# Zgradba tekstovne podatkovne zbirke

Zbirke brez organizacije iskalnih ključev,

invertirane strukture,

drevesa,

signature.

# Arhitekture zbirk - uvod

- Tematika arhitektura tekstovne podatkovne zbirke govori o tem, kako so
  - organizirani dokumenti v zbirki,
  - kakšne so podatkovne strukture, ki sestavljajo dokumente,
  - kako so urejeni načini dostopa do dokumentov…

- S to tematiko je tesno povezana učinkovitost algoritmov, ki opravljajo operacije nad dokumenti.

# Arhitekture zbirk - uvod

- Pri načrtovanju informacijskega sistema je odločitev o arhitekturi zbirk ena od pomembnih odločitev, ki v dobršni meri usmerja nadaljnje postopke.
- Kreator informacijskega sistema se odloča na osnovi
  - lastnosti podatkov, ki bodo shranjeni v zbirkah,
  - načrtovanih načinov njihove uporabe,
  - značilnosti uporabnikov sistema,
  - stopnje povezanosti sistema z okolico…

# Logični in fizični opis arhitekture

Različna pogleda na organizacijo podatkov.

- Fizični opis se ukvarja z namestitvijo podatkov na pomnilniškem mediju.
- Na nivoju fizičnega opisa rešujemo probleme, povezane predvsem z optimizacijo dostopa do podatkov.
- Problemi fizičnega opisa so, bolj kot od narave podatkov, odvisni od konkretne strojne opreme.

# Logični in fizični opis arhitekture

- Logični opis arhitekture zbirke se ukvarja s samimi podatki in ne abstraktnimi lokacijami na pomnilniku.
- Ukvarja se s problematiko njihove organizacije, ki bo najbolj ustrezala iskalnim (in ostalim) algoritmom.
- Logični opis je neodvisen od konkretne strojne opreme.

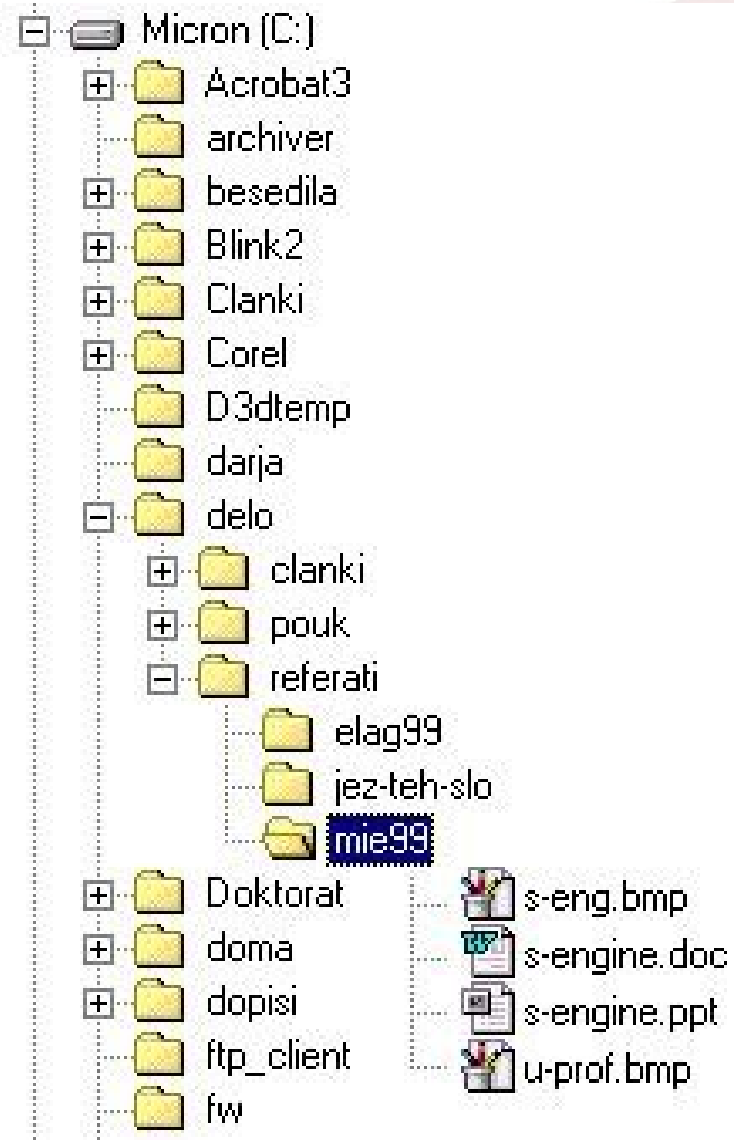
# Arhitekture zbirk - uvod

- Pomembna lastnost arhitekture tekstovne zbirke je organizacija iskalnih ključev.

- V splošnem ločimo
  - arhitekture brez posebne organiziranosti iskalnih ključev, in
  - arhitekture z iskalnimi ključi, organiziranimi v ločenih strukturah.

# Zgodovina: dokumenti v samostojnih datotekah

- Datoteke, tudi dokumenti, so na vsakem računalniku urejene v skladu z operacijskim sistemom.
- Tudi operacijski sistem je na nek način zbirka.



# Zgodovina: dokumenti v samostojnih datotekah

- Operacijski sistem opravlja nekatere funkcije, ki jih pričakujemo od zbirke.
- Obstajajo orodja, ki pregledujejo datoteke in iščejo želena znakovna zaporedja.
- Taka orodja delujejo po načelu prepoznavanja vzorcev.
- Primer: program grep, ki je del operacijskega sistema UNIX.

# Zgodovina: dokumenti v samostojnih datotekah

- Programi za iskanje vzorcev so zelo učinkoviti, vendar tak način iskanja po datotekah ne more učinkovito nadomestiti prave zbirke.
- Programi za iskanje vzorcev morajo pregledati vse datoteke. Porabljeni čas narašča linearno s številom in velikostjo datotek.
- Prednost: dokumenti so nespremenjeni in so zato uporabni z izvorno programsko opremo.

# Zgodovina: dokumenti v sekvenčni datoteki

Sekvenčna (zaporedna) datoteka.

- Dokumenti so združeni v skupni datoteki in tako izgubijo samostojnost.
- Dokumenti so shranjeni v datoteki zaporedno - sekvenčno.
- Dokumenti so na nek način urejeni:
  - najenostavnejši način je kronologija vnosa,
  - lahko so urejeni po avtorjih, abecedi naslovov…

# Zgodovina: dokumenti v sekvenčni datoteki

- Dokumenti so lahko strukturirani v polja in podpolja.
- Strukturiranost omogoča izdelavo pomožnih podatkovnih struktur, ki so integralni del datoteke.
- Taka struktura je informacija o poziciji polj v dokumentu in poziciji dokumenta v datoteki - zbirki.
- Pri iskanju po določenem polju taka informacija omogoča preskakovanje nerelevantnih delov dokumentov.

# Zgodovina: dokumenti v sekvenčni datoteki

- Pri iskanju je treba pregledati vse zapise.
- Čas, potreben za iskanje je linearno odvisen od števila in dolžine dokumentov.
- Če obstajajo pomožne strukture, se čas, potreben za iskanje, bistveno skrajša.

# Zgodovina: dokumenti v sekvenčni datoteki

- Zbirke kot sekvenčne datoteke so najstarejša oblika tekstovnih zirk.
- Sekvenčna datoteka je edina možna arhitektura zbirke, če je ta nameščena na magnetnem traku.
- Danes jih ne uporabljamo več, razen za zelo majhne količine podatkov.

# Zbirke z ločeno organiziranimi iskalnimi ključi

- Posebne strukture, v katerih so organizirani iskalni ključi, ima velika večina sodobnih arhitektur podatkovnih zbirk.
- Tako obstajata najmanj dve datoteki:
  - datoteka z dokumenti,
  - datoteka z iskalnimi ključi.

# Zbirke z ločeno organiziranimi iskalnimi ključi

- Dokumenti so v svoji datoteki najpogosteje urejeni le po kronologiji vnosa.
- Različne arhitekture se ločijo predvsem po strukturah v datoteki(kah) z iskalnimi ključi

# Zbirke z ločeno organiziranimi iskalnimi ključi

- Iskalni ključi so izolirani iz dokumentov in na nek način urejeni v samostojnih datotekah.
- Iskalni ključi:
  - bibliografske zbirke: imena avtorjev, deskriptorji…
  - zbirke polnih dokumentov: neblokirani besedni krni
- Iskanje poteka po datoteki z iskalnimi ključi, ki je
  - običajno krajša od datoteke z dokumenti in
  - urejena na način, ki omogoča hitro iskanje.

# Zbirke z ločeno organiziranimi iskalnimi ključi

- Pri iskanju iskalni algoritmi
  - pregledujejo datoteko z iskalnimi ključi,
  - izberejo identifikacije dokumentov, ki ustrezajo iskalni zahtevi,
  - v datoteki z dokumenti naberejo dokumente z znanimi idenifikacijami in jih ponudijo iskalcu.

# Zbirke z ločeno organiziranimi iskalnimi ključi

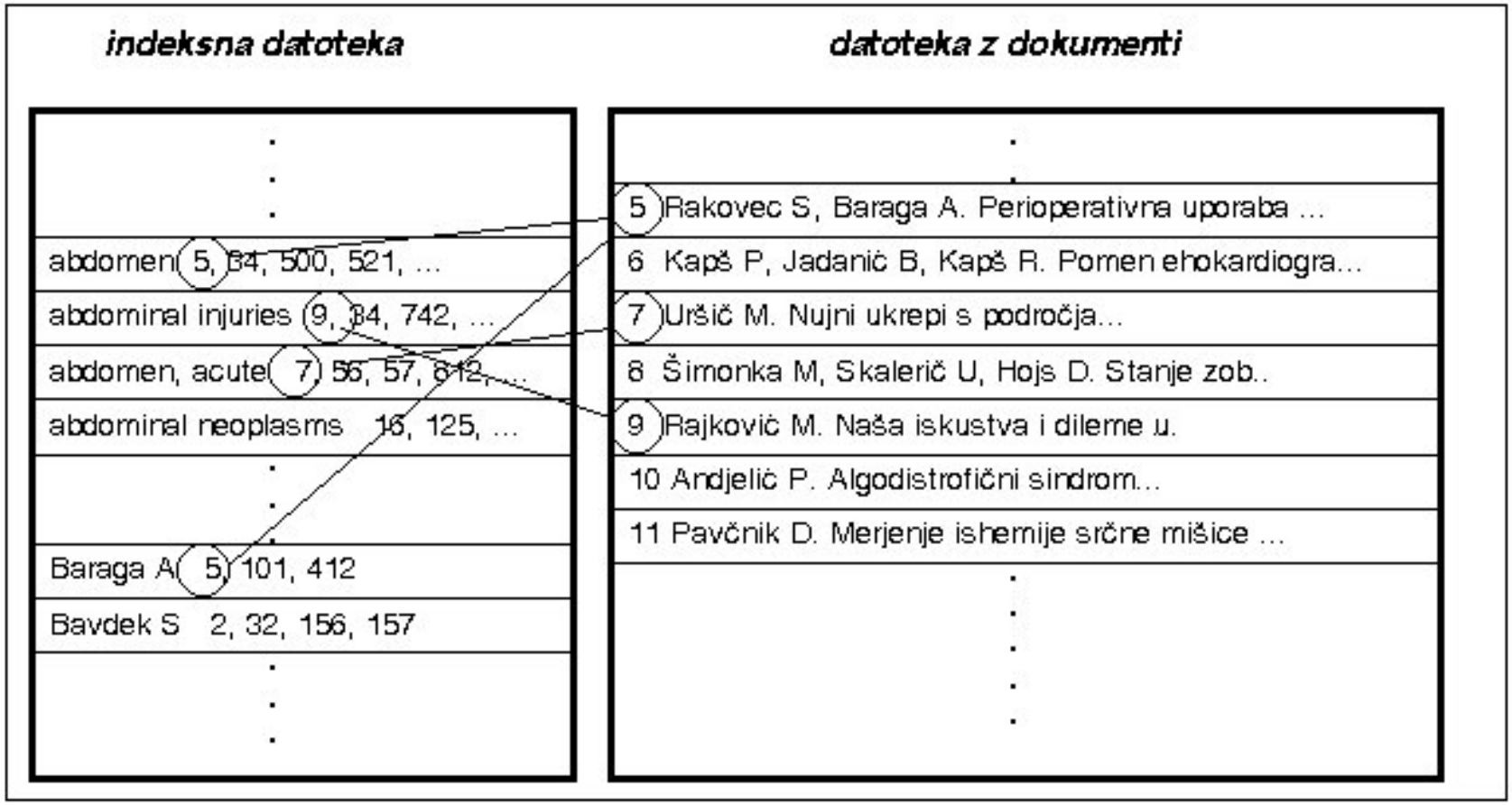
- Za razumevanje arhitekture zbirke je najpomembnejše razumevanje struktur z iskalnimi ključi.
- Ogledali si bomo tri najpogostejše oblike struktur:
  - invertirane datoteke,
  - drevesa,
  - signature.

# Invertirana arhitektura

- Datoteka z iskalnimi ključi ima podobno vlogo, kot kazalo (indeks) v knjigi, zato jo imenujemo *indeksna datoteka*.
- Iskalni ključi v indeksni datoteki so urejeni po abecedi ali številskih vrednostih.
- Vsak iskalni ključ je opremljen z dodatnimi informacijami, med njim vsaj z identifikacijo dokumenta, iz katerega izvira.

# Invertirana arhitektura

- Poenostavljen primer invertirane arhitekture



# Invertirana arhitektura

- Vsak iskalni ključ se v indeksni datoteki pojavi samo enkrat.
- Če iskalni ključ izvira iz več dokumentov, morajo biti ključu dodane identifikacije (kazalci) vseh teh dokumentov.
- Vsebinski opis vsakega dokumenta je razpršen po vsej indeksni datoteki.

# Invertirana arhitektura

- Iskanje po indeksni datoteki je zelo hitro.
- Najpogosteje se uporablja kakšna od različic algoritma “*z razpolavljanjem*”.

# Invertirana arhitektura

- Rezultat iskanja z vsakim iskalnim ključem je množica identifikacij dokumentov, iz katerih izvira iskalni ključ.
- Operacije z logičnimi operatorji so zelo enostavne:
  - če sta bila iskalna ključa povezana z operatorjem IN, je rezultat iskanja presek njunih množic identifikacij dokumentov...

# Invertirana arhitektura

- Invertirana arhitektura zelo primerna tudi za implementacijo iskanja z rangiranjem zadetkov.
- Povedne moči iskalnih ključev, ki so osnova rangiranju, so bile lahko izračunane pri oblikovanju zbirke in dodane ključem v indeksni datoteki.
- Tak način je primeren za statične zbirke.

# Invertirana arhitektura

- V dinamičnih zbirkah je potrebno sprotno računanje povednih moči med iskanjem.
- V dinamičnih zbirkah se frekvence iskalnih ključev pogosto spreminjajo.
- Kazalcem na dokumente so dodani podatki o frekvenci izraza v dokumentu,
- število kazalcev na dokumente pa neposredno pomeni število dokumentov s to besedo v zbirki (potrebno za računanje IDF).

# Invertirana arhitektura

- Izdelava indeksne datoteke.
- Primer: uvrstitev dveh dokumentov.

- Dokument 6:
  **Fizična struktura in logična struktura zbirk podatkov**.
- Dokument 9:
  **Podatkovni modeli v IR sistemih.**

| beseda | dok. # |
|---|---|
| fizič | 6 |
| strukt | 6 |
| logič | 6 |
| strukt | 6 |
| zbirk | 6 |
| podat | 6 |
| . | |
| . | |
| . | |
| podat | 9 |
| model | 9 |
| ir | 9 |
| sistem | 9 |

*sort* →

| beseda | dok. # |
|---|---|
| fizič | 6 |
| ir | 9 |
| logič | 6 |
| model | 9 |
| podat | 6 |
| podat | 9 |
| sistem | 9 |
| strukt | 6 |
| strukt | 6 |
| zbirk | 6 |

*odstran. duplikat.* →

| beseda | dok. # | frekv. |
|---|---|---|
| fizič | 6 | 1 |
| ir | 9 | 1 |
| logič | 6 | 1 |
| model | 9 | 1 |
| podat | 6 | 1 |
| podat | 9 | 1 |
| sistem | 9 | 1 |
| strukt | 6 | 2 |
| zbirk | 6 | 1 |

*računanje povedhih moči* →

| beseda | dok. # | PM |
|---|---|---|
| fizič | 6 | 2 |
| ir | 9 | 2 |
| logič | 6 | 2 |
| model | 9 | 2 |
| podat | 6 | 1 |
| podat | 9 | 1 |
| sistem | 9 | 2 |
| strukt | 6 | 4 |
| zbirk | 6 | 2 |

*gradnja indeksne datoteke* →

| beseda | število dokum. | kazalec1 |
|---|---|---|
| fizič | 1 | |
| ir | 1 | |
| logič | 1 | |
| model | 1 | |
| podat | 2 | |
| sistem | 1 | |
| strukt | 1 | |
| zbirk | 1 | |

| | | |
|---|---|---|
| 9 | 2 | kazalec2 |

| | | | | | |
|---|---|---|---|---|---|
| 6 | 2 | kazalec2 | 9 | 2 | kazalec2 |

| | | |
|---|---|---|
| 6 | 4 | kazalec2 |

***Razlaga:***

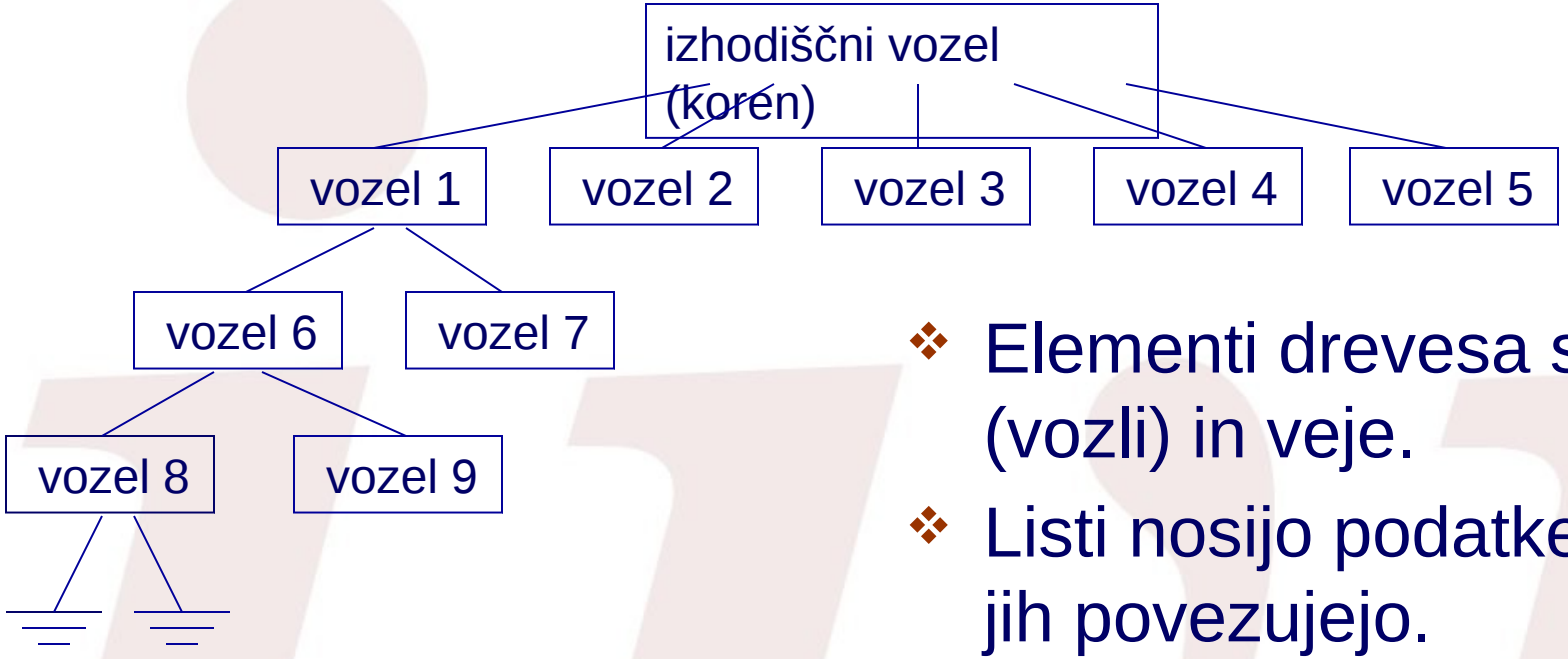
| | | |
|---|---|---|
| 9 | 2 | kazalec2 |

*identifikacija dokumenta* — *povedna moč besede* — *kazalec na lokacijo dokumenta v zbirki*

# Drevesa

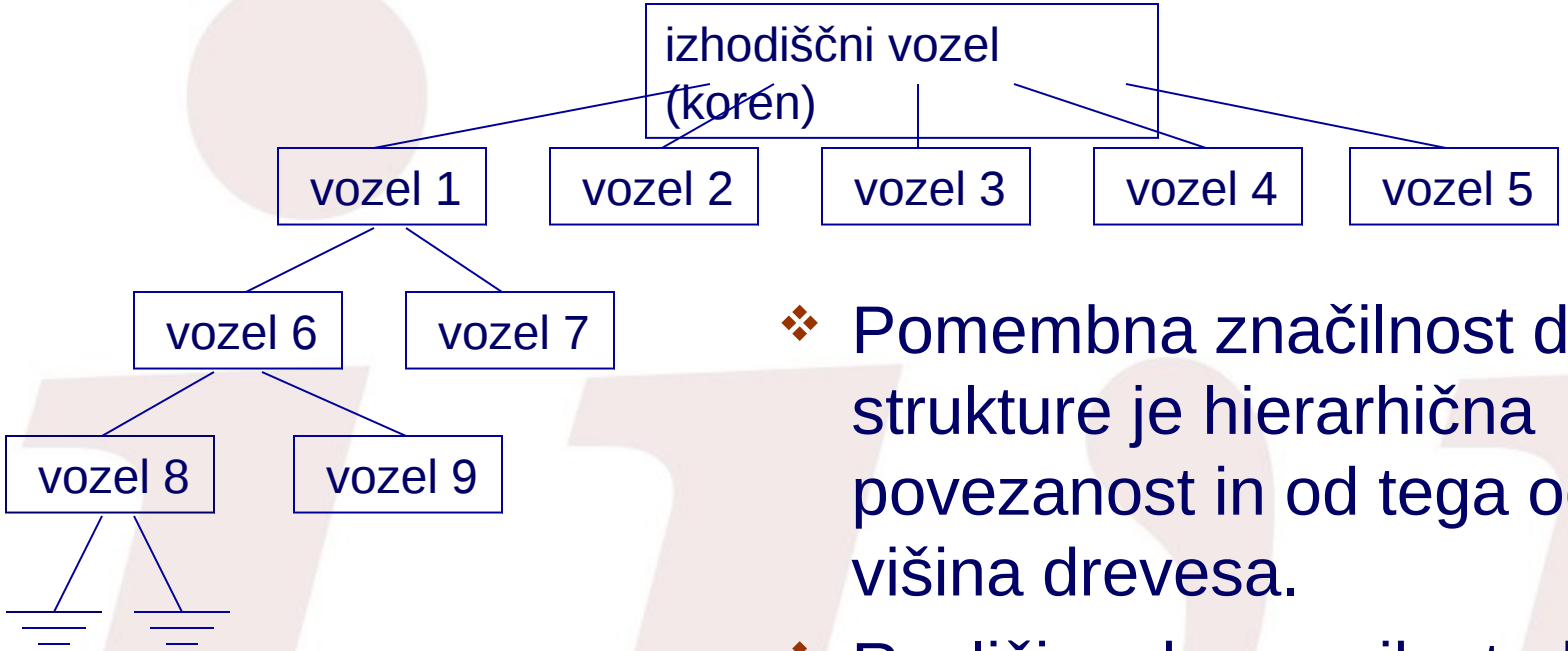
- Drevesne strukture se pogosto uporabljajo za shranjevanje in procesiranje različnih podatkov.
- Obstaja veliko zelo dognanih izpeljank.
- Drevesa so uporabna je tudi za shranjevanje iskalnih ključev v tekstovnih zbirkah.
- Med indeksno datoteko in drevesno strukturo je veliko podobnosti.

# Drevesa



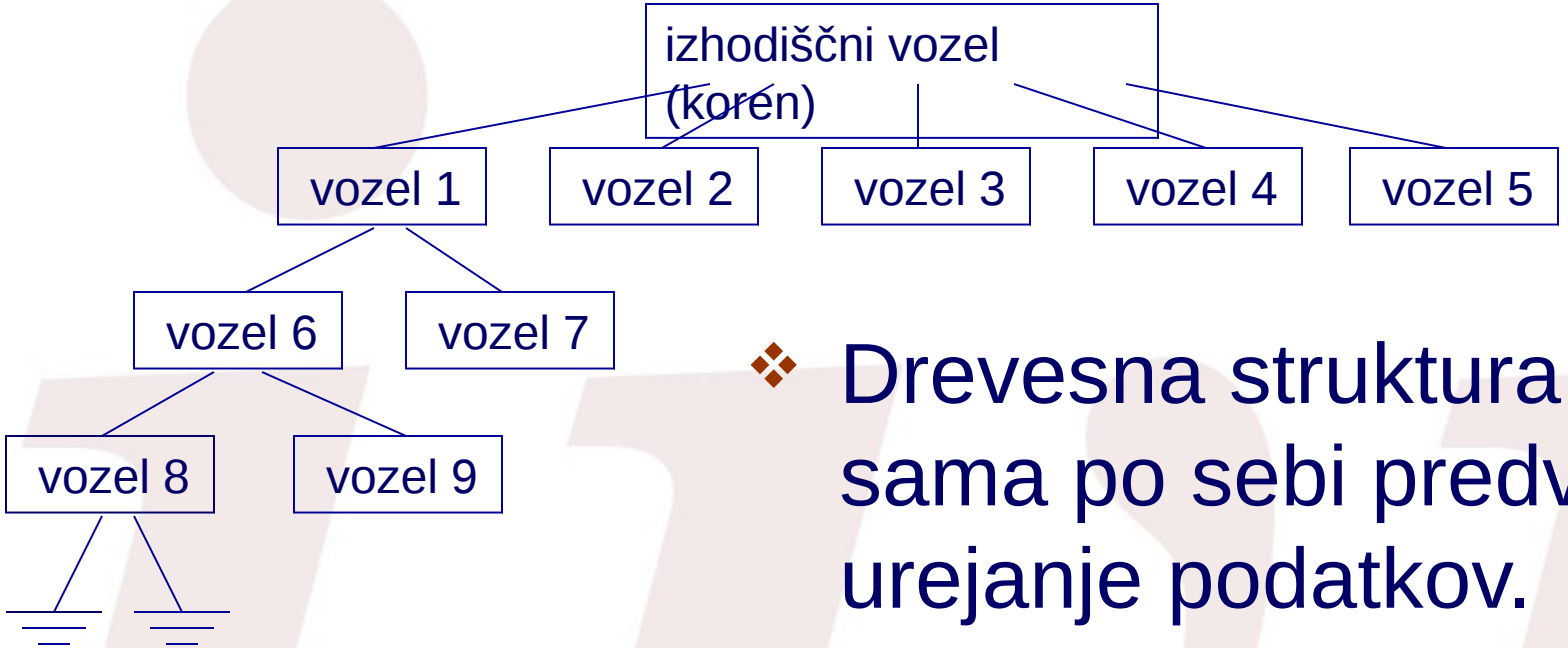

- Elementi drevesa so listi (vozli) in veje.
- Listi nosijo podatke, veje jih povezujejo.
- Poznamo notranje in zunanje (terminalne) liste.
- Terminalni listi so lahko prazni - slepe veje.

# Drevesa

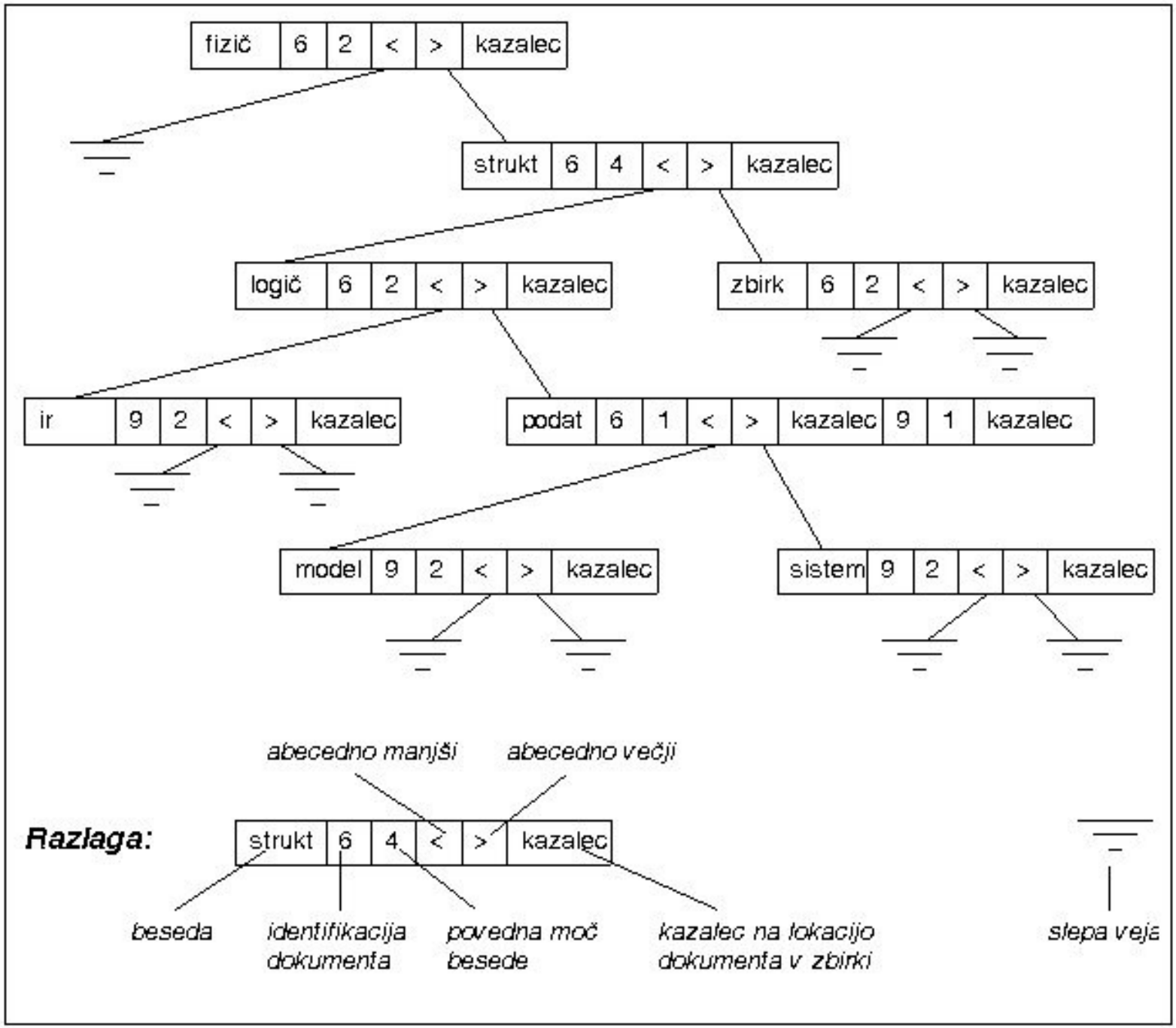



- Pomembna značilnost drevesne strukture je hierarhična povezanost in od tega odvisna višina drevesa.
- Različice drevesnih struktur izvirajo iz števila povezav, ki lahko povezujejo list s podrejenimi listi.

# Drevesa

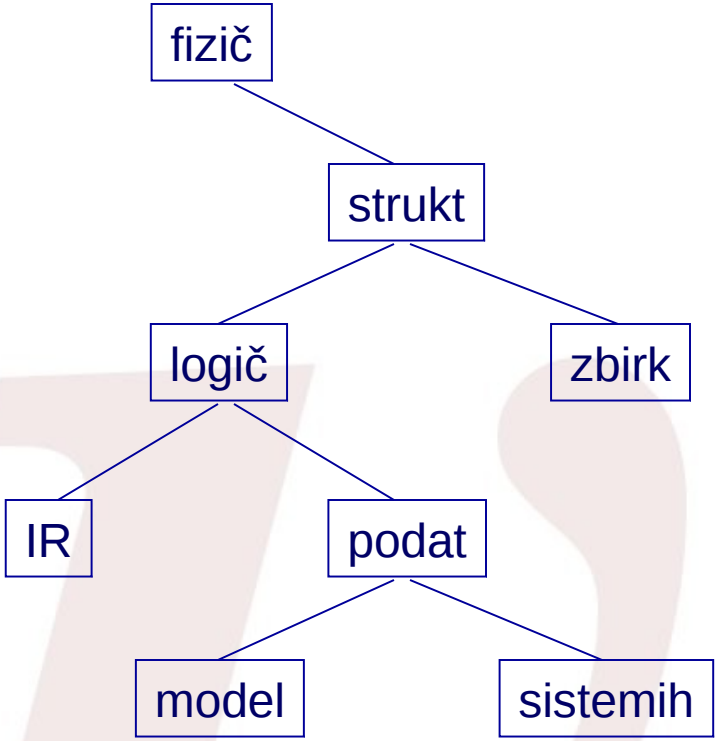



- Drevesna struktura že sama po sebi predvideva urejanje podatkov.
- Pogost kriterij urejanja podatkov je abeceda ali številske vrednosti izrazov (števil).


fizič 6 2 < > kazalec
strukt 6 4 < > kazalec
logič 6 2 < > kazalec
zbirk 6 2 < > kazalec
ir 9 2 < > kazalec
podat 6 1 < > kazalec 9 1 kazalec
model 9 2 < > kazalec
sistem 9 2 < > kazalec
abecedno manjši
abecedno večji
Razlaga:
strukt 6 4 < > kazalec
beseda
identifikacija dokumenta
povedna moč besede
kazalec na lokacijo dokumenta v zbirki
slepa veja

fizična struktura logična struktura zbirk podatkov

podatkovni modeli IR sistemih



# Drevesa

- Iskanje poteka podobno, kot gradnja drevesa.
- Ko iščemo besedo
  - začnemo pri korenu,
  - na vsakem nivoju se odločamo o nadaljnji smeri iskanja.
- Iskanje je končano
  - ko naletimo na iskano besedo, ali
  - naletimo na slepo vejo (iskane besede ni v drevesu).

# Drevesa

- Za drevesa velja, da so enako primerna za Boolove operacije ali iskanje z rangiranjem, kot invertirana arhitektura.

# Datoteke signatur

- S signaturami zapisujemo informacije o vsebini dokumenta na način, ki ne temelji na besedilnem zapisu.
- Temeljijo na principih kodiranja informacije iz predračunalniške dobe.
- Za spletne zbirke ta arhitektura ni značilna – tu jo predstavljamo zaradi zanimive ideje.

# Datoteke signatur

- Osnova postopka je pretvorba znakovnega zapisa besede v unikaten binarni zapis (sestavljen iz ničel in enic).
- Za to obstajajo enostavni in hitri računski postopki (razpršilne funkcije, hashing functions).
- Naloga razpršilne funkcije je pretvorba znakovnega zapisa v binarni na način, ki zagotavlja popolno unikatnost signature.

# Datoteke signatur

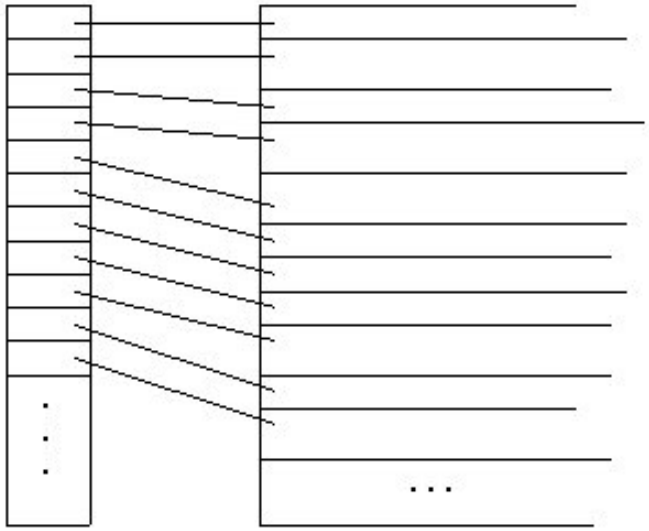
- Signature posameznih besed seštejemo v signaturo dokumenta z uporabo operatorja ALI.
- Signatura dokumenta vsebuje signature vseh besed v njem.
- Primer:
  - **iskanje** 0 1 0 0 0 1 0 1
  - **informacij** 1 0 0 0 1 0 0 1
  - **skupna signatura** 1 1 0 0 1 1 0 1

# Datoteke signatur

- Signaturo dokumenta dobimo s prekrivanjem - *superponiranjem* signatur posameznih besed, zato take signature imenujemo *superponirane signature*.
- Razen superponiranih obstajajo tudi druge različice signatur dokumentov.
- Signature vseh besed in signatura dokumenta so enako dolge.

| besede | signature |
|---|---|
| fizič | 000100000000000100000010 |
| strukt | 000001010000000000001000 |
| logič | 100000000010000010000000 |
| strukt | 000001010000000000001000 |
| zbirk | 000100000100000000100000 |
| podat | 100001000000100000000000 |
| **dok. 6** | 100101010110100110101010 |
| podat | 100001000000100000000000 |
| model | 000010000010000000000001 |
| ir | 000000100000100000000100 |
| sistem | 010000000100000000001000 |
| **dok. 9** | 110011100110100000001101 |

| | |
|---|---|
| *dok. 1* | 011000010011100011001011 |
| *dok. 2* | 101010001101011100100100 |
| *dok. 3* | 001010110001101001101010 |
| *dok. 4* | 100101100101011001011001 |
| *dok. 5* | 001000101100101100010001 |
| ***dok. 6*** | 100101010110100110101010 |
| *dok. 7* | 011000100101011000111001 |
| *dok. 8* | 001010100010110011010010 |
| ***dok. 9*** | 110011100110100000001101 |
| *dok. 10* | 001110001101001101001100 |
| *dok. 11* | 011001001011001010110000 |
| | ⋮ |




***datoteka signatur*** ***datoteka kazalcev*** ***datoteka besedil***

# Datoteke signatur

- Iskanje dokumentov v datoteki signatur:
- Beseda iz iskalne zahteve se pretvori v svojo signaturo.
- Za <u>vsako</u> signaturo dokumenta iz datoteke signatur se preveri, če vsebuje signaturo iskane besede (operacija IN).

# Datoteke signatur

- **iskanje 0 1 0 0 0 1 0 1**
- **informacij 1 0 0 0 1 0 0 1**
- **skupna signatura 1 1 0 0 1 1 0 1**

Iskanje z besedo

- **informacij 1 0 0 0 1 0 0 1**

- **skupna signatura 1 1 0 0 1 1 0 1**

IN **informacij 1 0 0 0 1 0 0 1**

**rezultat 1 0 0 0 1 0 0 1**

# Datoteke signatur

- Za signature dokumentov, pri katerih se je primerjanje pozitivno izteklo, se v datoteki kazalcev poiščejo identifikacije dokumentov.
- Za vsak poiskan dokument se preveri, če ne gre za lažni zadetek.

# Datoteke signatur

- Iskanje poteka zelo hitro, čeprav je treba zaporedoma pregledati vse signature.
- Razlogi za hitrost:
  - primerjanje signatur je izredno enostavno,
  - opis dokumenta je koncentriran na enem mestu,
  - fiksne dolžine signatur omogočajo njihovo učinkovito branje z zunanjega pomnilnika.

# Datoteke signatur

- Lažni zadetki so slaba lastnost zapisovanja vsebine s signaturami.
- Enice v signaturi iskane besede lahko ustrezajo enicam signature dokumenta, čeprav v njem iskane besede ni.
- Iskalni algoritem take napake, lažnega zadetka, ne more odkriti.

# Datoteke signatur

Primer lažnega zadetka:

- **iskanje 01000101**
- **informacij 10001001**

---

- **skupna signatura 11001101**

Iskanje z besedo

- **zajec 11000100**

se izteče pozitivno.

Besede v skupni signaturi dokumenta ni.

# Datoteke signatur

- Verjetnost lažnih zadetkov zmanjšamo s pametnim izborom lastnosti signatur.
- Lastnosti signature, ki najbolj vplivata na verjetnost lažnih zadetkov, sta dolžina signature in pogostost enic v njej.

# Datoteke signatur

Čim daljša je signatura,

- tem bolj razpršene so lahko enice in
- tem manjša je zato možnost lažnih zadetkov.

Čim več enic dovolimo v signaturi,

- tem več besed je lahko shranjenih v njej,
- tem večja je možnost lažnih zadetkov.

# Datoteke signatur

- Lastnosti si nasprotujeta in treba je najti pametno ravnotežje.
- Velja, da je tako ravnotežje pri 50% enic in 50% ničel.
- Daljša besedila je zato treba razdeliti na kose s takim številom različnih besed, da enice njihovih signatur približno do polovice "napolnijo" skupno signaturo.

# Datoteke signatur

- Pri 1024-mestnih signaturah in polovici enic v njih, pade verjetnost lažnih zadetkov do 10-5.