Univerza v Ljubljani

Fakulteta za računalništvo in informatiko

Igor Rožanc

Osnove algoritmov in podatkovnih struktur I

(OAPS I)

2. letnik, VSP Računalništvo in informatika, vse smeri

PROSOJNICE ZA 15. PREDAVANJA (11.1.2007)

Študijsko leto 2006/07

Vrste dreves - URAVNOTEŽENOST:

- Za **uravnoteženo drevo** velja, da se pri vsakem vozlišču višina vseh njegovih poddreves čimbolj enaka.
  - posredno zagotavlja majhno višino drevesa (približna polnost)
  - uravnoteženost zagotovimo z ustreznimi operacijami za vstavljanje in izločanje elementov
- **Popolnoma izravnano drevo** je dvojiško uravnoteženo drevo - pri vsakem vozlišču se število vozlišč njegovega levega in desnega poddrevesa razlikuje kvečjemu za 1
  - **Rekurzivna gradnja:** prvi element postane koren
    - iz prve polovice elementov (od drugega do srednjega elementa) rekurzivno zgradimo levo poddrevo
    - preostanek (od srednjega elementa do konca) rekurzivno zgradimo desno poddrevo

Zgled: dvojiško popolnoma izravnano drevo z 1, 2, 3, 4, 5, 6, 7, 8,...,15 vozlišči

- **AVL drevo** je primer uravnoteženega dvojiškega iskalnega drevesa

Vrste dreves - UREJENOST:

- Dodajanje in izločanje pri iskalnih drevesih upošteva urejenost in zagotavlja tako sestavo dreves, ki omogočajo učinkovito iskanje elementov
  - vloga vozlišč – usmerjanje pri iskanju
- Tipičen primer: **Dvojiško iskalno drevo (Binary Search Tree - BST)**
  - Za vsako vozlišče **iskalnega drevesa** velja:
    - (če je x vrednost trenutnega vozlišča)
    - vrednost vsakega vozlišča v levem poddrevesu je manjša od x
    - vrednost vsakega vozlišča v desnem poddrevesu je večja od x
    - (vsaka vrednost se v drevesu nahaja samo v enem vozlišču)
  - Najboljši, najslabši, povprečen primer

Zgled: postopna izgradnja BST za vozlišča: **{ 5, 2, 7, 1, 3, 4, 10, 8, 6, 5 }**
postopno brisanje vozlišč z vrednostmi: {**9, 5, 8, 3, 5, 4** }

- Drugi primeri: **Iskalna drevesa višjih stopenj (B drevesa)**
  AVL drevesa

Obhodi dreves:

- postopek, ki sistematično obišče vsa vozlišča drevesa (oziroma izvede določeno operacijo na vseh vozliščih – tipično izpis)
- **Poznamo naslednje obhode:**
  - Obhod po nivojih (ang. level order)
  - Premi vrsti red (ang. preorder)
  - Vmesni vrstni red (ang. inorder)
  - Obratni vrstni red (ang. postorder)

Zgled: vsi obhodi dvojiškega drevesa za aritmetični izraz

Postopki za vse vrste obhodov ...

Drevesa (Trees) 93

Tipična ponazoritev dreves v računalniku:

- s pomočjo tabele, kjer vsak element tabele predstavlja eno vozlišče (kot pri sortiranju s kopico)
- z (ustrezno) povezanimi objekti, ki predstavljajo posamezna vozlišča

Prikaz: ponazoritev drevesa s tabelo:

- primer: neizpolnjeno dvojiško drevo
- vrednosti v vozliščih so znaki
- preslikovalna funkcija: levi sin i = 2*i+1, desni sin i = 2*i+2

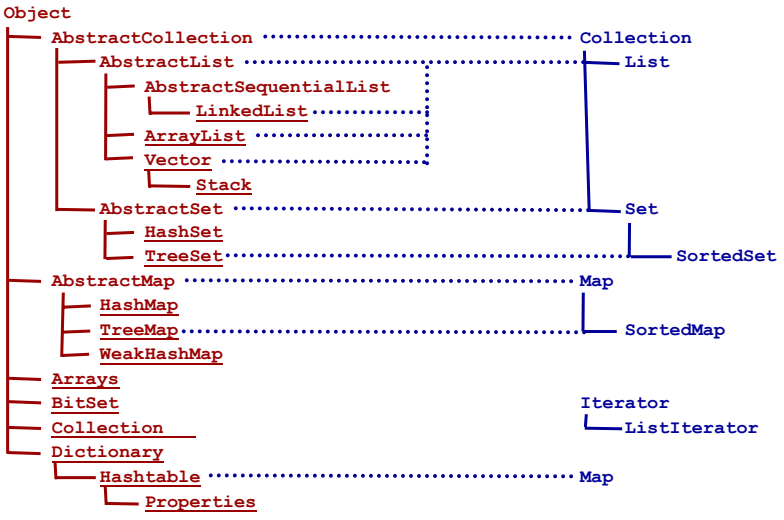
Prikaz: ponazoritev s povezanimi objekti

- primer: isto drevo
- vsako vozlišče ima povezavo na (prednika,) levega in desnega sina

Izvedba: razreda `BinaryTree()` in `TestBinaryTree()`

Primer: `BinaryTree.java, TestBinaryTree.java`

Collections Framework 94



Iskalna tabela (Map) 95

Predstavlja zaporedje parov <ključ,vrednost>, ki omogoča učinkovito iskanje vrednosti po ključih

- angleški izrazi: map, lookup table, associative array, dictionary

Ključi: enolični, lahko so poljubnega tipa - ne nujno celoštevilski,

Vrednosti: vsebujejo podatke

Primer: slovar, legenda, kazalo, seznam lokacij ...

`Collection Framework` vsebuje:

- generični vmesnika **`Map`** in **`SortedMap`**
- generični abstraktni razred **`AbstractMap`**
- več realiziranih generičnih razredov: **`HashMap`**, **`TreeMap`**, **`WeakHashMap`**

Iskalna tabela (Map) 96

```
public interface Map
{  int size();
   boolean isEmpty();
   boolean containsKey(Object k);
   boolean containsValue(Object v);
   Object get(Object k);
   Object put (Object k, Object v);
   Object remove(Object k);
   putAll(Map);
   clear();
   Set keySet();
   Set entrySet();
```

```
    ...
    Collection values();
    public interface Entry
    {  Object getKey();
       Object getvalue();
       Object setValue(Object v);
       boolean equals(Object o);
       int hashCode();
    }
    boolean equals(Object o);
    int hashCode();
}
```


Razred `HashMap`

- realizacija z (zaprto) razpršeno tabelo (hash table)
- v ozadju tabela določene kapacitete
- razporejanje s pomočjo razpršilne funkcije:
  - zagotavlja hitro vstavljanje in iskanje
  - problem kolizij:
    - linerno ponovno razprševanje
    - kvadratično ponovno razprševanje
    - odprte razpršene tabele
  - zagotavljanje primerne kapacitete in zasedenosti

Prikaz razprševanja: slovensko – angleški slovar

Primer: `Slovar.java`

Razred `TreeMap`

- realizacija z dvojiškim iskalnim drevesom
- v ozadju ustrezno povezano drevo objektov
- upošteva urejenost po ključih – izpis

Primer uporabe: kazalo

Razred `WeakHashMap`

- realizacija s šibkimi razpršenimi tabelami
- ko ključ (ali vrednost) ni več dosegljiv, se element odstani iz tabele

Primer uporabe: seznam datotek na disku ...