Drevesni indeksi

Iztok Savnik, FAMNIT

Prosojnice & učbenik

- Učbenik:
  - Raghu Ramakrishnan, Johannes Gehrke, *Database Management Systems, McGraw-Hill, 3rd ed., 2007.*
- *Prosojnice:*
  - *From „Cow Book“: R.Ramakrishnan, http://pages.cs.wisc.edu/~dbbook/*

Podatkovni vpis **k***

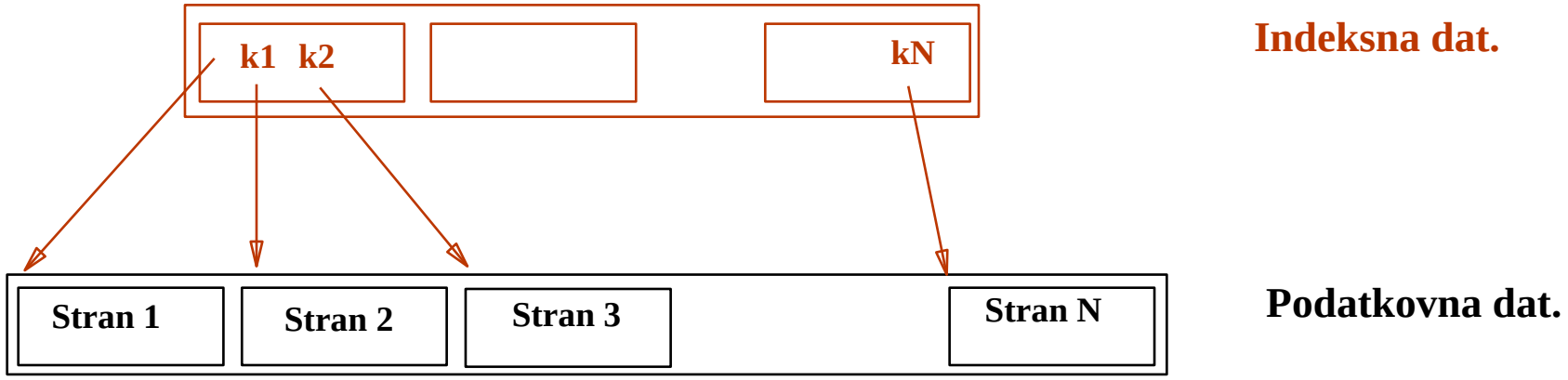
- Podatkovni vpis k* je lahko:
  - Podatkovni zapis z vrednostjo ključa **k.**
  - <**k**, rid>
  - <**k**, seznam rid>
- Izbira je ortogonalna izbiri indeksne tehnike uporabljene za iskanje pod. vpisov **k***
- Drevesno strukturirani indeksi podpirajo iskanje po enakosti in iskanje po področju.

Drevesni indeksi

- Drevesna indeksna struktura podpira tako
  - Iskanje področja kot tudi
  - Iskanje z enačajem
- *ISAM*:
  - Statična struktura
  - Sčasoma lahko postane drevo zelo neuravnoteženo
- *B+ tree*:
  - Dinamična struktura
  - Drevo je vedno uravnoteženo

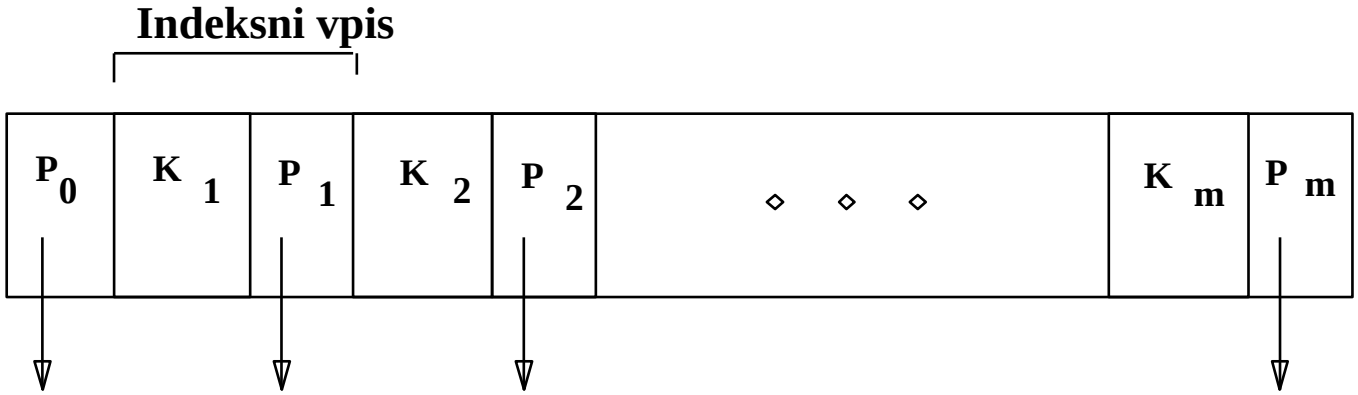
Iskanje področja

- ``*Poišči vse študente s povprečjem > 9.0*''
  - Če so podatki v urejenih datoteki naredimo binarno iskanje in nato pregled.
  - Cena binarnega iskanja je visoka.
- Ideja: Kreiraj ´´indeksno´´ datoteko.

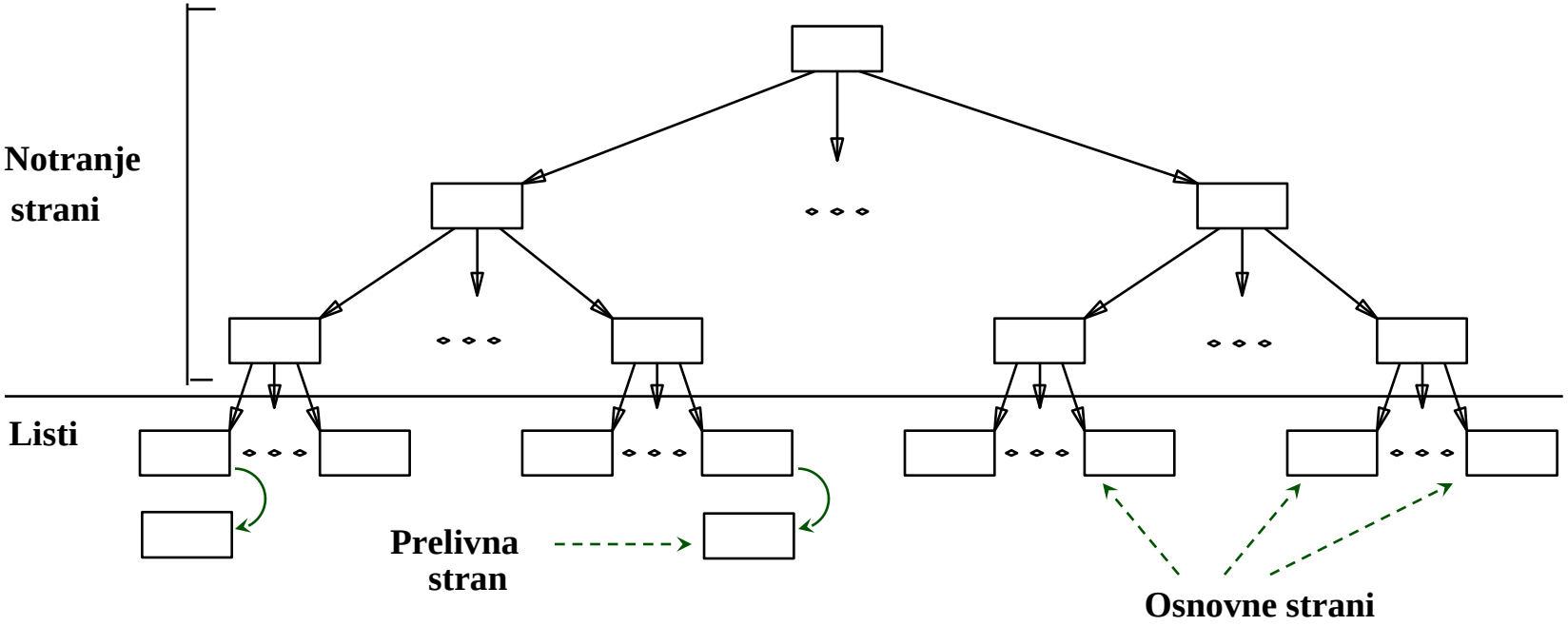


☛ *Iščemo lahko preko manjše indeksne datoteke!*

ISAM



- Indeksna datoteka je vseeno lahko precej velika. Idejo lahko ponovimo!



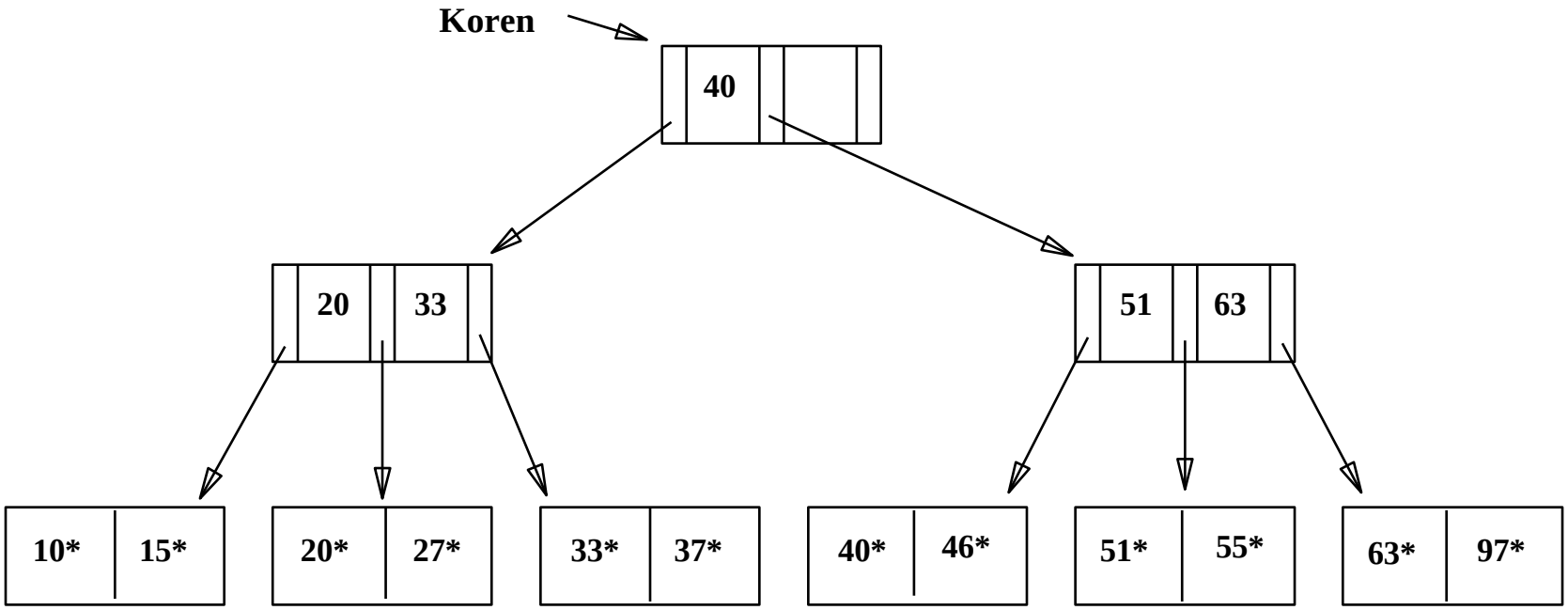
☛ *Listi (strani) vsebujejo podatkovne vpise.*

Komentarji o ISAM

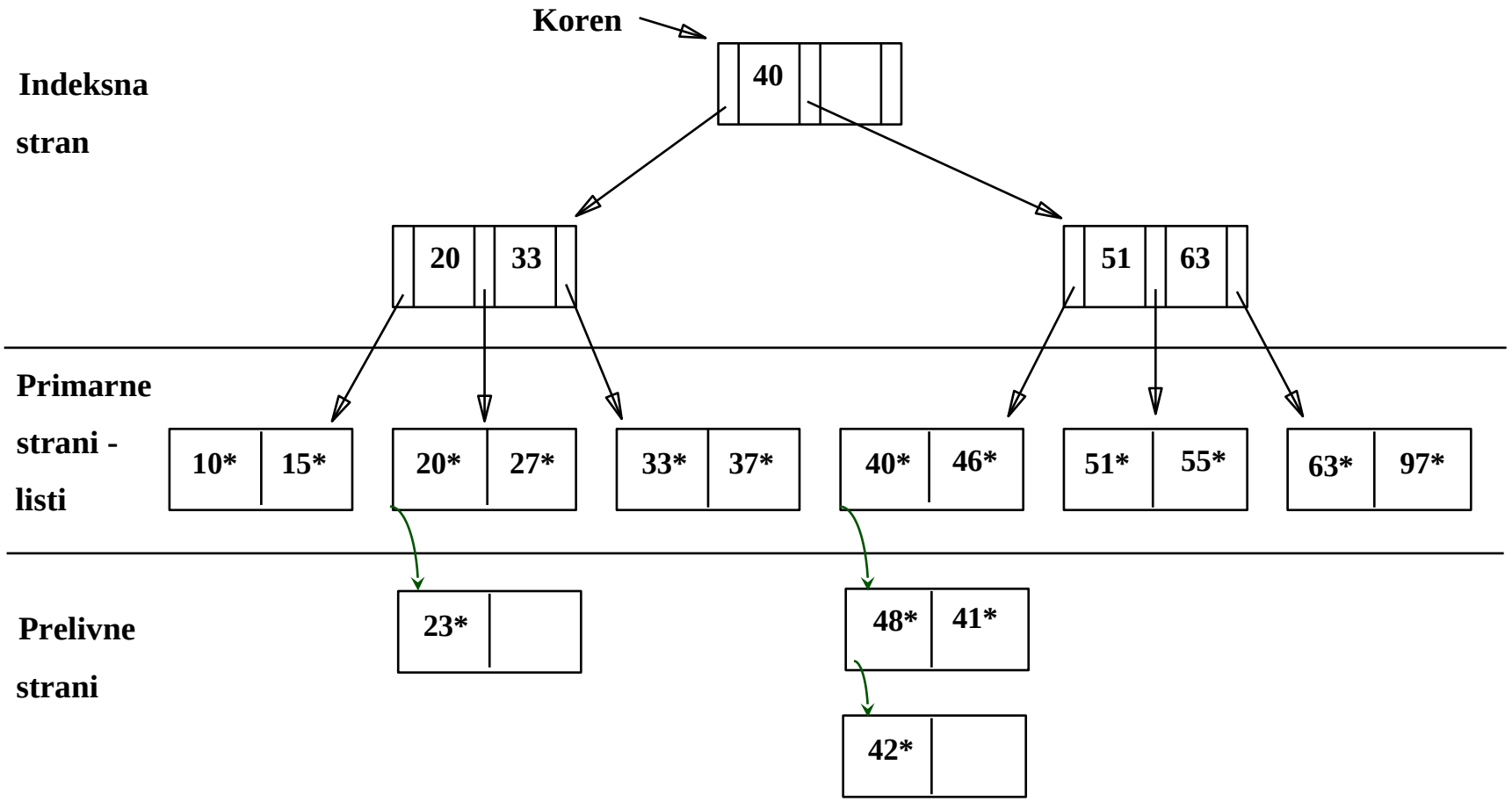
- Kreiranje datoteke: Liste zasežemo sekvenčno sortirane po iskalnem ključu;
  - Zasežemo indeksne strani iterativno po nivojih
  - Prostor za prelivne strani
- Indeksni vpisi: <vrednost isk.ključa, id strani> Usmerjajo iskanje podatkovnih vpisov v listih.
- *Iskanje*: Začni v korenu; primerjaj z iskalnim ključem za iskanje listov. Cena: $\log_F N$; F = # vnos/ind.stran, N = # listov (strani)
- *Dodaj*: Poišči list kamor spada podatkovni vpis in ga dodaj.
- *Brisanje*: išči z isk.ključem; izbriši pod.vpis v listu; če je prelivna stran prazna jo sprosti.
  - Statična struktura: vnosi/brisanja spreminjajo samo liste!

Primer ISAM drevesa

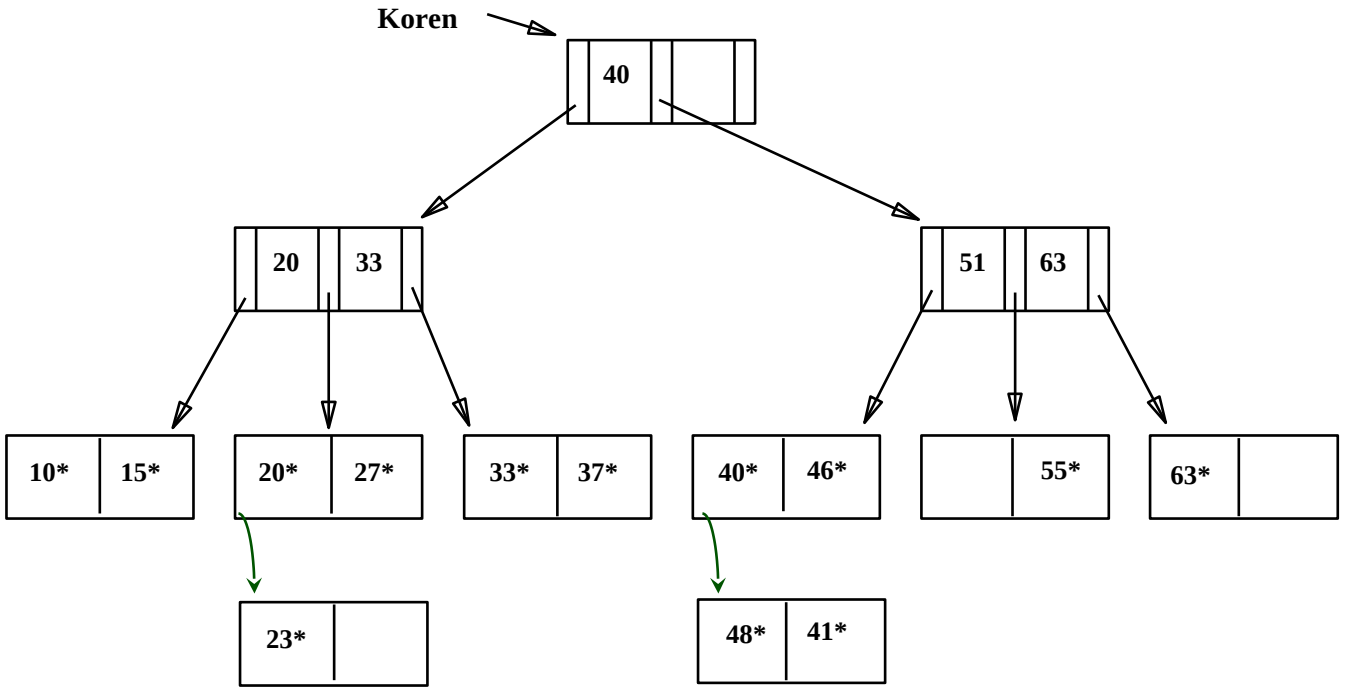
- Vsako vozlišče lahko vsebuje dva vpisa; ni potrebe po kazalcih na naslednji list. (Zakaj?)



Po vstavljanju 23*, 48*, 41*, 42* ...



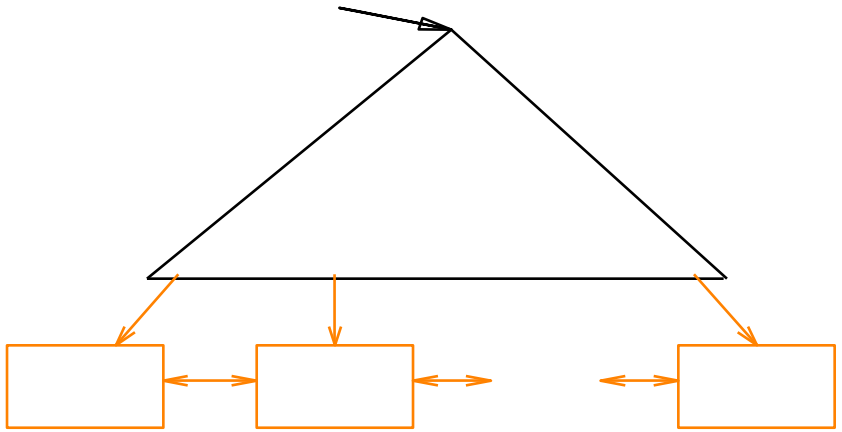
... potem brisanje 42*, 51*, 97*



☛ *Vpis 51* se pojavi v notranjih straneh in ne tudi v listih!*

B+ Drevo: Najbolj razširjen indeks

- Kompleksnost op. Vstavi/Izbriši = $\log_F N$
  (F = fanout, N = # listov)
- Drevo je zelo uravnoteženo.
- Indeksne strani so zapolnjene vsaj 50%(razen korena). Vsako vozlišče vsebuje **d** <= *m* <= 2**d** vpisov.
  Parameter **d** imenujemo *stopnja* odrevesa.
- Učinkovito podpira iskanje po enakosti in iskanje področja.

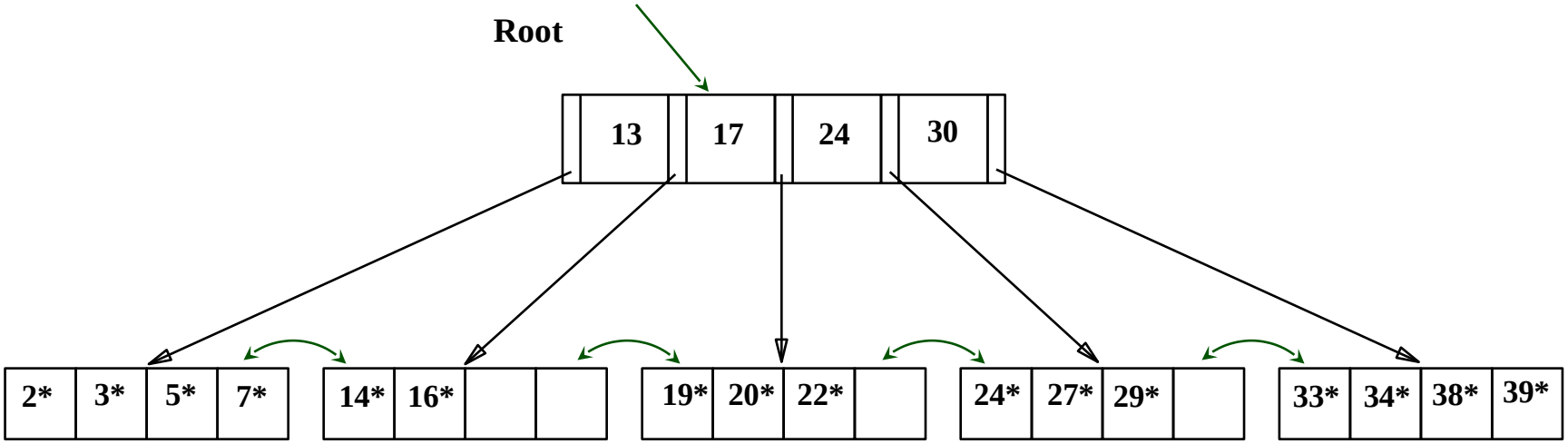


Indeksni vpis
(Usmerjajo iskanje)

Podatkovni vpisi
(Referenca/podatki)

Primer B+ Drevesa

- Iskanje se začne v korenu; primerjava ključev vodi do lista s pod. vpisom (kot pri ISAM).
- Išči 5*, 15*, vse pod. vpise >= 24* ...



☛ *Iz iskanja 15* vemo, da ključa ni v drevesu!*

B+ Drevesa v praksi

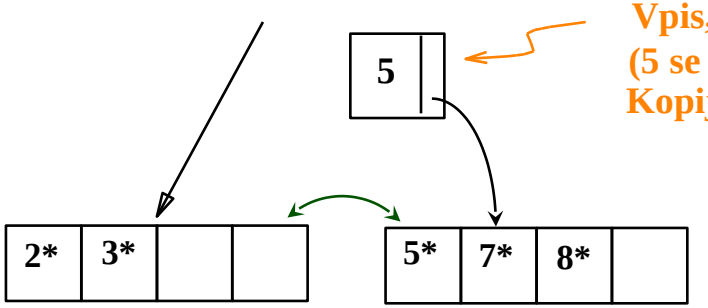
- Tipična stopnja: 100. Tipična zapolnjenost: 67%.
  - Povprečen fanout = 133
- Tipične kapacitete:
  - Višina 4: 133^4 = 312,900,700 zapisov
  - Višina 3: 133^3 = 2,352,637 zapisov
- Večinoma lahko shranimo višje nivoje drevesa v vmesni pomnilnik:
  - Nivo 1 = 1 stran = 8 Kb
  - Nivo 2 = 133 strani = 1 Mb
  - Nivo 3 = 17,689 strani = 133 MB

Vstavljanje pod. vpisa v B+ drevo

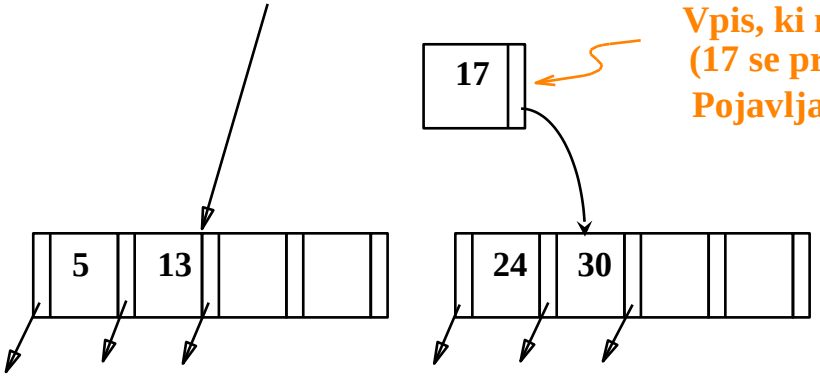
- Poišči list *L.*
- Vstavi pod. vpis v *L*.
  - Če ima *L* zadosti prostora, je narejeno!
  - Sicer je potrebno *razcepiti L (v L in novo vozlišče L2)*
    - Enakomerno porazdeli vpise in **prepiši** srednji ključ gor.
    - Vstavi indeksni vpis, ki kaže na *L2,* v starša od *L*.
- To se lahko zgodi rekurzivno
  - Razcep indeksnega vozlišča: enakomerno porazdeli vpise in **prenesi** srednji ključ gor. (Razlika z razcepom lista.)
- Razcepi "širijo" drevo; razcep korena zviša drevo.

Vstavljanje 8* v B+ drevo (primer)

- Minimalna zasedenost po razcepitvi je zajamčena v listih kot tudi v indeksnih straneh.
- Poglej razliko med prepisom in prenosom ključa navzgor.
- Kakšen je razlog za razliko?

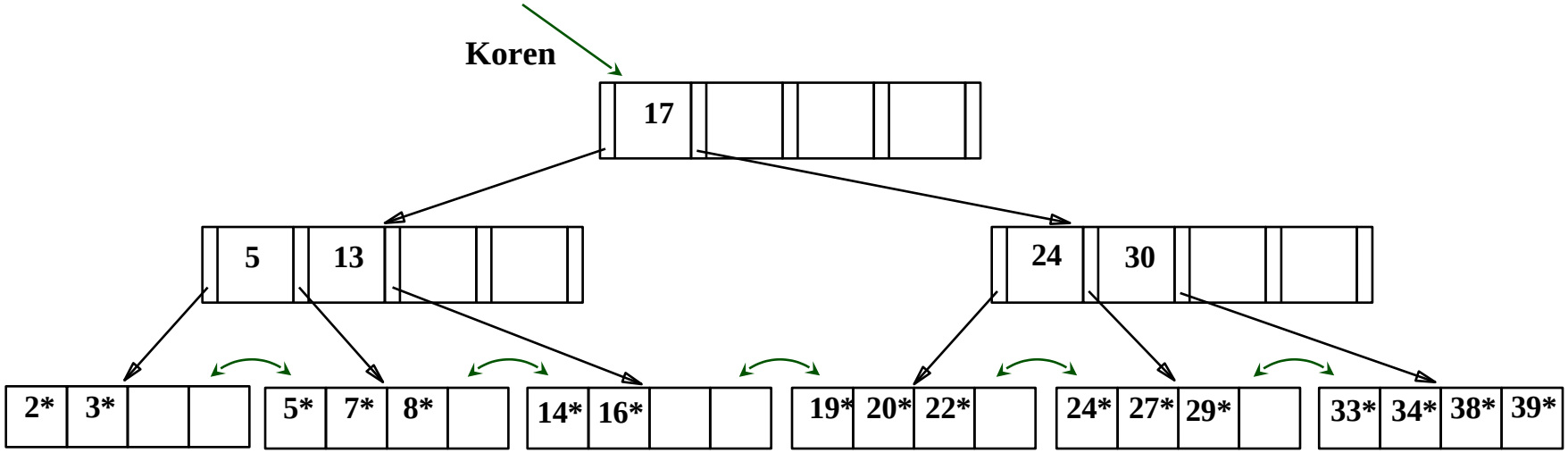


Vpis, ki naj se vstavi v starša (5 se prepiše navzgor; Kopija ostane v listu).



Vpis, ki naj se vstavi v starša (17 se prenese navzgor in se Pojavlja v indeksu samo enkrat).

Primer B+ drevesa po vnosu 8*

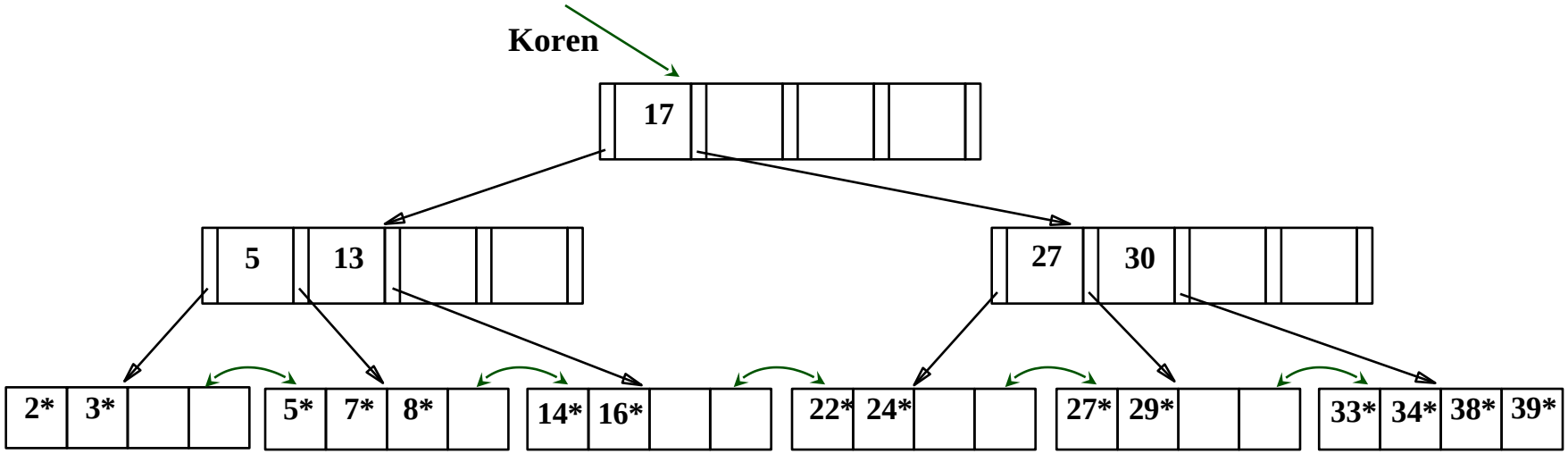


- Vidimo, da se je razcepil koren in se je povečala višina drevesa.
- V tem primeru bi se lahko izognili razcepu strani z porazdelitvijo vpisov; to se običajno ne uporablja v praksi.

Brisanje pod. vpisa iz B+ drevesa

- Začni pri korenu in poišči list z vpisom.
- Izbriši vpis.
  - Če je L vsaj polovico poln potem *končaj!*
  - Če vsebuje L **d-1** vpisov,
    - Poskusimo porazdeliti vpise tako, da si jih sposodimo od sosedov.
    - Če porazdelitev ne uspe, *zlij L in soseda.*
- V primeru zlitja moramo zbrisati vpis (ki kaže na *L* ali na soseda) iz starša od *L*.
- Zlivanje se lahko nadaljuje vse do korena; v tem primeru se zniža višina drevesa.

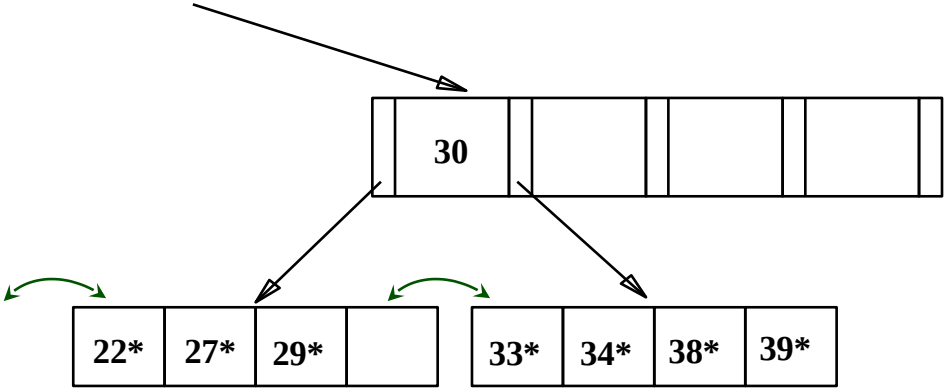
Drevo po vstavljanju 8* in brisanju 19* in 20* ...

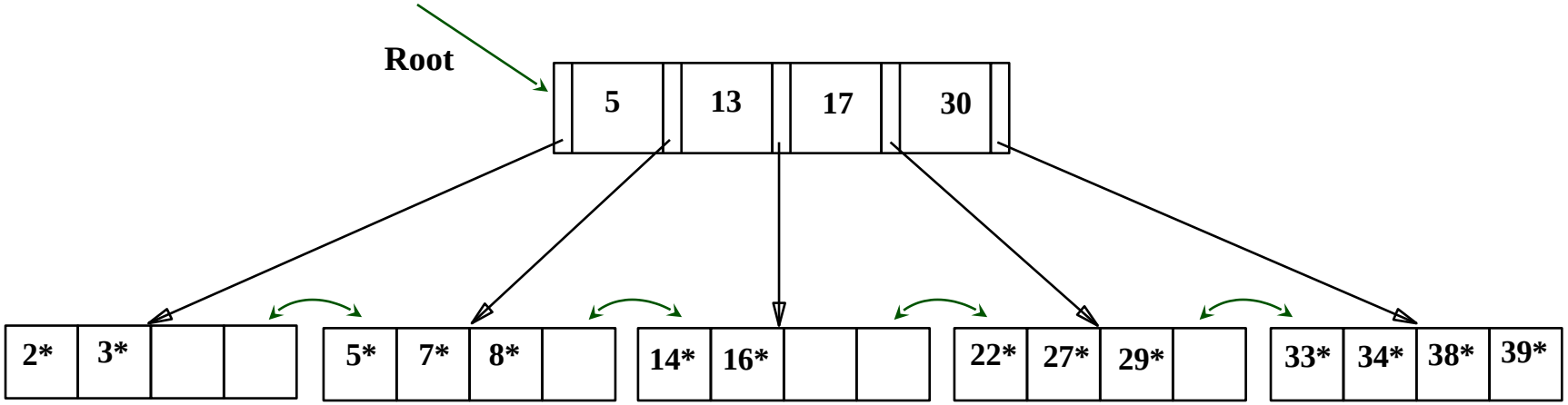


- Brisanje 19* je enostavno.
- Brisanje 20* je narejeno s porazdelitvijo vpisov. Srednji ključ je prepisan navzgor.

... in brisanje 24*

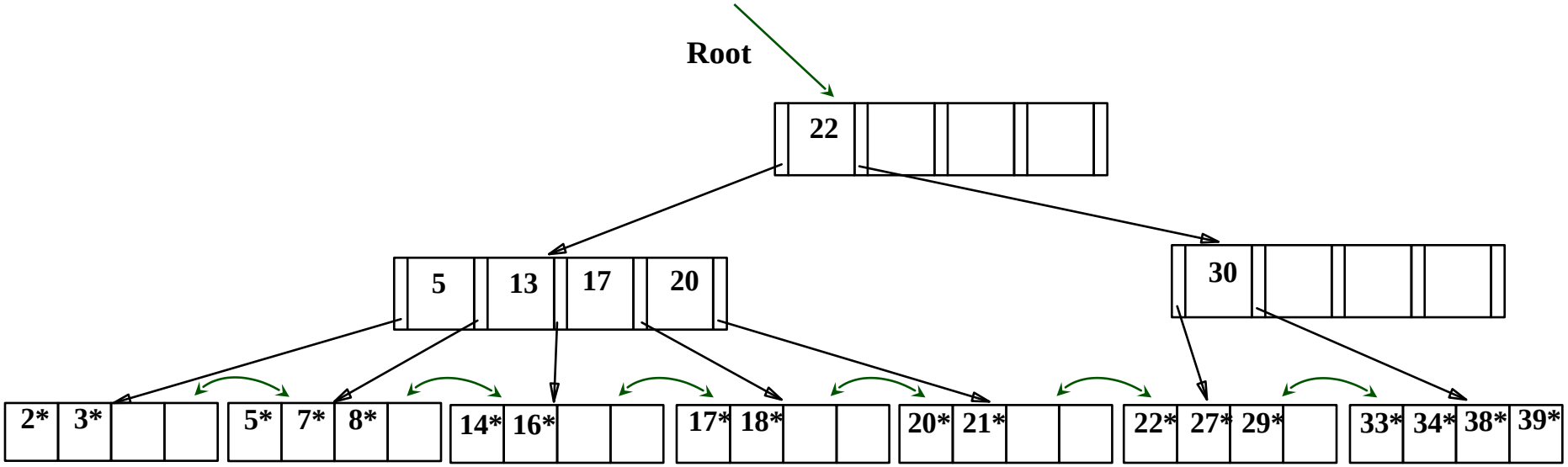
- Potrebno zlivanje.
- Zlitje podatkovnih vpisov na desni ter indeksnih vpisov spodaj.





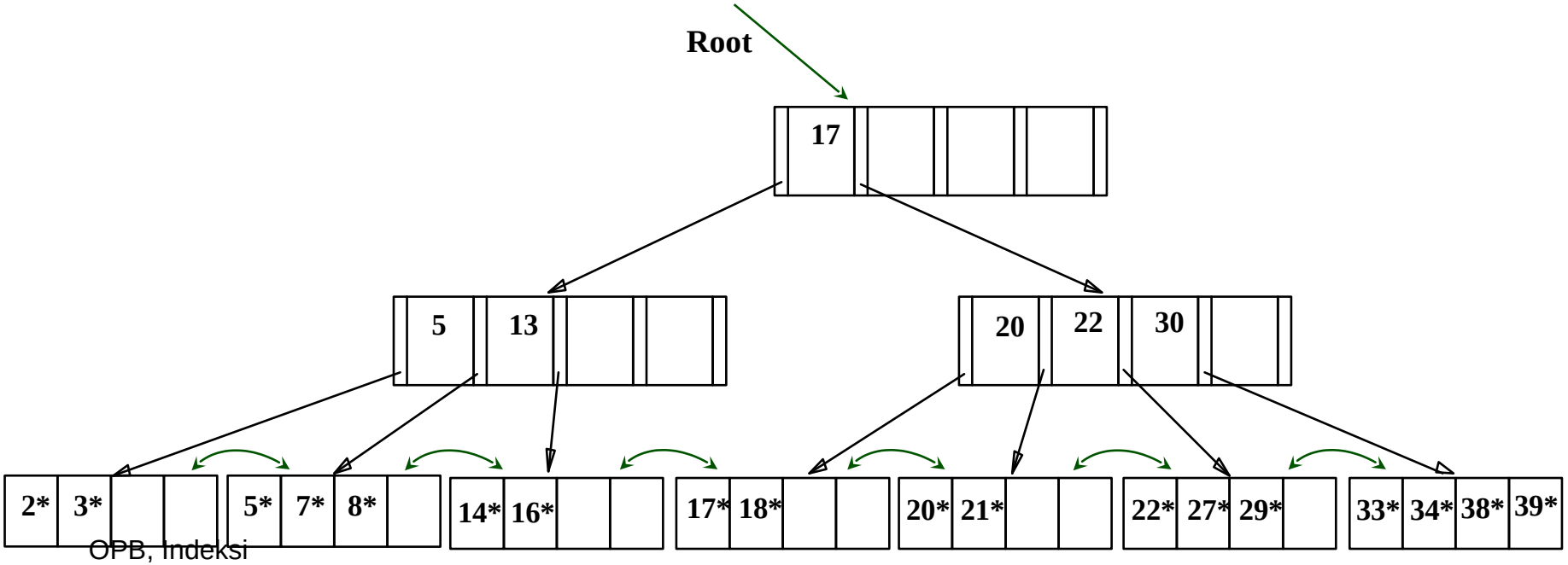
Primer prerazporeditve notranjega vozlišča

- Drevo spodaj: med brisanjem 24*. (Kaj bi lahko bilo začetno drevo?)
- Nasprotno prejšnjem primeru lahko porazdelimo vpise iz levega otroka korena desnemu otroku.



Po prerazporeditvi

- Intuitivno so vpisi prerazporejeni s `potiskanjem skozi' vmesni vpis v korenu.
- Zadošča, da se prerazporedi indeksni vpis s ključem 20; prerazporedili smo tudi 17.

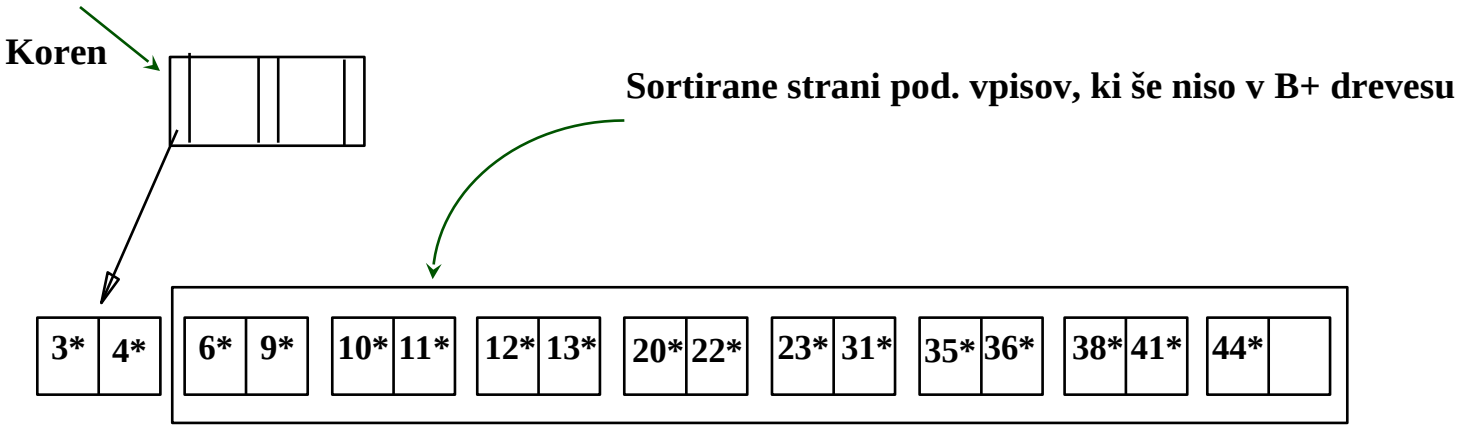


Stiskanje s predponami ključev

- Pomembno je povečati fan-out. (Zakaj?)
- Vrednosti ključev v indeksnih vpisih samo `usmerjajo promet'; pogosto jih lahko stisnemo.
  - Recimo, da vsebujejo sosedni indeksni vnosi iskalne ključe *Dannon Yogurt*, *David Smith in Devarakonda Murthy.* Ključ *David Smith* lahko okrajšamo v *Dav*. (Preostali ključi se tudi lahko stisnejo).
    - Je to korektno? Ne čisto! Kaj če imamo še podatkovni vnos *Davey Jones*? (Lahko skrčimo *David Smith v Davi*)
    - V splošnem, med stiskanjem je potrebno izbrati ključ, ki je večji od vseh ključev na levi strani drevesa (od danega ključa).
- Vstavljanje/brisanje mora biti primerno prilagojeno.

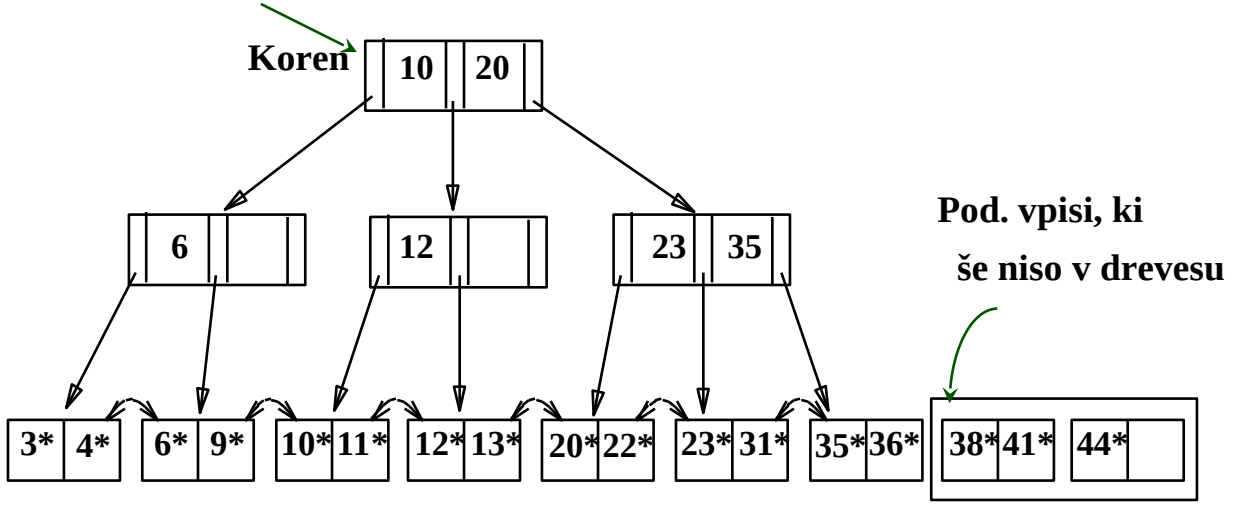
Masovno polnjenje B+ drevesa

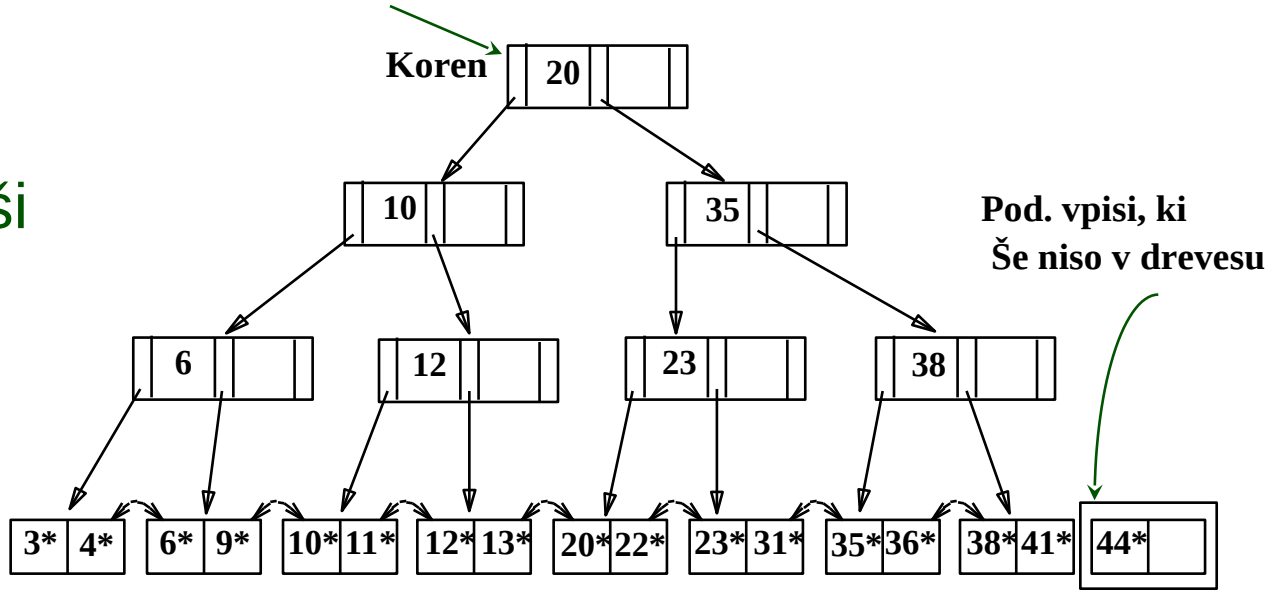
- Polnjenje B+ drevesa z vstavljanjem posameznega zapisa je lahko zamudno v primeru, da je podatkov veliko.
- *Polnjenje* lahko naredimo veliko hitreje.
- *Inicializacija*: Uredi podatkovne vpise, dodaj kazalec na prvo stran v novem korenu.



Masovno polnjenje

- Indeksni vpisi, ki kažejo na liste so vedno vstavljeni na skrajno desno indeksno stran tik nad listi. Ko se ta napolni jo razcepimo – cepitev se lahko ponovi na višjih nivojih.
- Algoritem je hitrejši od vstavljanja zapisov.





Povzetek polnjenja B+ drevesa

- Možnost 1: iterativno vstavljanje.
  - Počasno.
  - Ne dobimo sekvenčen zapis indeksnih strani po nivojih.
- Možnost 2: *Polnjenje (Bulk Loading)*
  - Prednosti pri kontroli sočasnega dostopa.
  - Manj I/O operacij med gradnjo.
  - Listi so shranjeni sekvenčno.
  - Lahko kontroliramo “faktor zapolnjenosti” na straneh.

O `stopnji' vozlišča

- *Koncept stopnja* (**d**) se zamenja s kriterijem, ki temelji na zapolnjenosti strani: `vsaj do polovice polna' stran.
  - Indeksne strani lahko tipično shranjujejo veliko več vnosov kot strani med listi drevesa.
  - Ključi variabilne dolžine: različna vozlišča shranjujejo različno število vnosov.
  - Tudi če imamo ključe fiksnih velikosti in uporabimo alternativo (3) dobimo variabilno število indeksnih in podatkovnih vnosov po različnih straneh.

Povzetek

- Drevesni indeksi so idealni za iskanje področij vendar dobri tudi za iskanje z enačajem.
- ISAM je statični indeks.
  - Spreminjajo se samo listi; potrebne so prelivne strani.
  - Prelivne strani lahko zelo poslabšajo učinkovitost strukture.
- B+ drevo je dinamična struktura.
  - Operaciji vnos/brisanje ohranjata uravnoteženost drevesa; cena operacij je $\log_F N$ cost.
  - Velik fanout (**F**) omogoča da višina drevesa redko presega 3 ali 4.
  - Drevo je skoraj vedno boljše od urejene datoteke.

Povzetek

- B+ drevo
  - Tipično so indeksne strani zapolnjene 67%.
  - B+ drevo je običajno boljše od ISAM.
  - Če so podatkovni vpisi enaki podatkovnim zapisom potem cepitve listov spremenijo identifikatorje zapisov!
- Polnjenje B+ drevesa je lahko veliko hitrejše kot vstavljanje posameznih zapisov v drevo.
- B+ drevo je najbolj razširjen indeks v sistemih za delo s podatkovnimi bazami.
  - Je tudi ena od najbolj optimiziranih komponent SUPB.