Programiranje I – RIN
Računalništvo I – MA

REKURZIVNE PODATKOVNE STRUKTURE
(SEZNAM & DREVO)

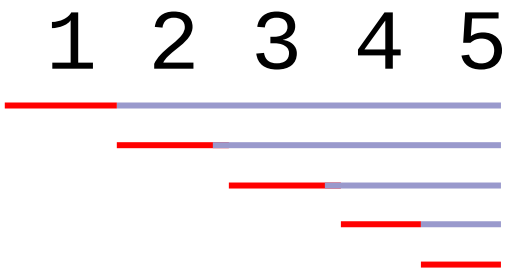
8. predavanje

Rekurzivne podatkovne strukture

- *Kakšna je rekurzivna funkcija?*
- Takšna, ki za izračun uporablja samo sebe.
- *Iz katerih delov sestoji rekurzivna funkcija?*
- Iz **ustavitvenega pogoja** ter iz dela **deli in vladaj**.
- Podobno pri rekurzivnih podatkovnih strukturah
  - struktura uporablja sebe za hranjenje dela podatkov.

Seznam kot RPS

- Seznam je lahko: **prazen** ali **neprazen**
- Če je seznam neprazen sestoji iz: **glave** in **podseznama** (slednji je lahko prazen)
- Primer:



Seznam celih števil

- Operacije nad seznamom:
  - tvorjenje in uničevanje
  - dodajanje na začetku ali na koncu
  - odvzemanje na začetku ali na koncu
  - poizvedovanje po prvem elementu (glavi) in preostanku seznama (rep)
  - poizvedovanje o dolžini in elementu v seznamu

Seznam v JAVI

- Poseben seznam je ***prazen seznam*** **`null`**
- Razred nepraznih seznamov celih števil:

```
public class Seznam {
  private int  glava = 0;
  private Seznam rep = null;

  …
} // Seznam
```

Tvorjenje

```
public Seznam(int elt, Seznam lst) {
  this(elt, lst);
} // Seznam
public Seznam(int elt) {
  this(elt, null);
} // Seznam
```

- **tvorjenje in uničevanje**
- dodajanje na začetku ali na koncu
- odvzemanje na začetku ali na koncu
- poizvedovanje po prvem elementu (glavi) in preostanku seznama (rep)
- poizvedovanje o dolžini in elementu v seznamu

Dodajanje na začetku ali na koncu

- **dodajanje na začetku ali na koncu**
- odvzemanje na začetku ali na koncu
- poizvedovanje po prvem elementu (glavi) in preostanku seznama (rep)
- poizvedovanje o dolžini in elementu v seznamu

Razred PovezanSeznam

```
public boolean prepend(int elt){
  Seznam tmp = new Seznam(elt);
  tmp.rep = sz;
  sz = tmp;
}
```

Poizvedovanja: glava, rep, dolžina, ...

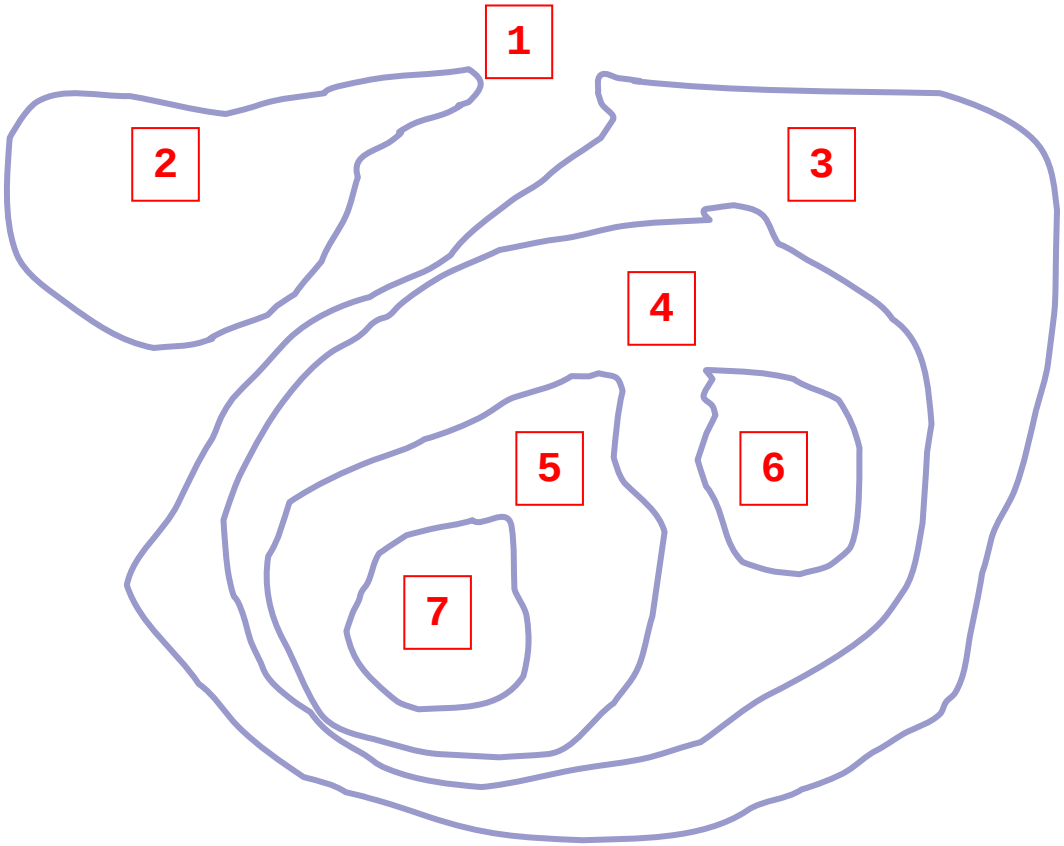
```
public int  glava(void) { return glava; }
public Seznam rep(void) { return rep; }

public int dolzina(void) {
  if (rep == null) return 1;
  else return (1 + rep.dolzina());
} // dolzina
```

- odvzemanje na začetku ali na koncu
- **poizvedovanje po prvem elementu (glavi) in preostanku seznama (rep)**
- **poizvedovanje o dolžini** in elementu v seznamu

Dvojno rekurzivni seznam

- Dvojno rekurzivni seznam je, kot običajni seznam lahko: **prazen** ali **neprazen**
- Če je seznam neprazen sestoji iz: **glave** in ***dveh* podseznamov** (vsak od slednjih je lahko prazen)

Dvojno rekurzivni seznam



Dvojno rekurzivni seznam

- Dvojno rekurzivni seznam imenujemo **dvojno** (**binarno**) **drevo**
- *Glavo* pri drevesu imenujemo **koren** in *podseznama* **levo poddrevo** ter **desno poddrevo**
- Obstajajo seveda tudi k-terno rekurzivni seznami, ki jih podobno imenujemo **k-tiška drevesa**
  - ta drevesa imajo (do) *k* poddreves
  - in (do) *k-1* elementov v korenu

Primer dvojnega drevesa

- Posebno drevo je ***prazno drevo*** **`null`**
- Razred nepraznih dreves celih števil:

```
public class Drevo {
  private int glava = 0;
  private tree levo = null;
  private tree desno = null;
  …
} // Drevo
```

Tvorjenje

```
public Drevo(int elt, Drevo novoLevo, Drevo novoDesno){
  this(elt, novoLevo, novoDesno);
} // Drevo

public Drevo(int elt) {
  this(elt, null, null);
} // Drevo
```

Dodajanje

- Dodajanje na začetku (vrhu / korenu)

```
public Drevo prepend(int elt) {
  Drevo novoDrevo = new Drevo(elt, this, null);
  return novoDrevo;
} // prepend
```

- ali na koncu (pri listih)

```
public Drevo append(int elt) {
  if (right == null) right= new Drevo(elt);
  else               right= right.append(elt);
  return this;
} // append
```

Poizvedovanja

- Poizvedovanja: koren, poddrevo

```
public int  value(void)        { return root;  }
public Drevo levoPodDrevo(void)  { return left;  }
public Drevo desnoPodDrevo(void) { return right; }
```

- Poizvedovanja: višina

```
public int visina(void) {
  int levaVisina;
  int desnaVisina;
  if (levo == null) levaVisina = 0;
  else              levaVisina = levo.visina();
  if (desno == null) desnaVisina = 0;
  else               desnaVisina = desno.visina ();
  return (1 + max(levaVisina, desnaVisina));
} // visina
```

Poizvedovanja

- Poizvedovanja: vsebina

```
public boolean najdi(int elt) {
  boolean nasel;
  if (glava == elt) return true;
  if (levo == null) nasel = false;
  else nasel = levo.search(elt);
  if(!nasel)
    if (desno != null) nasel = desno.najdi(elt);
  return nasel;
} // najdi
```

Obhodi in izpisi

- Prvi (*preorder*)

```
public void preorder(void) {
  System.out.println(glava);  // PRE - pred
  if (levo != null) levo.preorder();
  if (desno != null) desno.preorder();
} // preorder
```

- Vmesni (*inorder*)

```
public void inorder(void) {
  if (levo != null) levo.inorder();
 System.out.println(glava); // IN - vmes
  if (desno != null) desno.inorder();
} // inorder
```

- Zadnji (*postorder*) - (mala) domača naloga

Binarno iskalno drevo

```
public class Drevo {
      int glava;
      Drevo levi, desni;
//metode
}
```

Binarno iskalno drevo - vstavi

```
boolean vstavi(int elt){
    if(glava > elt){
      if(levi == null){
        levi = new Drevo();
        levi.glava = elt;
        return(true);
      }
      else{return(levi.vstavi(elt));}
    }
    else{
      if(desni == null){
        desni = new Drevo();
        desni.glava = elt;
        return(true);
      }
      else{return(desni.vstavi(elt));}
    }
  }
```

Binarno iskalno drevo - isci

```
boolean isci(int elt){
    if(glava == elt){return(true);}
    else{
      if(glava > elt){
        if(levi == null){
          return(false);
        }
        else{return(levi.isci(elt));}
      }
      else{
        if(desni == null){
          return(false);
        }
        else{
          return(desni.isci(elt));
        }
      }
    }
}
```

Binarno iskalno drevo - brisi

```
boolean brisi(int elt){
    if(glava > elt){
      if(levi == null){return(false);}
      else{
        if(levi.glava == elt){
          //prevec za prvi letnik
          return(true);
        }
        else{return(levi.brisi(elt));}
    }
    else{
      if(desni == null){
        return(false);
      }
      else{
        if(desni.glava == elt){
          //prevec za prvi letnik
          return(true);
        }
        else
          return(desni.brisi(elt));
      }
    }
}
```

Zaključek

- Ogledali smo si:
  - Rekurzivne podatkovne strukture: *seznami*, *drevesa*
  - Preproste za rokovanje: vse, o čemer mora struktura kaj vedeti, je:
    - ***o sebi*** in
    - o svoji ***neposredni okolici*** (naslednik, poddrevo, …).

Povzetek

- Rekurzivne podatkovne strukture
- Seznam
  - tvorjenje
  - dodajanje (na začetek/konec)
  - poizvedovanje
  - odvzemanje (z začetka/konca)
  - Seznam kot vrsta
  - Prazni seznami
- Dvojno rekurzivni seznam (drevo)
  - Obhodi in izpisi v drevesih