Napredni algoritmi i strukture podataka

Merkle stablo, B stablo

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

- Formalno gledamo, Merkle stabla uzimaju skup podataka $(x_1, \ldots, x_n)$ na ulazu
- Povratnu vrednost je **Merkle root hash** $h = \mathrm{MHT}(x_1, \ldots, x_n)$
- MHT **collision-resistant hash funkcija**
- *Hash* funkcija je **collision-resistant hash funkcija** ako je teško pronaći dva ulaza koja *hash*-iraju isti izlaz
- Formalno, za ulaze a i b, $a \neq b$ ali $H(a) = H(b)$

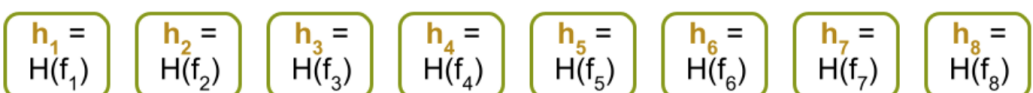
Merkle stablo — formiranje

- Algoritam za formiranje Merkle stabla je relativno jednostavan
- Merkle stablo ima **bottom-up** pristup izgradnje, zbog svoje specifičnosti
- Formiranje stabla počinje od dna tj. konkretizovanih podataka — **data block**
- Polako idemo do vrha, gradeći **Merkle root** element
- Prvi element koji gradimo je **list**
- Svaki podatak propustimo kroz *hash* funkciju, i tako formiramo prvi nivo — **list**

- Nakon toga, svaka **dva susedna** elementa grade naredni nivo propuštajući njihove zajedničke hash vrednosti kroz hash funkciju
- Pošto radimo sa binarnim stablima, ako na nekom nivou nemamo odgovarajući čvor, možemo da dodamo *empty* elment da bi algoritam mogao da se nastavi
- Kada propustimo poslednja dva čvora kroz hash funkciju dobijamo **Merkle root** element
- Time se algoritam za formiranje završava i formirali smo Merkle stablo

Merkle stablo - formiranje, primer

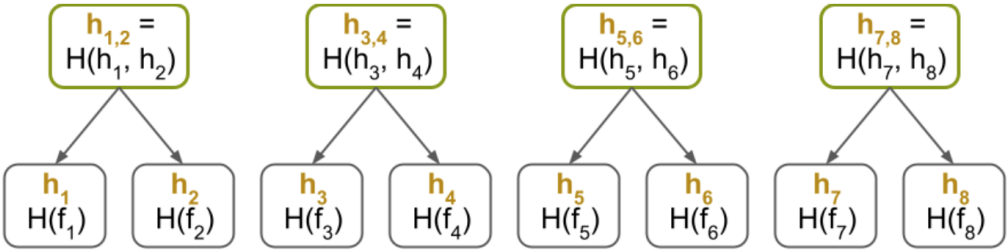
- Pretpostavimo da imamo 8 blokova podataka (fajlova) $f = (f_1, ..., f_8)$
- Svaki podataka f_i propustimo kroz hash funckiju H i dobijamo njegov *hash*
- Dobijamo hash vrednost za prvi nivo $h_i = H(fi), h_i = (h_1, ..., h_8)$
- H reprezentuje **collision-resistant hash** funkciju



(Decentralized Thoughts, Merkle trees)

Merkle stablo - formiranje, primer

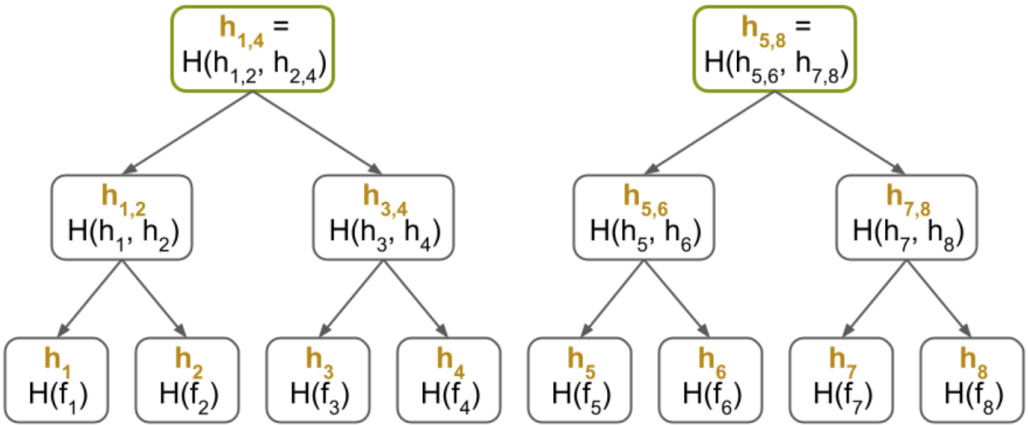
- Nakon formiranja listova, heširamo svaka dva susedna *hash-a*, da bi formirali sledeći nivo $h_{k,m} = H(h_i, h_{i+1})$
- Ako nam fali *hash*, da bi svako imao suseda :(, prosto napravimo prazan *hash* i nastavimo dalje



(Decentralized Thoughts, Merkle trees)

Merkle stablo - formiranje, primer

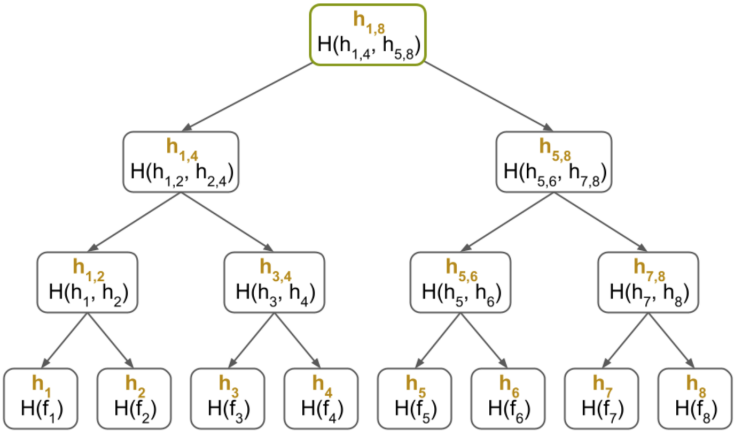
- Formiramo naredni nivo stabla



(Decentralized Thoughts, Merkle trees)

Merkle stablo - formiranje, primer

Idemo isto... i dobijamo $h_{1,8} = H(h_{1,4}, h_{5,8})$



(Decentralized Thoughts, Merkle trees)

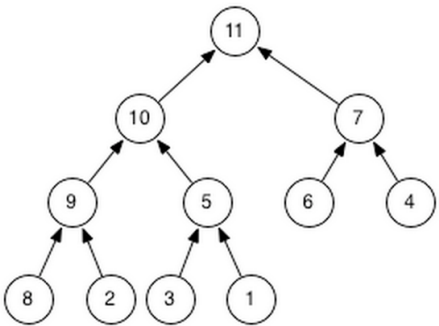
Merkle stablo — napomena

- Ono što smo dobili na kraju $h_{1,8}$ je **Merkle root hash**
- Obratiti pažnju da svaki čvor u stablu čuva **hash** vrednost
- Listovi čuvaju hash vrednost (blokova) podataka $h_i = (h_1, ..., h_8)$
- Čvorovi koji nisu listovi, i nisu **Merkle root hash**, čuvaju *hash* vrednost svoje dece — **internal node**

- Ako nam na nekom nivou fali par za neki element, prosto dodamo **prazan hash** kako bismo formirali par
- Može se lako generalizovati i izračunati Merkle stablo za bilo koji broj n podataka
- Formalno zapisano, prethodni primer se može zapisati kao $h_{1,8} = \mathrm{MHT}(f_1, ..., f_8)$
- Merkle stabla se formiraju rekurzivno, od dna ka vrhu
- Ovaj proces može biti procesno zahtevan!
- To nikada nemojte izgubiti iz vida

Serijalizacija stabla

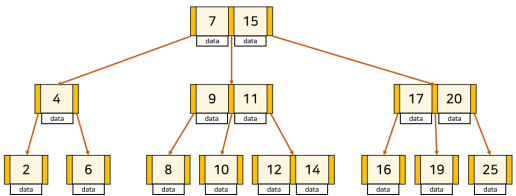
- Ako imamo stablo kao sa slike, treba da idemo kroz njega nekim od poznatih algoritama
- Jedna opcija je da idemo po nivoima:
  - [11 10 7 9 5 6 4 8 2 3 1]
- Treba voditi računa da ako na nekom nivou imamo manjka elmenata, treba da zapišemo nekakav marker da nam bude jasan znak za kasnije!
- Ovo neće biti problem kod Merkle stabala, ali u opštem slučaju treba voditi računa



(Ritambhara, Storing Binary Tree in a file)

B stablo

- B stablo je stablo pretrage koje može da sadrži više ključeva unutar jednog čvora
- Ključevi u čvoru su sortirani u rastućem redosledu
- Između svaka dva ključa nalazi se podstablo sa ključevima koji su veći od levog i manji od desnog ključa
- Pre prvog/posle poslednjeg ključa nalaze se podstabla sa manjim/većim ključevima



Svojstva

- B stablo nasleđuje svojstvo stabala pretrage da se za svaki čvor u levom podstablu nalaze samo čvorovi sa manjim elementima, a u desnom podstablu samo čvorovi sa većim elementima
- Pored toga, za B stablo reda **m** važi da:
  - Svaki čvor sadrži do m dece i do m-1 ključeva
  - Svaki čvor (sem korena) sadrži barem $\mathrm{b} = \lceil \mathrm{m}/2 \rceil$ dece i barem $\mathrm{b} - 1$ ključeva
  - Svaki list je iste dubine
- Ovakva svojstva osiguravaju da je svaki čvor (sem korena) barem polovično popunjen
- Maksimalna dubina stabla sa N elemenata je $1 + \log_{\mathrm{b}}((\mathrm{N}+1)/2)$ -> vreme pretrage, dodavanja i brisanja je $\mathrm{O}(\log \mathrm{N})$

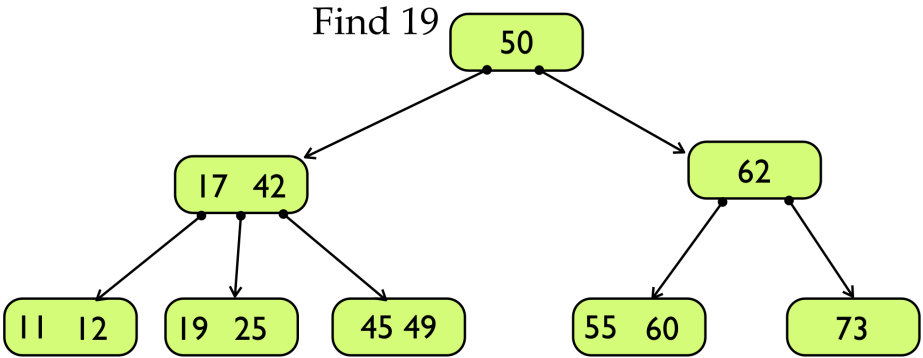
B stablo vs binarno stablo pretrage

- Vreme izvršavanja svih operacija zavisi od dubine stabla
- U opštem slučaju, faktor grananja je kod B stabla veći nego kod binarnog stabla pretrage, što znači da će dubina B stabla biti manja
- Koristimo ih kao efikasnu strukturu u memoriji kada je broj podataka jako veliki, ili još češće za indeksiranje fajlova na disku

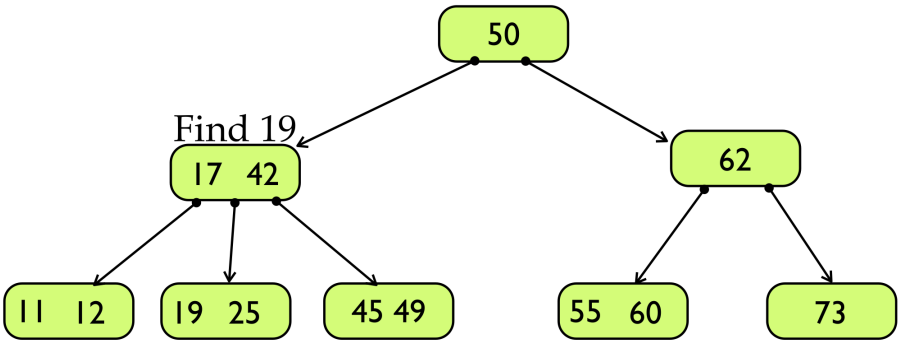
Pretraga

- Pretraga karakteristična za bilo koje stablo pretrage (manji elementi levo od trenutnog, veći su desno)
- Tražeći ključ *key*, krenemo od korena i proveravamo svaki ključ *k* u čvoru:
  - key == k — pronašli smo traženi ključ
  - key < k — pređi na čvor koji je *index(k)*-to dete trenutnog čvora
  - key > k AND no next k — pređi na čvor koji je *index(k)+1*-to dete trenutnog čvora
  - key > k AND exists next k — k = next k

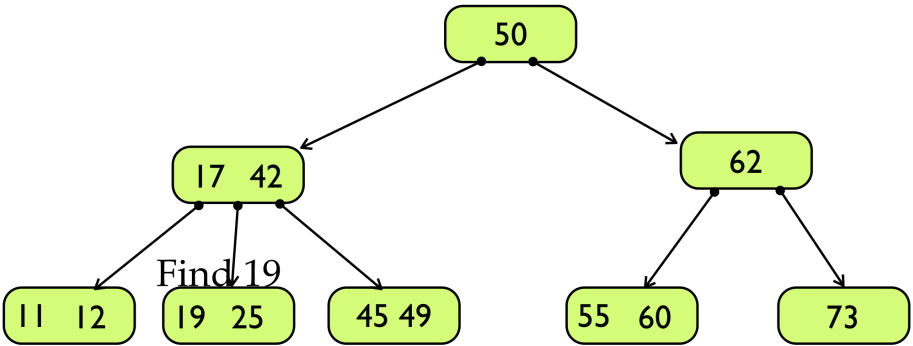
Primer



Primer



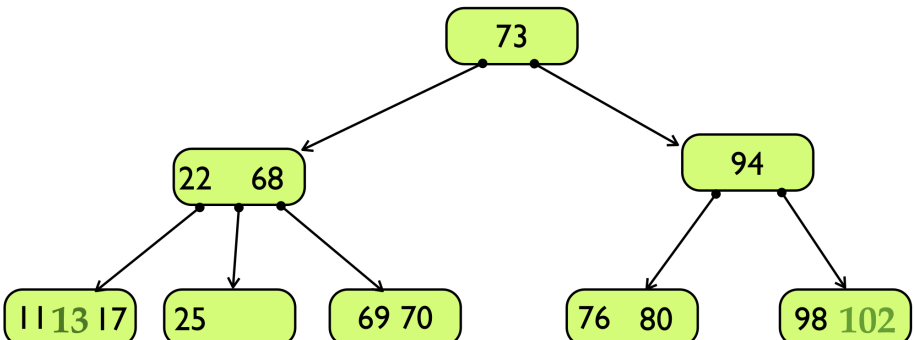
Primer



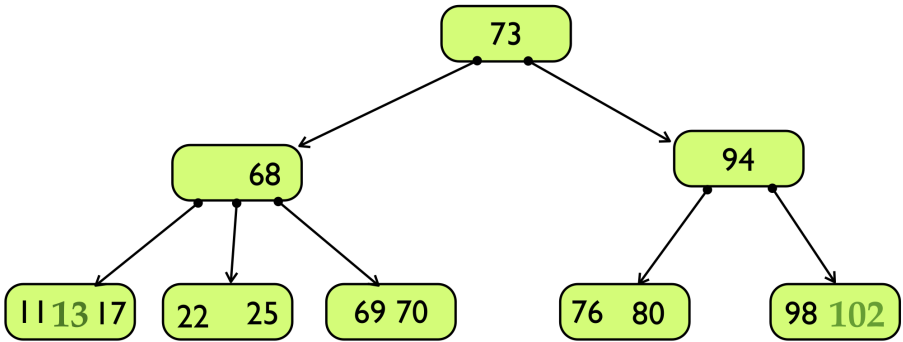
Dodavanje

- Započinje neuspešnom pretragom ključa i uvek se završava u listu
- Kada dodamo ključ u odgovarajući čvor i on nema više od MAX elemenata, završavamo proceduru
- U suprotnom, radimo rotaciju ključeva:
  - Pronađi sibling čvor koji sadrži manje od MAX elemenata
  - Prebaci odgovarajući ključ iz roditelja u sibling čvor
  - Prebaci novi ključ iz deteta u roditeljski čvor
- Ili podelu čvora (kada rotacija nije moguća):
  - Čvor delimo na tri dela: ključ u sredini, ključeve pre njega i ključeve posle njega
  - Ključ iz sredine prebacujemo u roditeljski čvor
  - Od preostala dva niza ključeva pravimo dva nova čvora, na koje će pokazivati roditeljski čvor
  - **Napomena** — prebacivanje ključeva može dovesti do overflow-a u roditeljskim čvorovima, rekurzivno delimo čvorove sve do korena

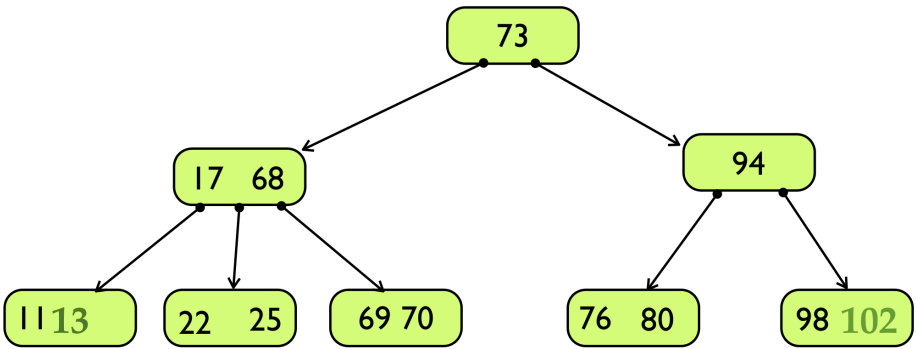
Primer - Rotacija



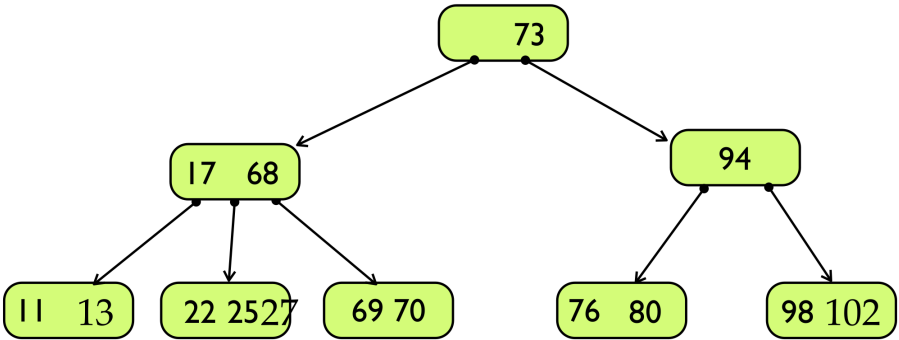
Primer - Rotacija



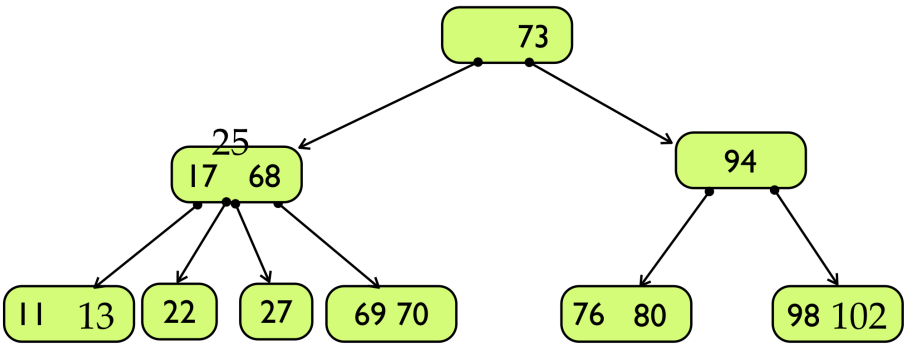
Primer - Rotacija



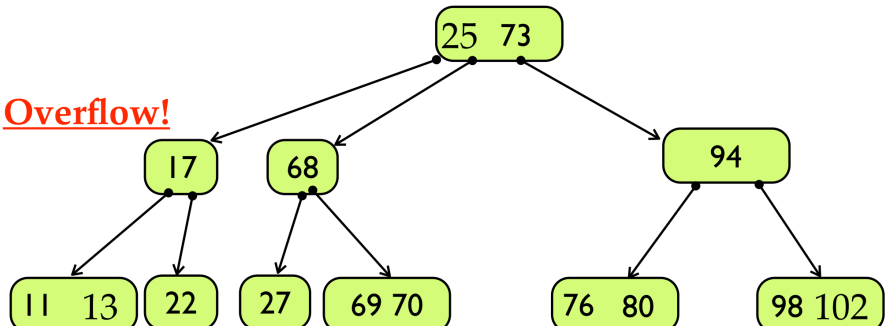
Primer - Podela čvorova



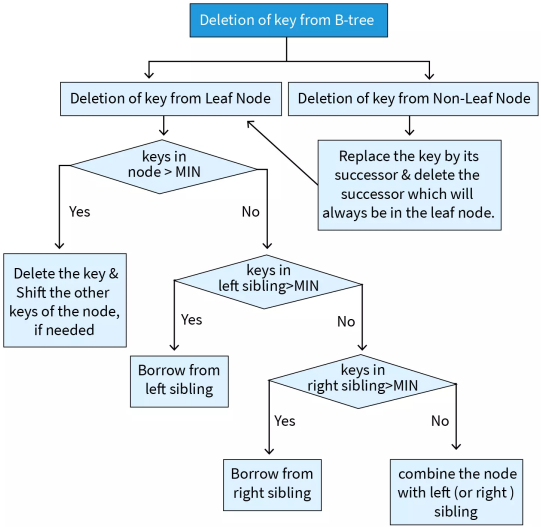
Primer - Podela čvorova



Primer - Podela čvorova

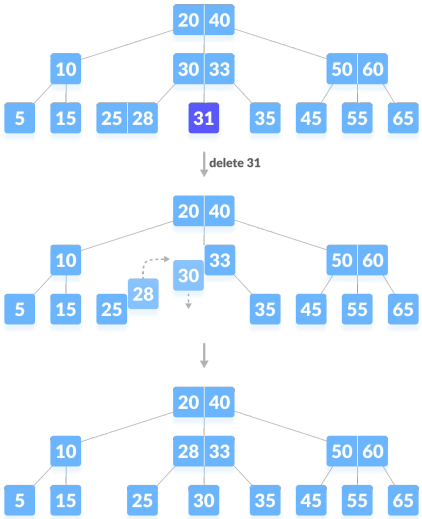


Brisanje


Deletion of key from B-tree
Deletion of key from Leaf Node
Deletion of key from Non-Leaf Node
keys in node > MIN
Yes
No
Replace the key by its successor & delete the successor which will always be in the leaf node.
Delete the key & Shift the other keys of the node, if needed
keys in left sibling>MIN
Yes
No
Borrow from left sibling
keys in right sibling>MIN
Yes
No
Borrow from right sibling
combine the node with left (or right) sibling

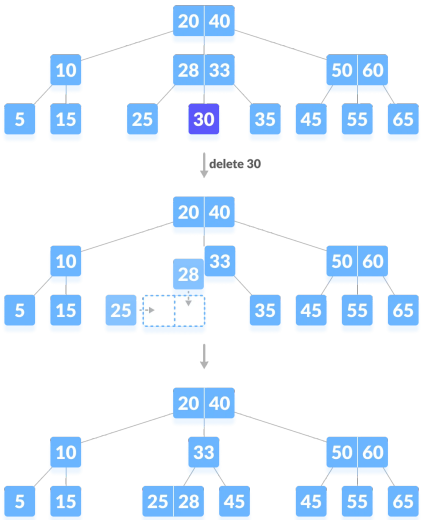
Pozajmljivanje iz sibling čvora

- Kada uzimamo iz levog sibling-a:
  - Preuzimamo ključ iz roditeljskog čvora koji je prethodnik čvora obrisanog ključa
  - Na ispražnjeno mesto u roditeljskom čvoru dodajemo najveći ključ iz levog sibling-a
- Kada uzimamo iz desnog sibling-a:
  - Preuzimamo ključ iz roditeljskog čvora koji je sledbenik čvora obrisanog ključa
  - Na ispražnjeno mesto u roditeljskom čvoru dodajemo najmanji ključ iz desnog sibling-a
- **Napomena 1:** sibling se ne mora nalaziti neposredno pored čvora i u tom slučaju pozajmljivanje se obavlja u više koraka



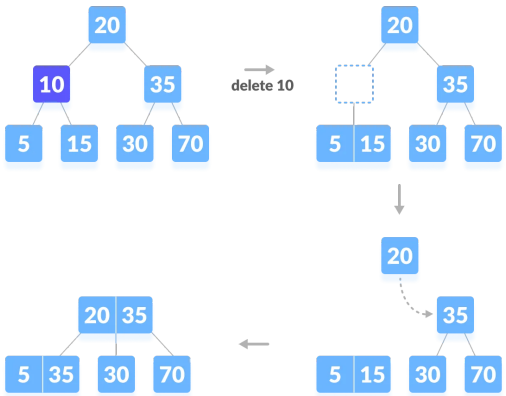
Spajanje čvorova

- Strategiju koristimo kada i levi i desni sibling čvorovi imaju minimalan broj ključeva
- Ključ iz roditeljskog čvora, koji je prethodnik ili sledbenik čvora obrisanog ključa, prebacujemo na poslednje mesto levog čvora
- Sve elemente desnog čvora dodajemo na kraj levog čvora
- Uklanjamo premešteni ključ iz roditeljskog čvora, kao i sada prazan desni čvor



Spajanje čvorova

- **Napomena:** Spajanjem čvorova roditelj može spasti na broj ključeva manji od minimuma
- Moramo rekurzivno proći kroz roditeljske čvorove, sve do korena, i popuniti čvorove koji nemaju dovoljno ključeva
- **Napomena:** ako pozajmljujemo iz sibling-a koji nije list, potrebno je da podstablo koje je ostalo "višak" prebacimo u čvor koji je imao manjka ključeva



Dodatni materijali

- Organization and maintenance of large ordered indices
- The Ubiquitous B-Tree
- B-Tree Visualization
- 2-3 Trees and B-Trees
- B+Tree Indexes

Zadaci

- Merkle stablo:
  - Implementirati Merkle stablo za proizvoljan skup podataka
  - Napisati funkcije za serijalizaciju i deserijalizaciju Merkle stabla
- B stablo
  - Implementirati B stablo strukturu podataka
  - Podržati operacije pretrage, dodavanja i brisanja elementa
  - Dodati funkciju koja vraća listu svih elemenata stabla, sortiranih u rastućem redosledu