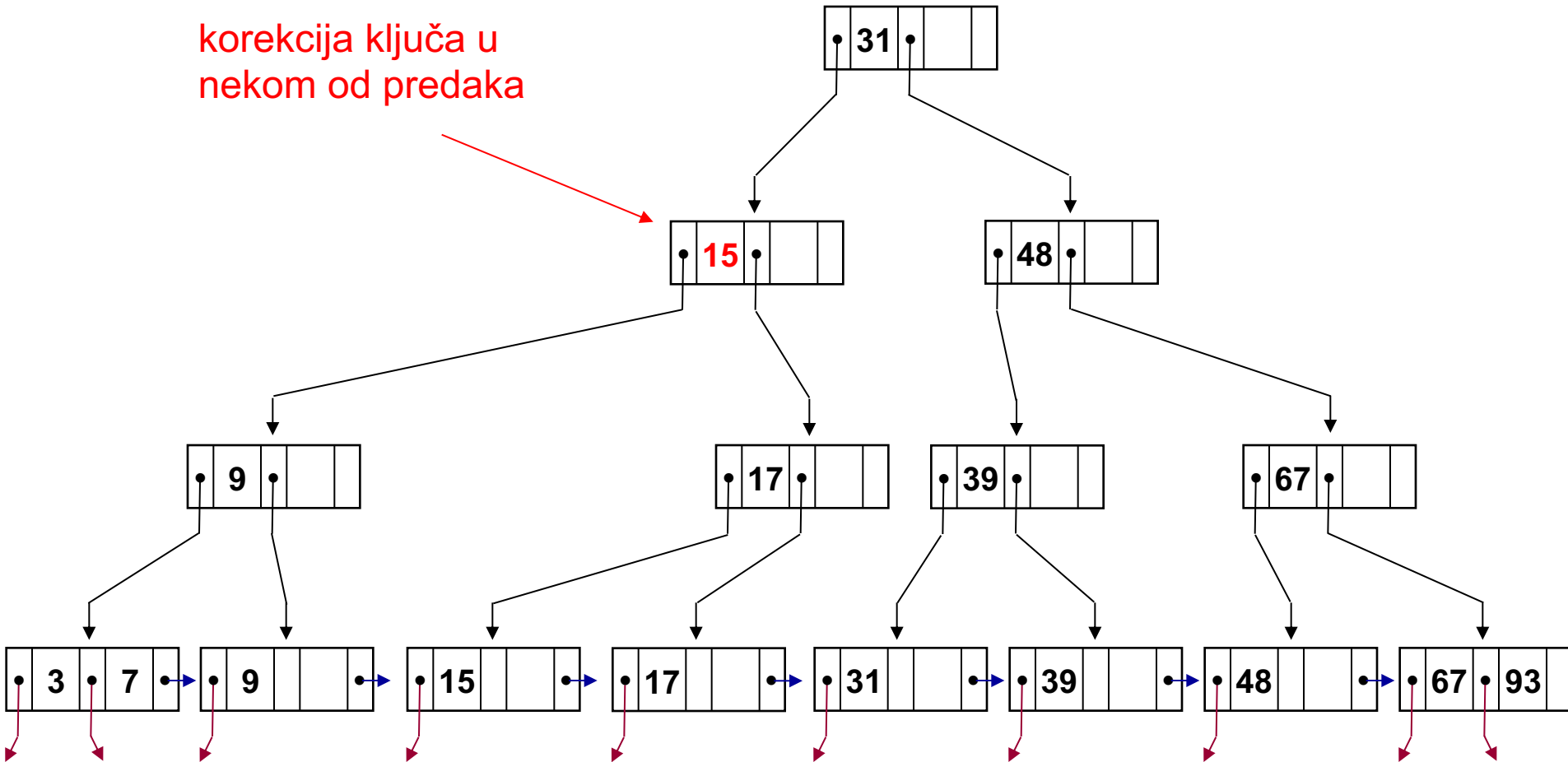
brisanje zapisa s ključem 13



obaviti korekciju ključa u nekom od predaka jer je obrisani prvi zapis lista

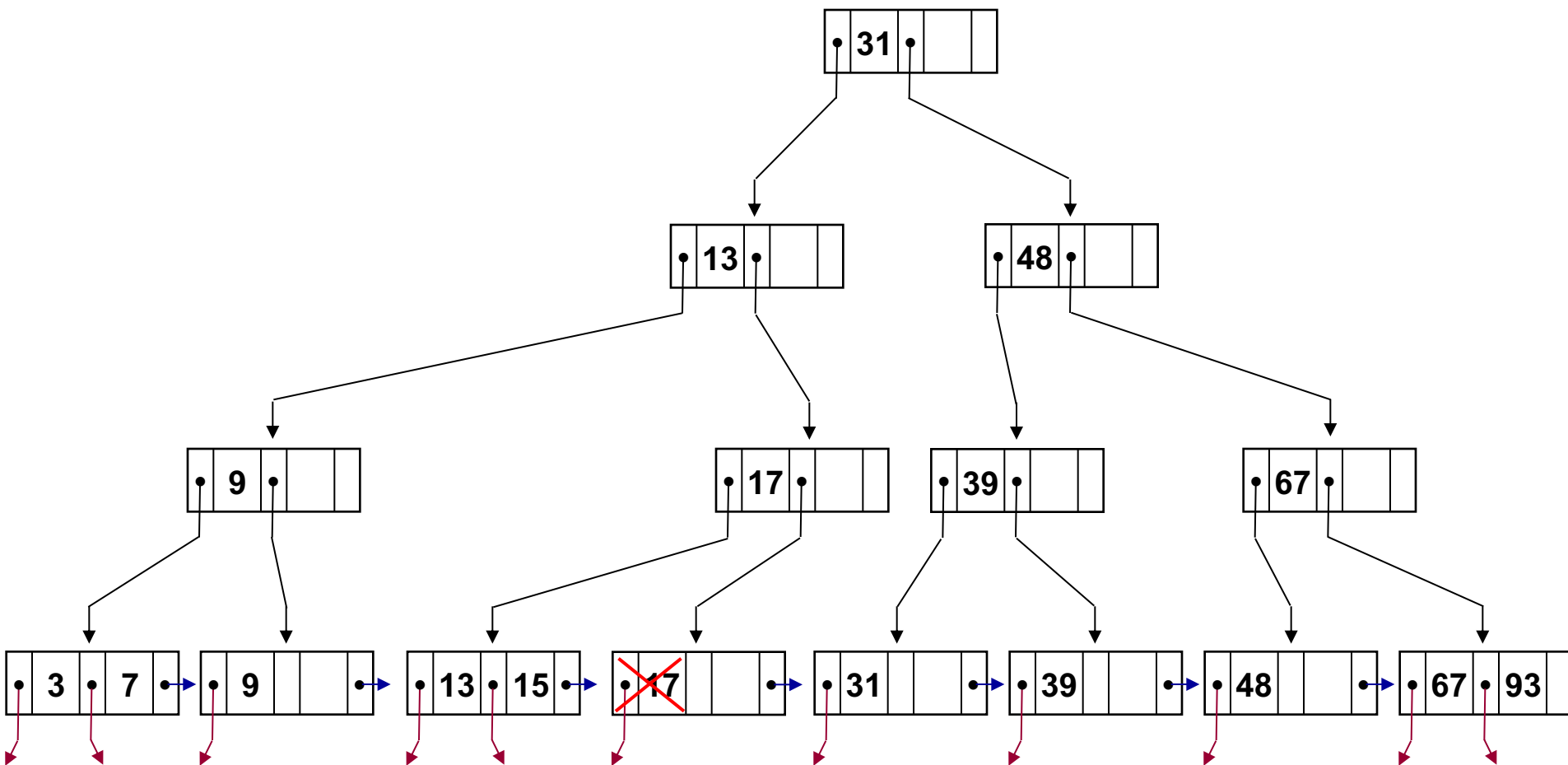
brisanje zapisa s ključem **13**

korekcija ključa u  
nekom od predaka





brisanje zapisa s ključem **17**

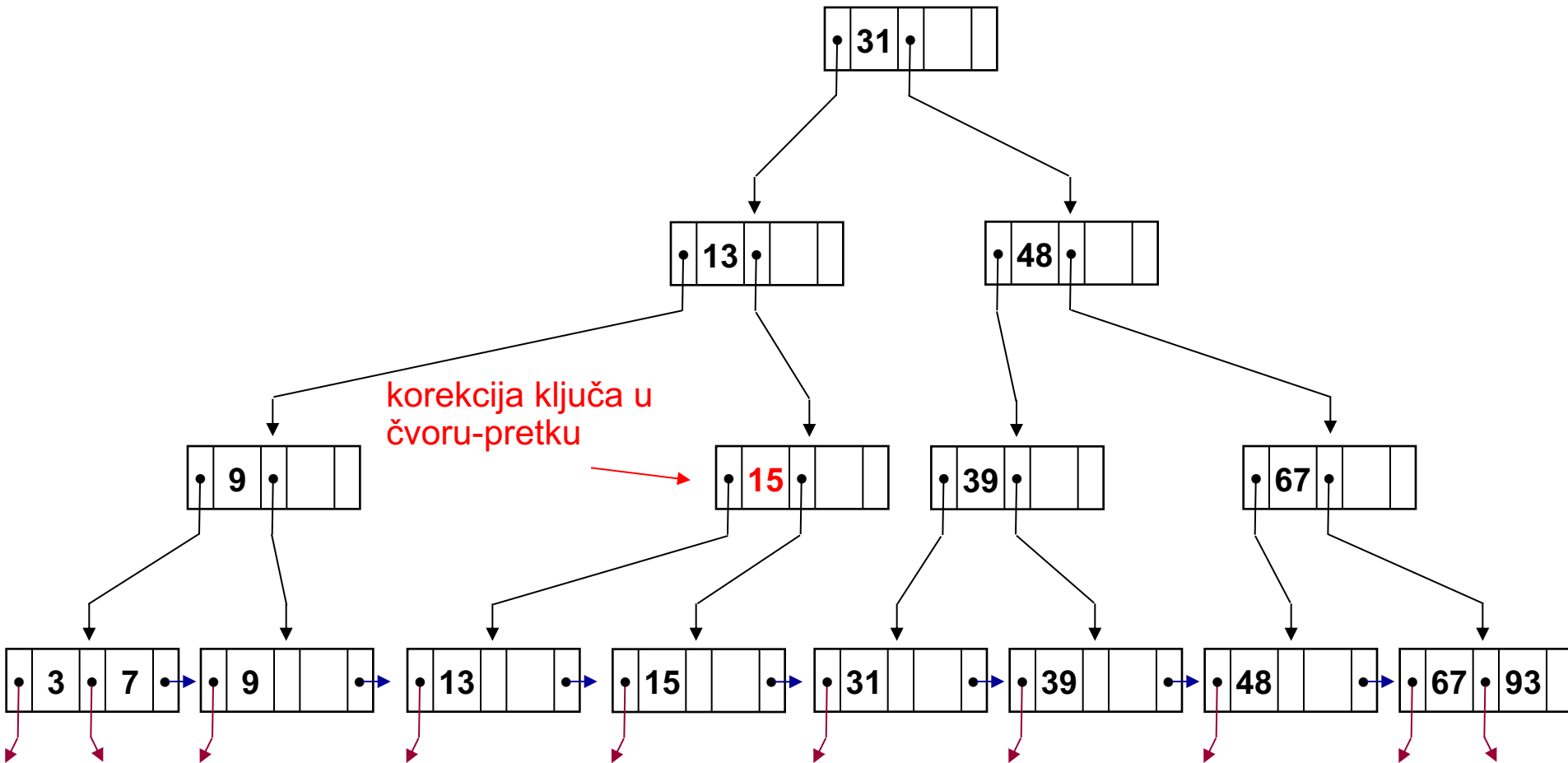


ima li brat  
 $L_x$  više  
nego treba?

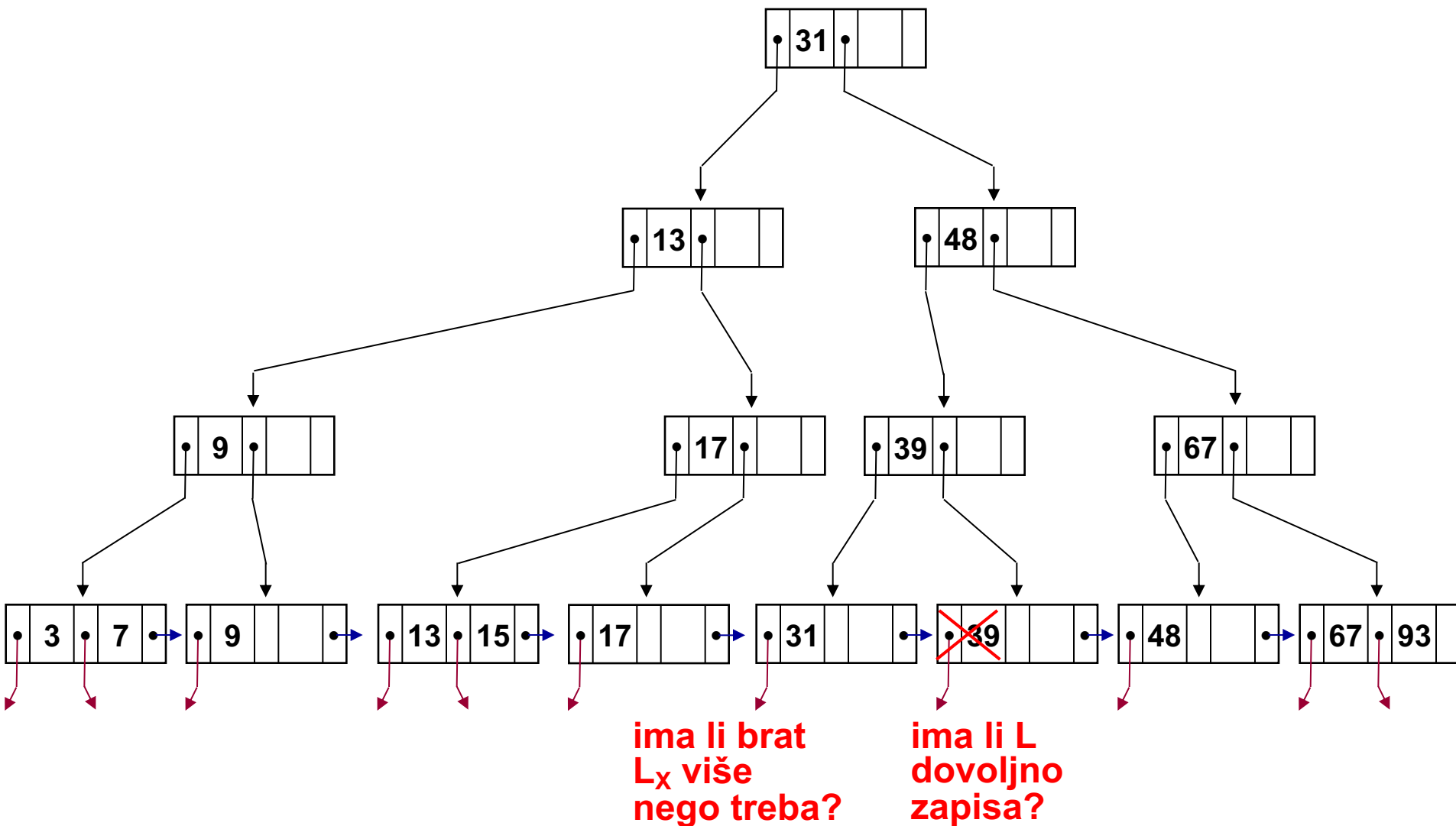
ima li L  
dovoljno  
zapisa?

podijeliti zapise s bratom  $L_x$  i obaviti potrebne  
korekcije ključeva u pretcima

brisanje zapisa s ključem **17**

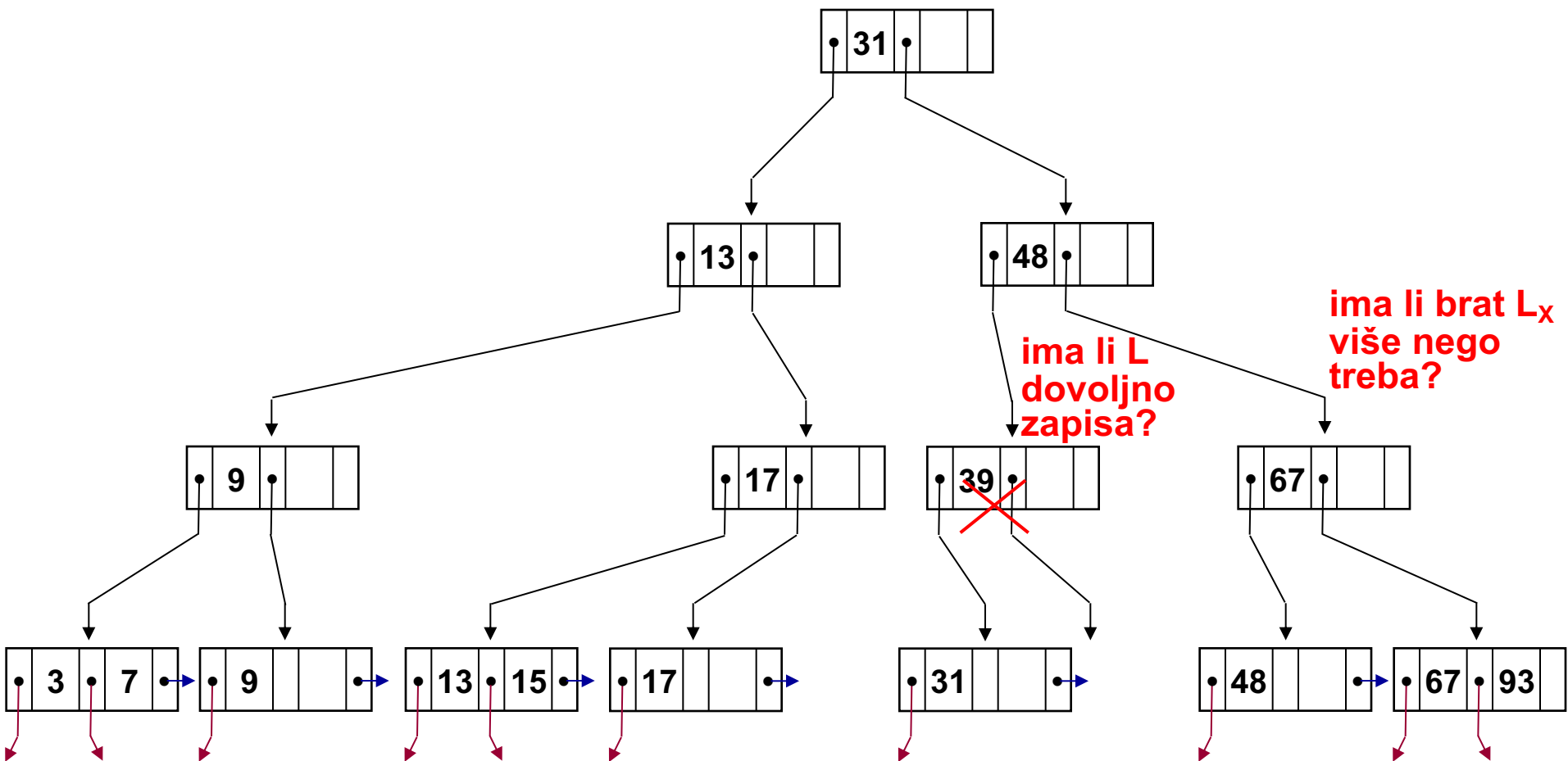


brisanje zapisa s ključem **39**



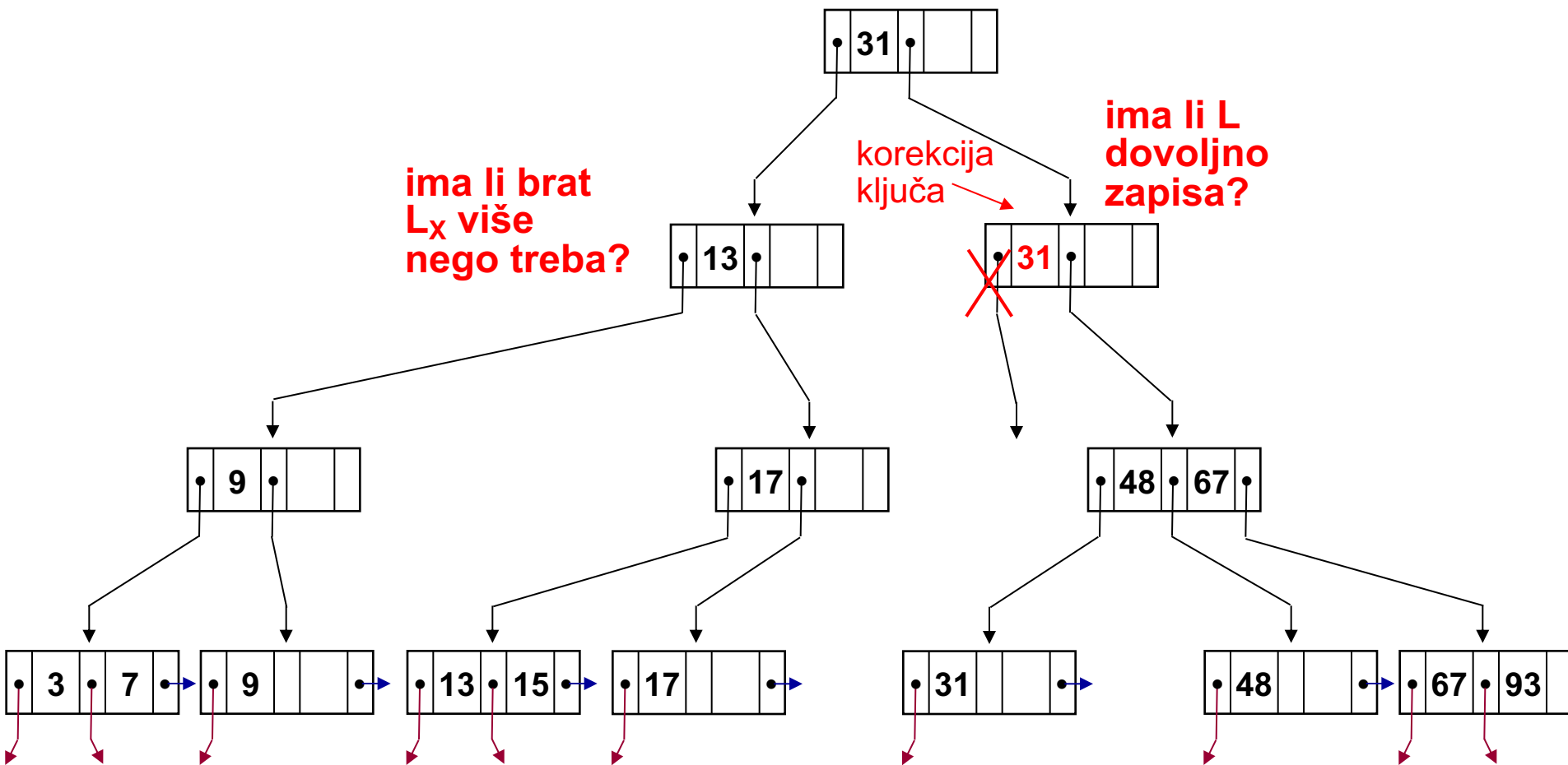
spojiti L i  $L_x$  . Obrisati kazaljku na L iz nadređenog čvora i (eventualno) promijeniti vrijednost ključa u nekom od predaka.

brisanje zapisa s ključem **39**



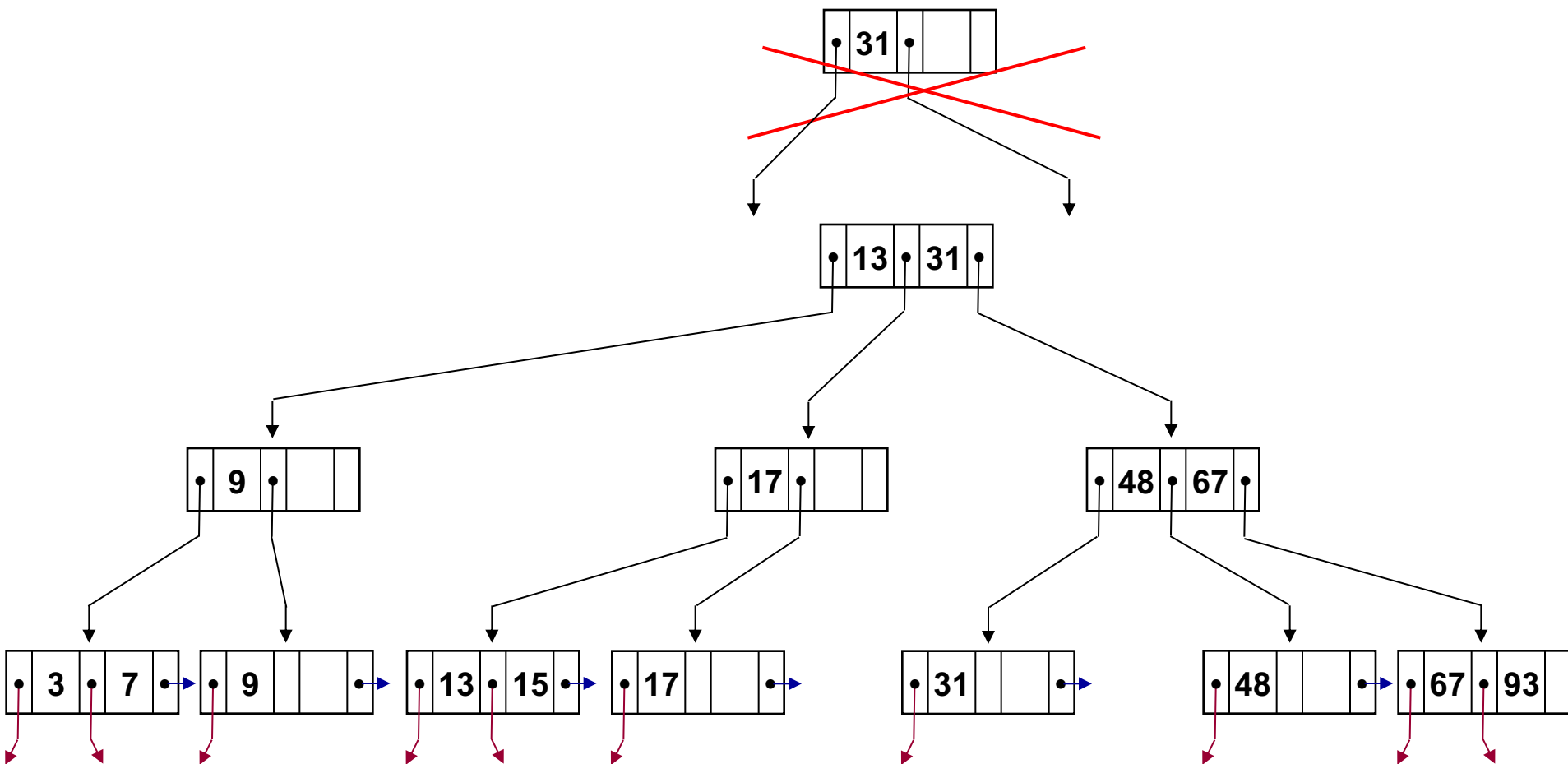
spojiti L i  $L_x$  . Obrisati kazaljku na L iz nadređenog čvora i (eventualno) promijeniti vrijednost ključa u nekom od predaka.

brisanje zapisa s ključem **39**



spojiti L i  $L_x$ . Spajanjem L i  $L_x$  nastaje novi korijen.  
Obrisati stari korijen.

brisanje zapisa s ključem **39**



**rezultat je balansirano stablo manje dubine**

## 4. Učinkovitost operacije pretrage u B-stablu

- broj I/O operacija u stablu pri traženju zapisa ovisi o broju razina u stablu
- pretpostavka: stablo reda  $n$  sadrži kazaljke na  $m$  zapisa podataka
- stablo će imati najveći broj razina ako su čvorovi najmanje popunjeni
  - najmanja popunjenost korijena je  $2$
  - najmanja popunjenost internog čvora:  $\lceil n / 2 \rceil$
  - najmanja popunjenost lista:  $\lceil (n - 1) / 2 \rceil \approx \lceil n / 2 \rceil$ , za dovoljno veliki  $n$
- u korijenu (1. razina) ima 1 čvor i najmanje  $2$  kazaljke
- na 2. razini ima najmanje 2 čvora i zato najmanje  $2 \cdot \lceil n / 2 \rceil$  kazaljki
- u čvorovima 3. razine ima najmanje  $2 \cdot \lceil n / 2 \rceil \cdot \lceil n / 2 \rceil$  kazaljki
- u čvorovima  $i$ -te razine ima najmanje  $2 \cdot \lceil n / 2 \rceil^{i-1}$  kazaljki
- za broj zapisa u podatkovnim blokovima stabla koje ima  $d$  razina vrijedi:
  - $m \geq 2 \cdot \lceil n / 2 \rceil^{d-1}$
- iz toga slijedi
  - $d \leq \log_{\lceil n/2 \rceil} (m / 2) + 1$

$$d \leq \log_{\lceil n/2 \rceil} (m / 2) + 1$$


**Primjer:** za  $m = 1\,000\,000$ ,  $n = 70$ , ukupni broj razina (uključujući i razinu korijena) u najgorem slučaju je 4.

1.  točno 1 čvor, najmanje 2 kazaljke

2.  najmanje 2 čvora, najmanje 70 kazaljki

3.  najmanje 70 čvorova, najmanje 2 450 kazaljki

4.  najmanje 2 450 čvorova, najmanje 85 750 kazaljki

5.  najmanje 85 750 čvorova, najmanje 3 001 250 kazaljki

B<sup>+</sup>-stablo koje bi imalo ukupno 5 razina, moralo bi imati **najmanje** 3 001 250 kazaljki na zapise.