Algoritmi in podatkovne strukture 1

Visokošolski strokovni študij Računalništvo in informatika

Ukoreninjena drevesa

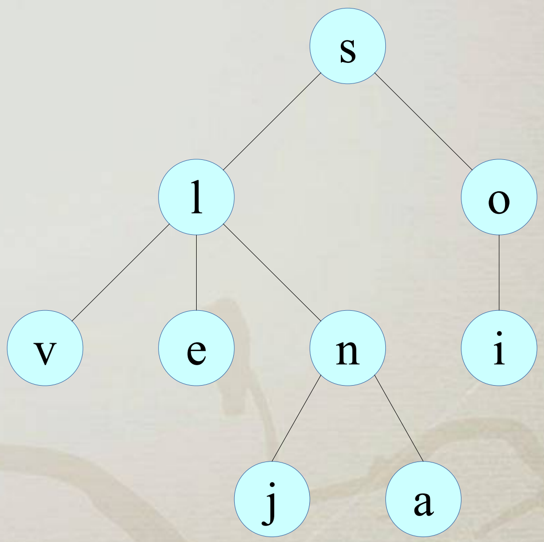
Jurij Mihelič, UniLj, FRI

Drevesa

Drevesa

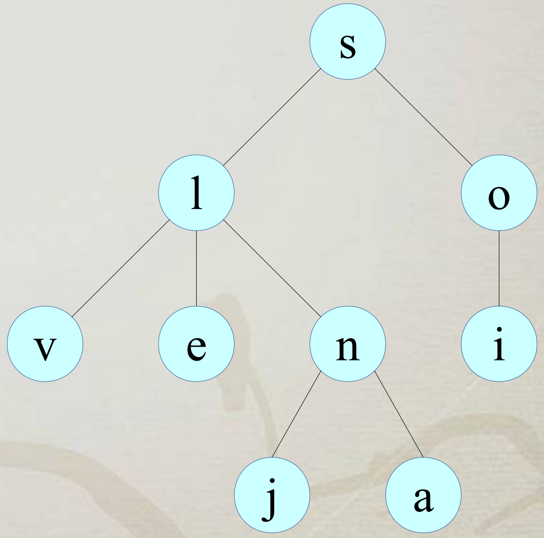
Drevesa

- Ukoreninjeno drevo (*rooted tree*)
  - drevo s **korenom**
  - sestoji iz **vozlišč**
    - vozlišča lahko vsebujejo nek podatek (element, oznaka)
    - koren
    - notranje vozlišče
    - končno (zunanje) vozlišče oz. list
  - in **povezav**
    - povezuje dve vozlišči



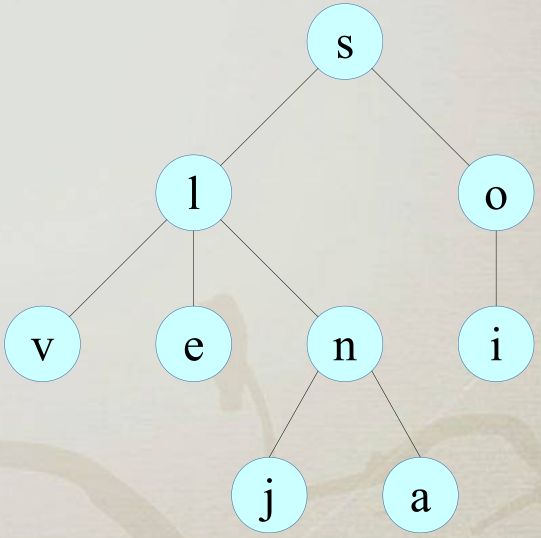
Drevesa

- Odnosi med vozlišči
  - starš
  - otrok
  - prednik
  - potomec



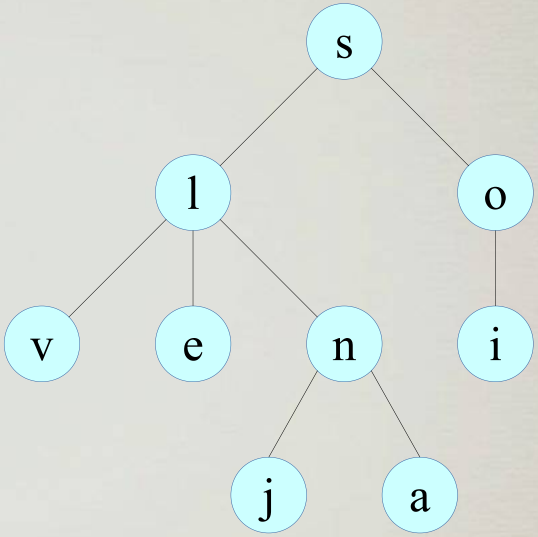
Drevesa

- Pot
  - povezave od **izvora** do **cilja**
    - navadno je izvor kar koren drevesa
  - dolžina poti = št. povezav na poti



Drevesa

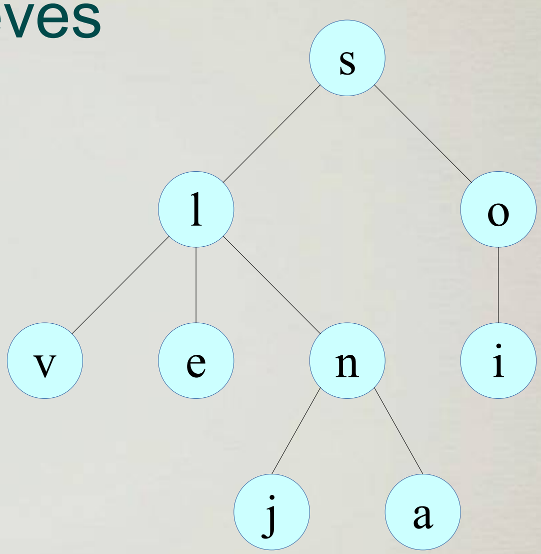
- Poddrevo
  - vozlišče in **vsi** njegovi potomci
  - vozlišče je koren poddrevesa
  - poddrevo je drevo



- Gozd
  - množica dreves

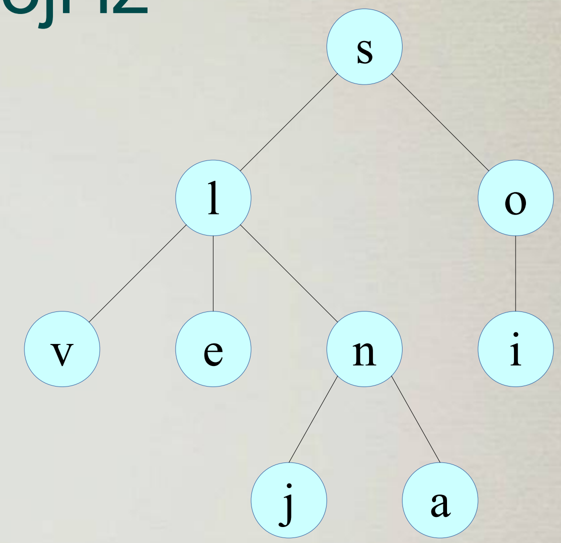
Drevesa

- Urejeno in neurejeno drevo
  - glede na vrsti red otrok
    - razlikovanje strukturno enakih dreves
  - glede na urejenost starš / otrok
    - pogosto se pojavi v praksi



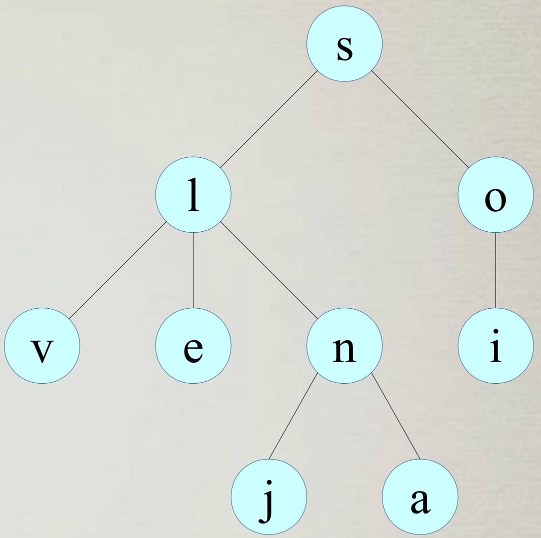
Drevesa

- Definicija
  - Ukoreninjeno drevo $T=(V,E,r)$ sestoji iz
    - končne množice vozlišč V in
    - končne množice povezav E
  - pri čemer
    - je eno vozlišče koren r *(root)*
    - ima vsako vozlišče 0 ali več otrok
    - potomci korena razpadejo na *disjunktno* unijo poddreves



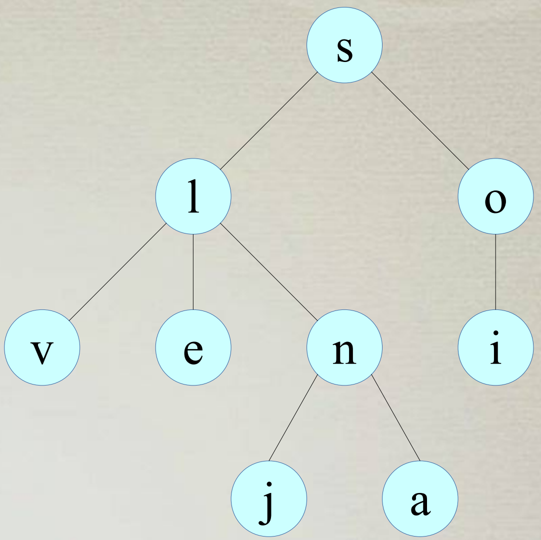
Drevesa

- Grafična ponazoritev
  - povezan acikličen graf
  - po nivojih

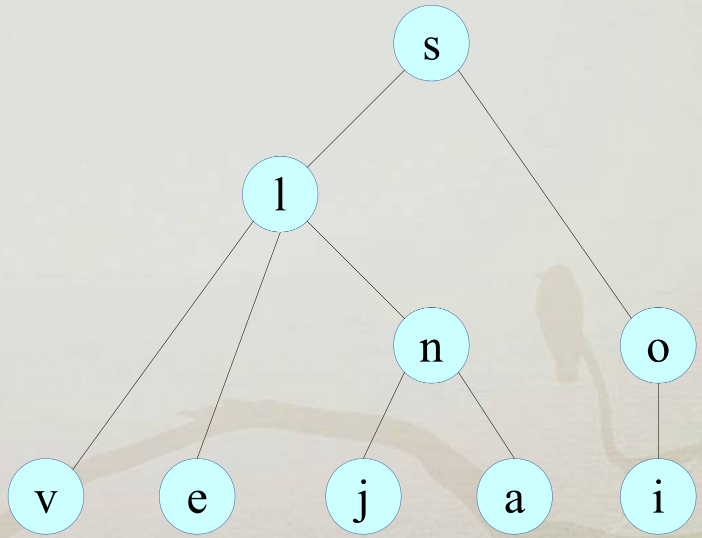


Lastnosti

- Globina vozlišča
  - dolžina poti od korena do vozlišča
  - nivo: vozlišča na isti globini

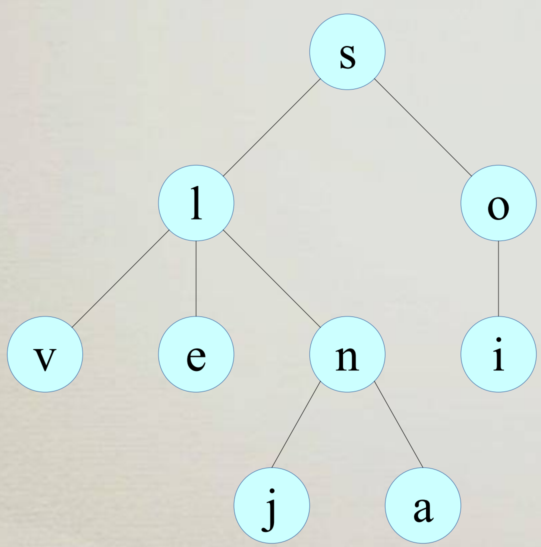


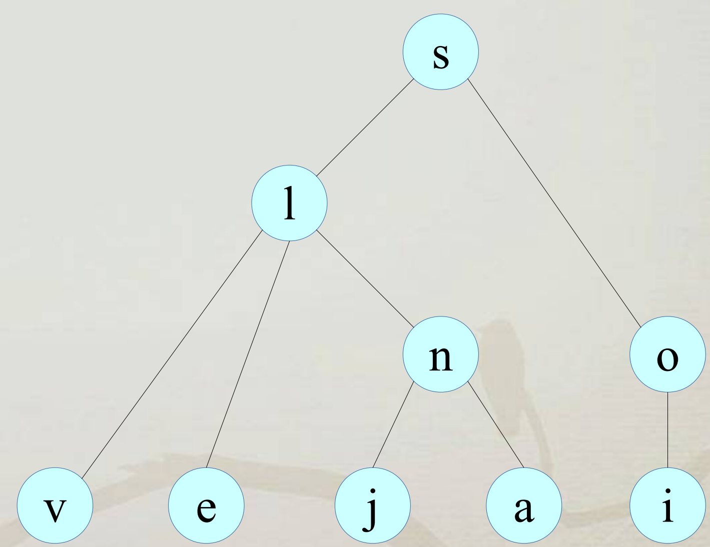
- Višina vozlišča
  - dolžina najdaljše poti od vozlišča do lista



Lastnosti

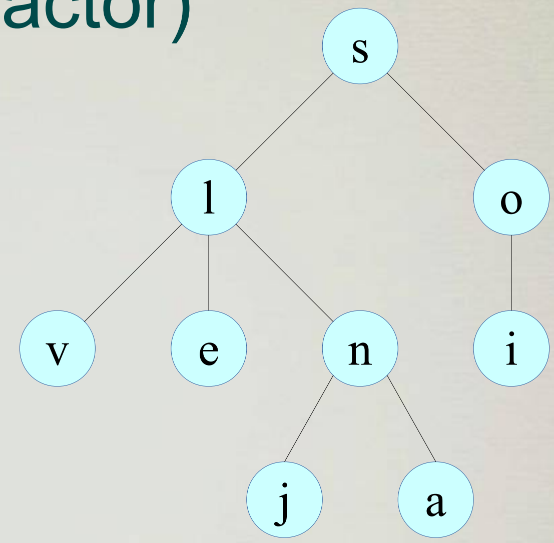
- Višina (oz. globina) drevesa
  - je enaka višini korena
  - oz. dolžina najdaljše poti od korena do lista





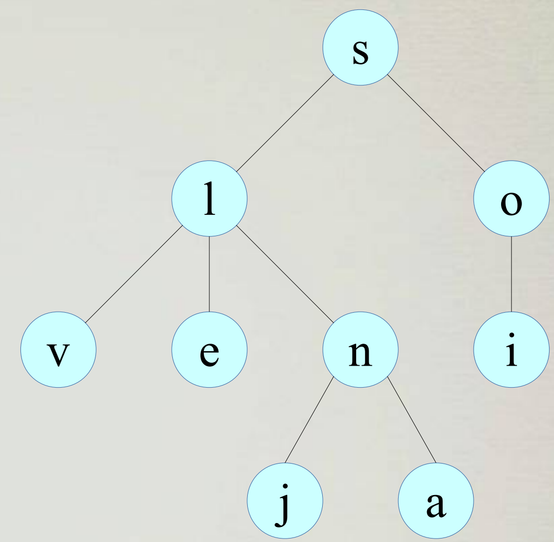
Lastnosti

- Stopnja vozlišča
  - tudi vejitveni faktor (branching factor)
  - **št. otrok** (oz. poddreves)
  - oznake: $\deg(v) = \quad (v)$
  - Kakšna je stopnja?
    - list
    - notranje vozlišče



Lastnosti

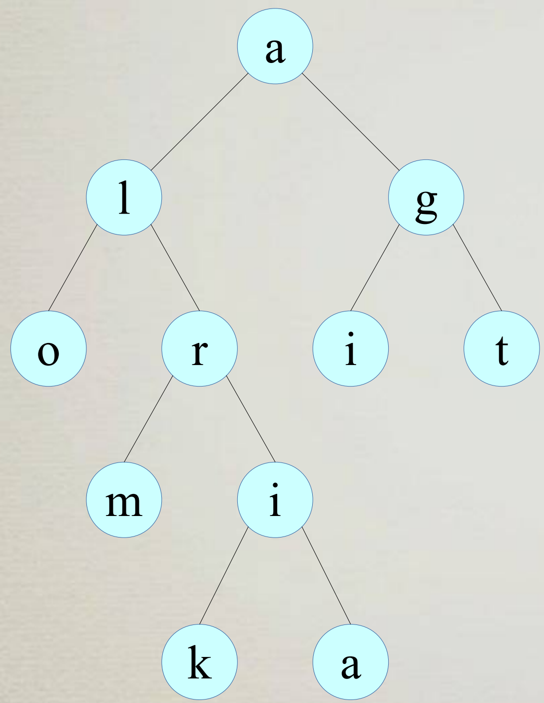
- Stopnja drevesa
  - največja stopnja vozlišča
  - dvojiško (*binary*) drevo
  - trojiško (*ternary*) drevo
  - d-tiško (d-*ary*) drevo
    - v: $\deg(v)$ $\deg(T)$

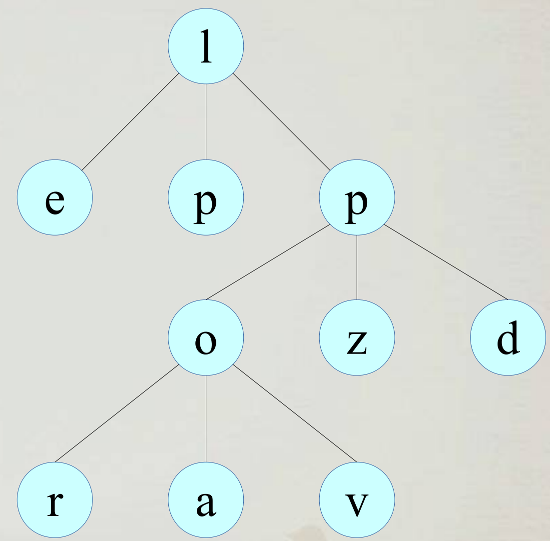


- Polno vozlišče
  - stopnja vozlišča je enaka stopnji drevesa
    - $\deg(v) = \deg(T)$

Lastnosti

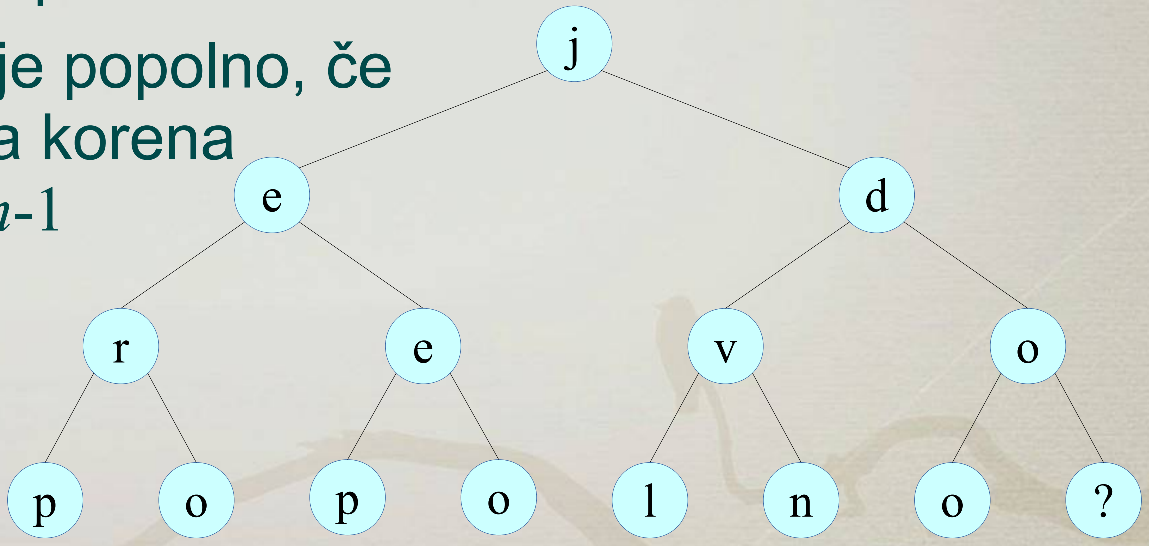
- Polno drevo (*full tree*)
  - vsa notranja vozlišča so polna





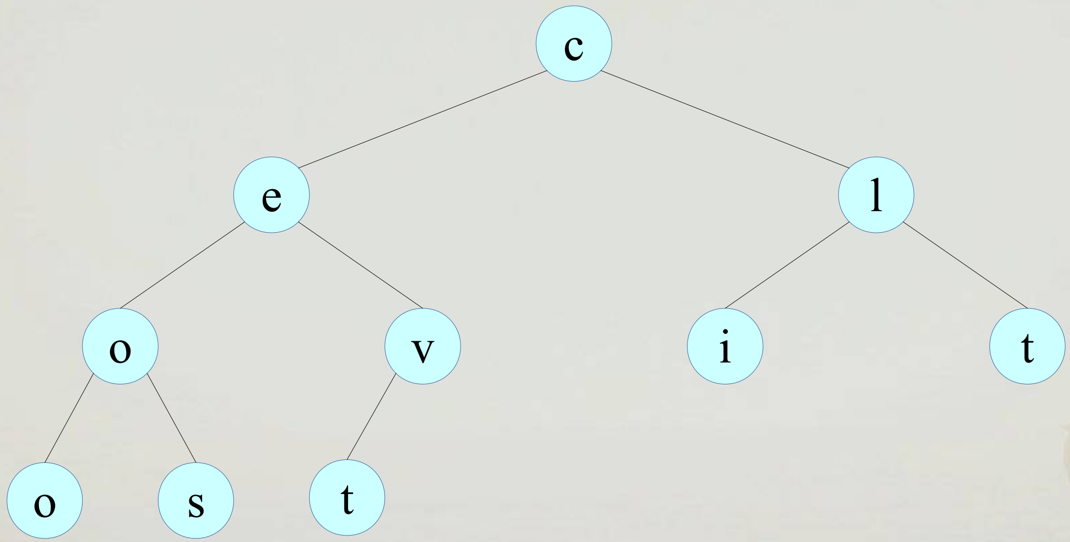
Lastnosti

- Popolno drevo (*perfect tree*)
  - polno drevo
  - vsi listi so na istem nivoju
  - rekurzivna definicija
    - drevo višine 0 je popolno
    - drevo višine $h > 0$ je popolno, če so vsa poddrevesa korena popolna in višine h-1

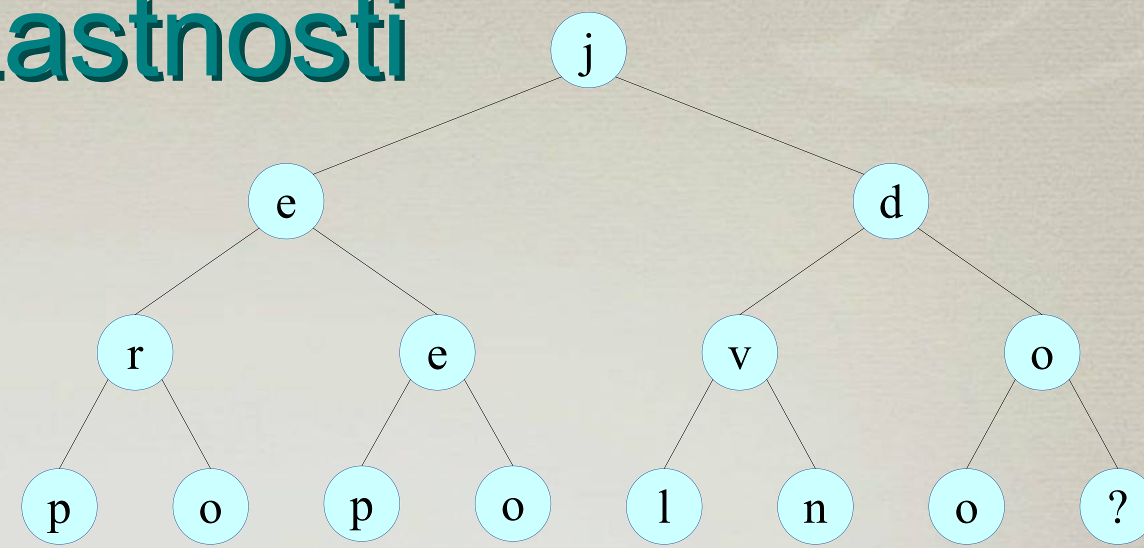


Lastnosti

- Celovito drevo (*complete tree*)
  - vsi nivoji (razen morda zadnji) so polni
  - vsi listi na zadnjem nivoju so levo



Lastnosti



popolno drevo	popolno dvojiško	popolno trojiško	popolno d-tiško
št. vozlišč na nivoju i	2^i	3^i	d^i
št. listov	2^h	3^h	d^h
št. notranjih vozlišč	$2^h - 1$	$\frac{3^h - 1}{2}$	$\sum_{i=0}^{h-1} d^i = \frac{d^h - 1}{d-1}$
št. vozlišč, n	$2^{h+1} - 1$	$\frac{3^{h+1} - 1}{2}$	$\sum_{i=0}^{h} d^i = \frac{d^{h+1} - 1}{d-1}$
višina, h	$\lg(n+1) - 1$ $\lfloor \lg n \rfloor$	$\log_3(2n+1) - 1$ $\lfloor \log_3(2n) \rfloor$	$\log_d((d-1)n+1) - 1$ $\lfloor \log_d((d-1)n) \rfloor$

Lastnosti

celovito drevo	dvojiško	trojiško	d-tiško
št. vozlišč	$2^h \leq n \leq 2^{h+1} - 1$	$\frac{3^h + 1}{2} \leq n \leq \frac{3^{h+1} - 1}{2}$	$\frac{d^h - 1}{d - 1} + 1 \leq n \leq \frac{d^{h+1} - 1}{d - 1}$
višina	$\lfloor \lg n \rfloor$	$\lfloor \log_3(2n) \rfloor$	$\lfloor \log_d((d-1)n) \rfloor$

poljubno drevo	dvojiško	trojiško	d-tiško
št. vozlišč	$h + 1 \leq n \leq 2^{h+1} - 1$	$h + 1 \leq n \leq \frac{3^{h+1} - 1}{2}$	$h + 1 \leq n \leq \frac{d^{h+1} - 1}{d - 1}$
višina	$\lfloor \lg n \rfloor \leq h \leq n - 1$	$\lfloor \log_3(2n) \rfloor \leq h \leq n - 1$	$\lfloor \log_d((d-1)n) \rfloor \leq h \leq n - 1$

Drevesni algoritmi

- Predstavitev dreves
  - vozlišče z atributi
    - left, right … levi in desni otrok (dvojiška drevesa)
    - children … seznam otrok
    - parent … starš
    - item … element, ki se hrani v vozlišču

```
class Node is
    item: Object
    left: Node
    right: Node
end
```

```
class Node is
    item: Object
    children: [Node]
end
```

Drevesni algoritmi

- Od zgoraj navzdol (*top down*)
  - začnemo v poljubnem vozlišču
    - navadno v korenu
  - potujemo proti listom drevesa
    - uporabljamo atribute left, right ali children
  - rekurzija

```
fun topDown(v) is
   if v == null then
      ... ni vozlišče
   elif isLeaf(v) then
      ... list drevesa
   else
      ... notranje vozlišče
      topDown(v.left)
      topDown(v.right)
   endif
endfun
```

Drevesni algoritmi

- Od spodaj navzgor (*bottom up*)
  - začnemo v poljubnem vozlišču
  - potujemo proti korenu
    - uporabljamo atribut parent
  - rekurzija

```
fun bottomUp(v) is
   if v == null then
      ... ni vozlišče
   elif isRoot(v) then
      ... koren drevesa
   else
      ... notranje vozlišče
      bottomUp(v.parent)
   endif
endfun
```

Drevesni algoritmi

- Primeri
  - štetje vozlišč
  - štetje listov
  - štetje notranjih vozlišč
  - višina vozlišča
  - globina vozlišča
  - stopnja drevesa
  - vsota stopenj vseh vozlišč

Drevesni obhodi

- Sistematičen obisk vseh vozlišč drevesa
- Vrste obhodov (*traversal*)
  - premi obhod (*preorder*)
  - obratni obhod (*postorder*)
  - vmesni obhod (*inorder*)
    - le za dvojiška drevesa
  - obhod po nivojih (*level order*)

Drevesni obhodi

- Premi (direktni, neposredni) obhod drevesa
  - koren nato otroci
  - ideja algoritma
    - najprej obdelaj koren drevesa
    - nato obhodi še poddrevesa otrok

```
fun preorder(v) is
    if v == null then return
    println(v)
    preorder(v.left)
    preorder(v.right)
end
```

Drevesni obhodi

- Obratni obhod
  - otroci nato koren
  - ideja algoritma
    - najprej obhodi poddrevesa otrok
    - nato obdelaj koren drevesa

```
fun postorder(v) is
    if v == null then return
    postorder(v.left)
    postorder(v.right)
    println(v)
end
```

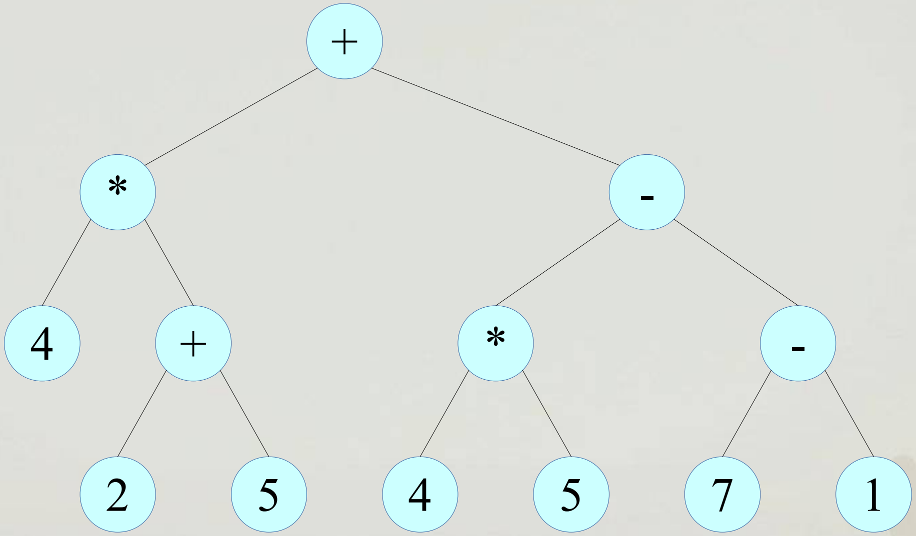
Drevesni obhodi

- Vmesni obhod
  - le na dvojiških drevesih
  - levo poddrevo, koren, desno poddrevo
  - ideja algoritma
    - obhodi poddrevo levega otroka
    - obdelaj koren
    - obhodi poddrevo desnega otroka

```
fun inorder(v) is
    if v == null then return
    inorder(v.left)
    println(v)
    inorder(v.right)
end
```

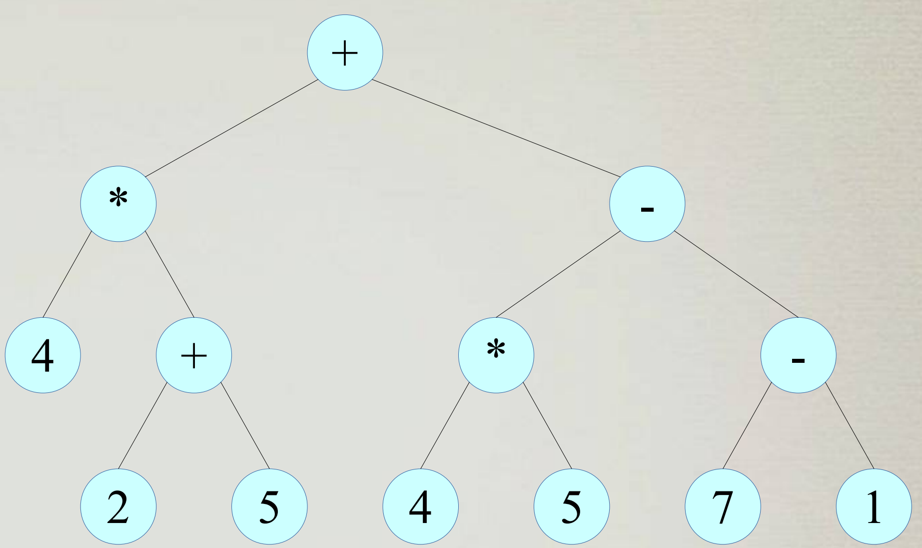
Drevesni obhodi

- Primeri
  - drevo za aritmetični izraz
    - 4 * (2+5) + (4*5 - (7-1))



Drevesni obhodi

- Primeri
  - premi obhod
    - +*4+25-*45-71
  - obratni obhod
    - 425+*45*71--+
  - vmesni obhod
    - ((4*(2+5))+((4*5)-(7-1)))



Drevesni obhodi

- Obhod po nivojih
  - zaporedoma obdelujemo nivoje
  - ideja algoritma
    - otroke shranjujemo v zbirko
    - katero zbirko uporabiti?

Drevesni obhodi

- Obhod po nivojih

```
fun levelOrder() is
   queue = Queue()
   queue.enqueue(root)
   while !queue.isEmpty() do
      x = queue.dequeue()
      println(x)
      for c in x.children do
         queue.enqueue(c)
   endwhile
endfun
```

Predstavitev dreves

- S kazalci
  - otroci
    - zaporedje kazalcev na otroke
    - binarno drevo: kazalca na levega in desnega otroka
  - prvi otrok in sorojenci (*sibling*)
    - kazalec na prvega otroka
    - vsak otrok ima kazalec na naslednjega sorojenca
  - starš
    - kazalec na starša
    - za *neurejena* drevesa
    - za *urejena* drevesa s navadno kombinira s prejšnjima dvema metodama

Predstavitev dreves

- V polju (implicitna predstavitev)
  - dvojiška drevesa
    - vozlišče z indeksom *i*
    - otroka: $l = 2i+1$, $r = 2i+2$
    - starš: *p* = ¥(*i-1*) / 2∂
  - *d*-tiška drevesa
    - otroci: indeks *j*-tega otroka = $d{\cdot}i + 1 + j$
    - starš: *p* = ¥(*i-1*) / d∂
    - otroci so torej na indeksih od $d{\cdot}i + 1$ do $\mathrm{d}{\cdot}i + d$
  - učinkovitost predstavitve
    - kapaciteta polja in velikost drevesa
    - izrojena in celovita drevesa

Predstavitev dreves

- Celovita (dvojiška) drevesa v polju
  - `items` ... polje elementov
  - `last` ... indeks zadnjega
  - notranja vozlišča: prvih $\lfloor n/2 \rfloor$ elementov
  - listi: zadnjih $\lceil n/2 \rceil$ elementov