Koki

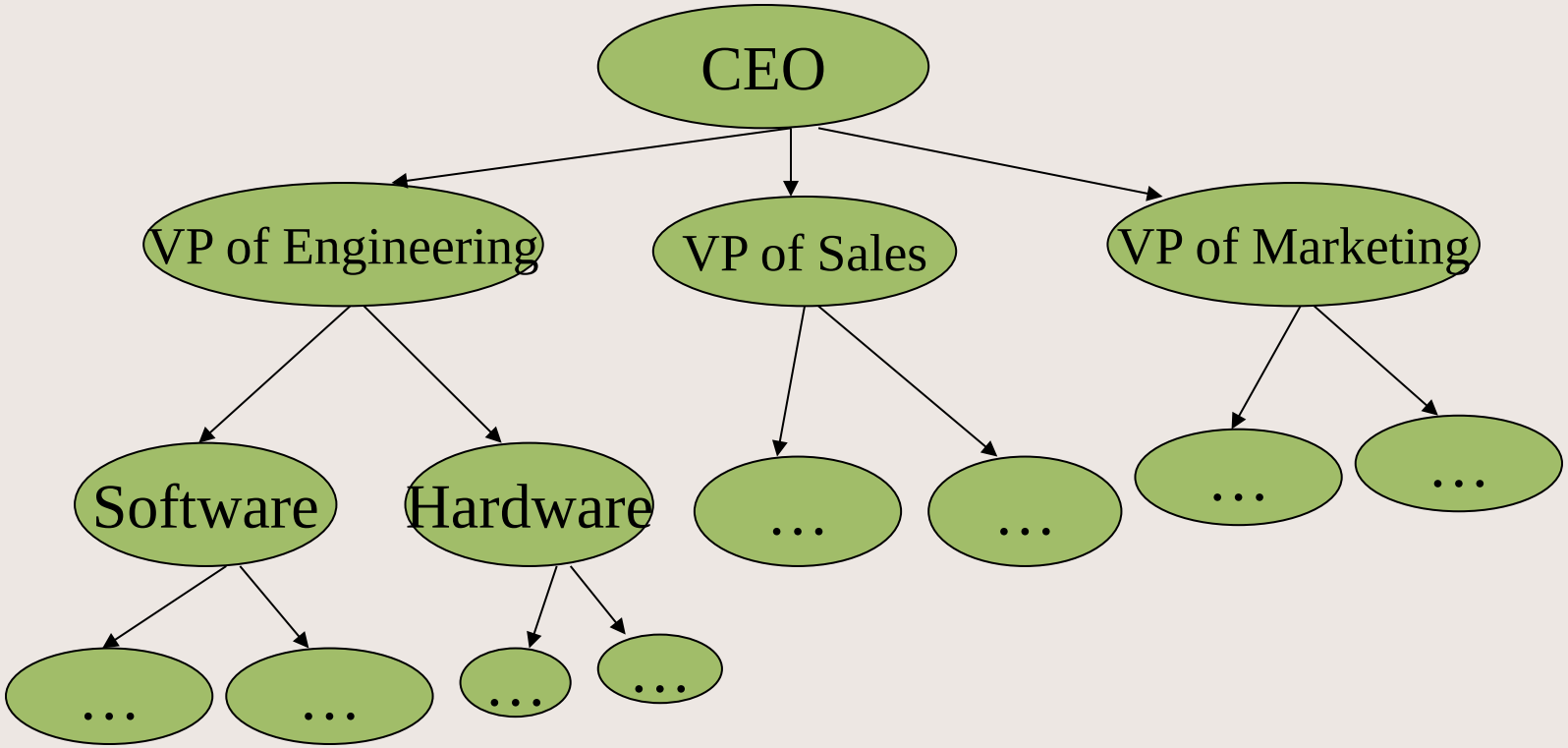
Guntis Arnicāns

2015

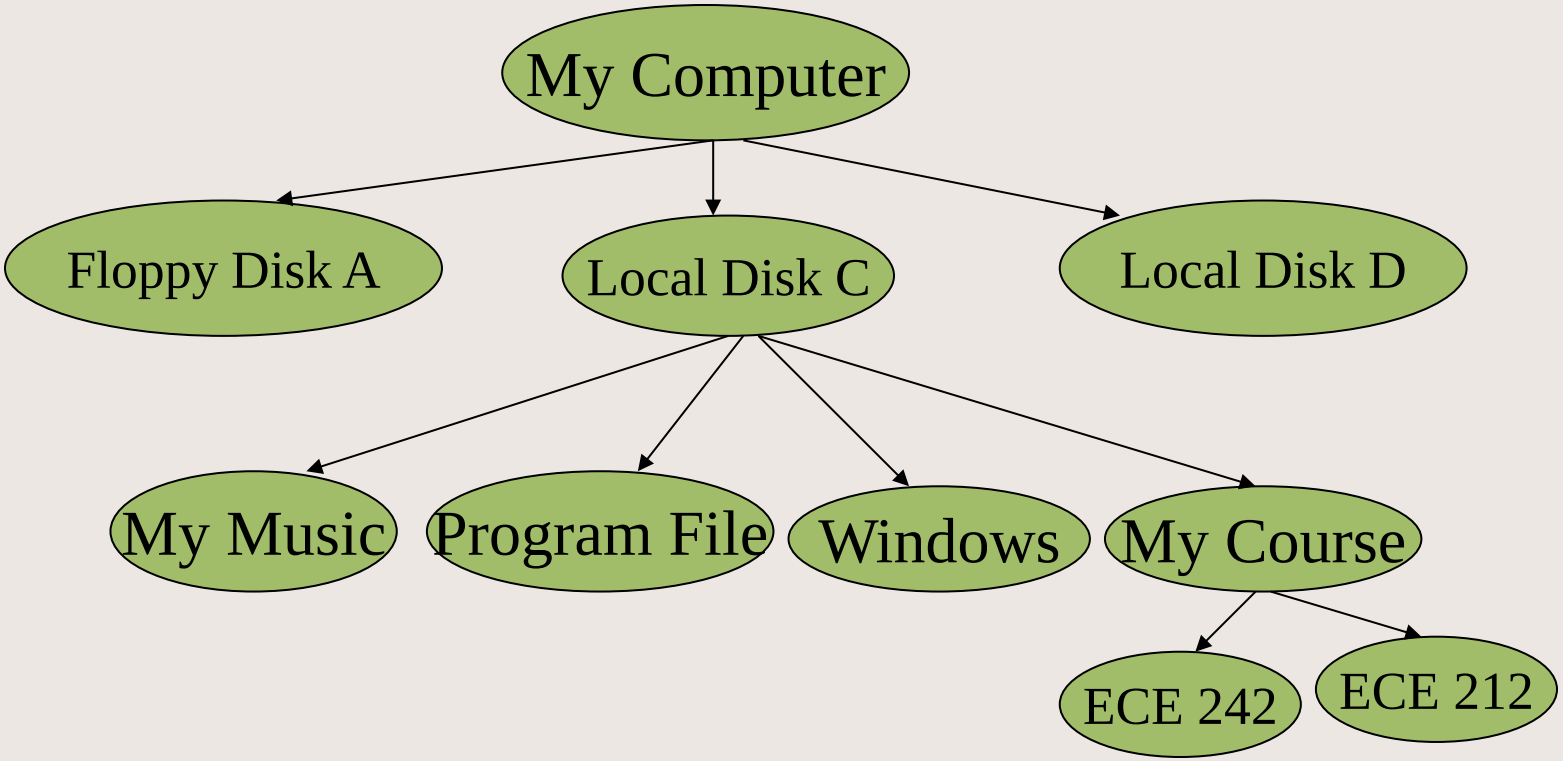
Koki mūsu dzīvē

- Svarīga datu struktūra, kas reprezentē hierarhijas
- Tipiskākie izmantojumi dzīvē:
  - Specificēšanas/skaidrošanas koks (is-a)
  - Sastāvdaļu koks (part-of)
  - Vecāku bērnu attiecības (dzimtas) koks (parent-child)

Organizacionālā struktūra

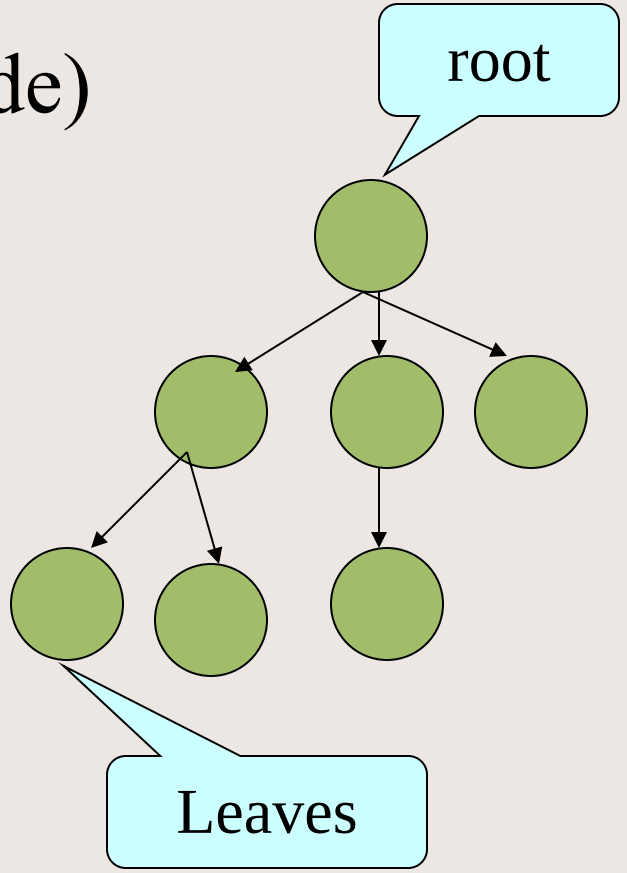


Datņu sistēma



Patvaļīgs koks

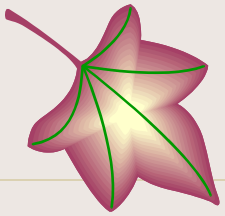
- Sastāv no mezgliem (node) un šķautnēm (edge)
- Augšā ir sakne (root) un apakšā ir lapas (leaves)
- Kokā nav ciklu



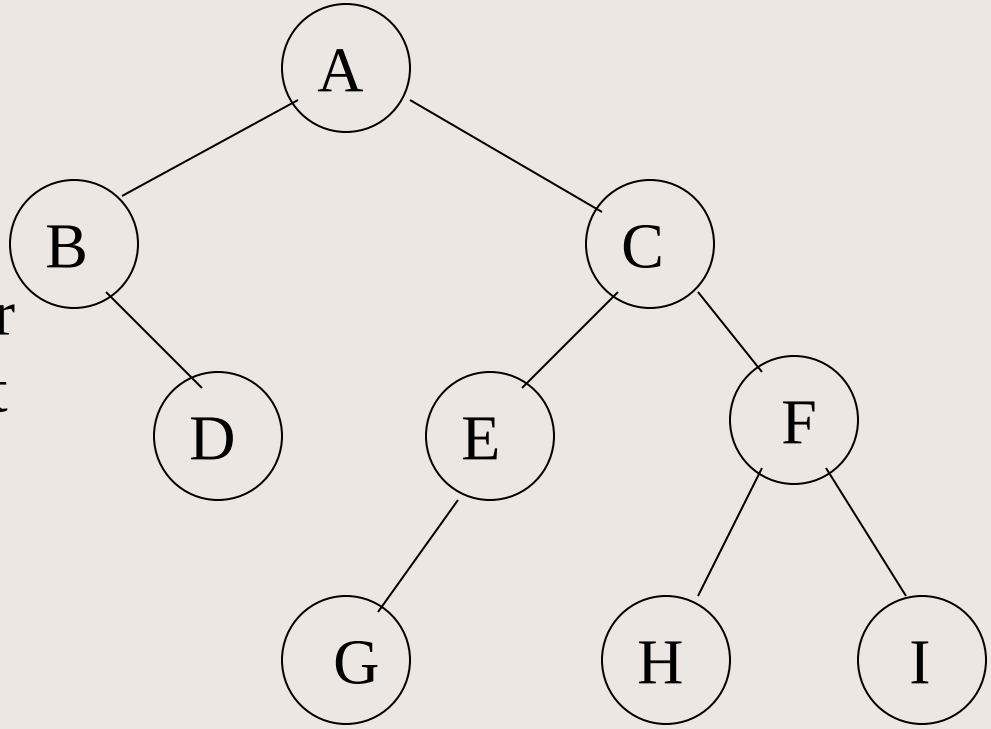
Binārs koks

1. Binārs koks kopa ar galīgu skaitu mezgliem.
2. Binārs koks ir vai nu tukšs vai sastāv no saknes kopā ar diviem bināriem kokiem, kurus sauc par kreiso un labo apakškoku.
3. Kreisajam apakškokam, labajam apakškokam un to saknei nav kopīgu mezglu.

Termini



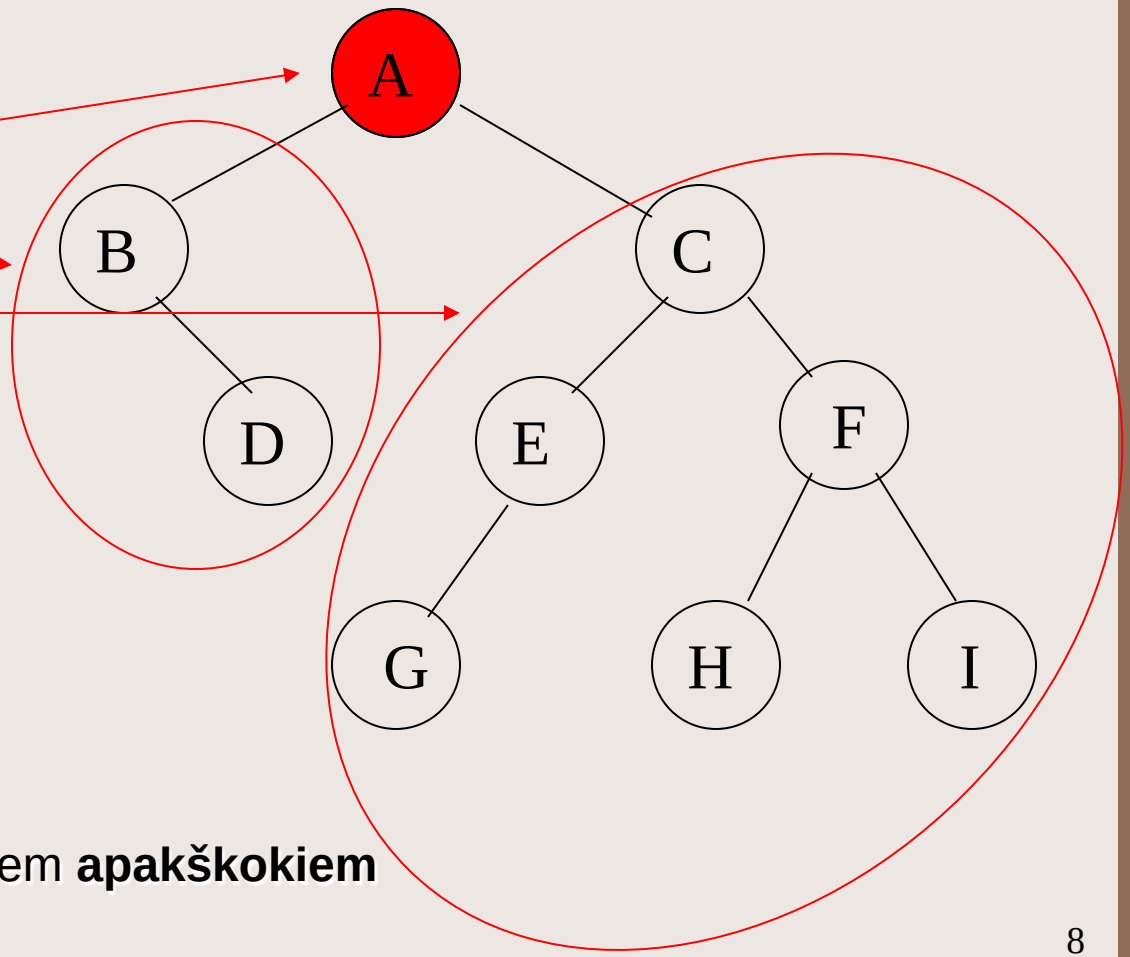
Mezgls - node
Bērns - children
Škautne - edge
Vecāks - parent
Priekštecis - ancestor
Pēctecis - descendant
Ceļš - path
Dziļums - depth
Augstums - height
Līmenis - level
Lapa - leaf
Iekšējs mezgls - internal node
Apakškoks - subtree



Termins - mezgls

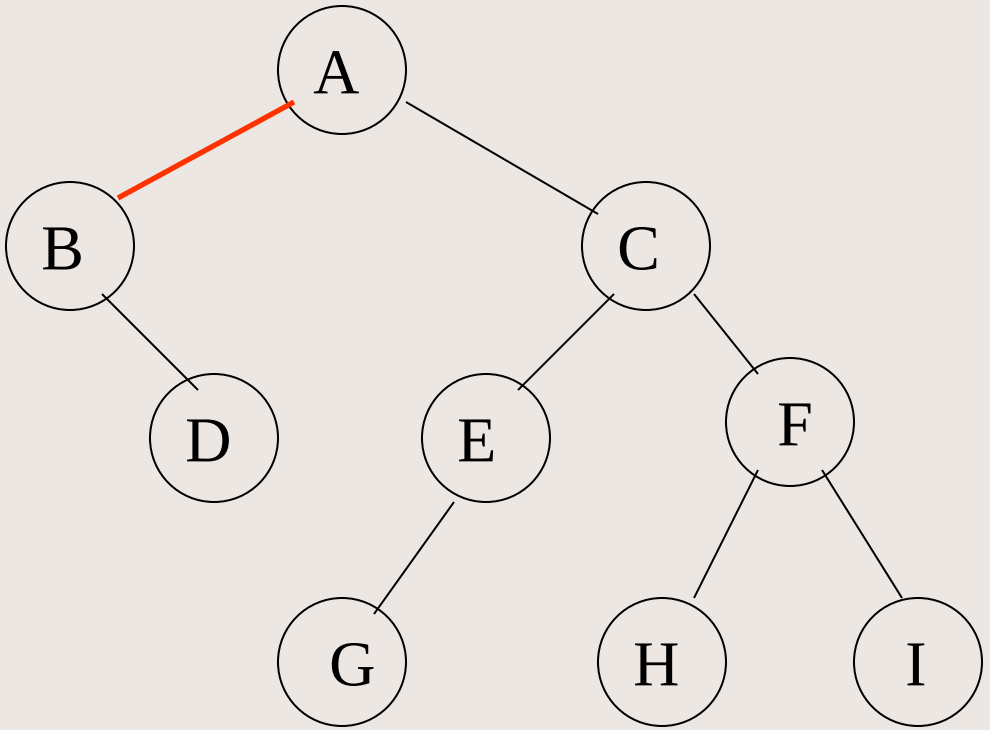
mezgls

- tukšs
- sakne
  - Kreisais apakškoks
  - Labais apakškoks



Sastāv no **saknes** un diviem **apakškokiem**

Termins - šķautne

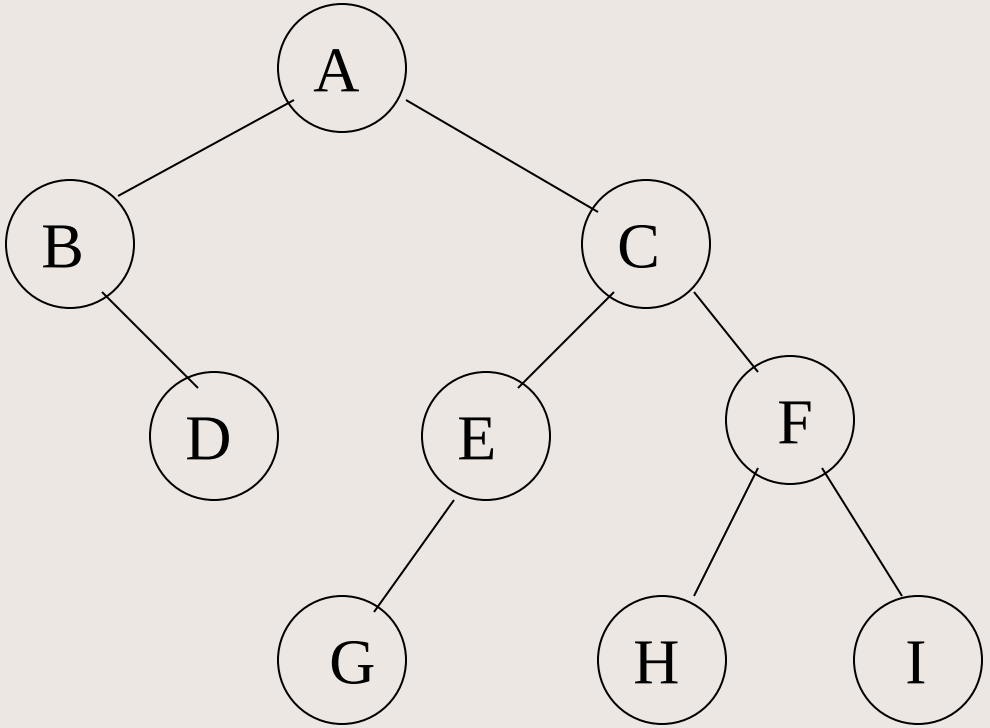


Šķautne -

Te ir **šķautne** no saknes A uz tās bērnu B

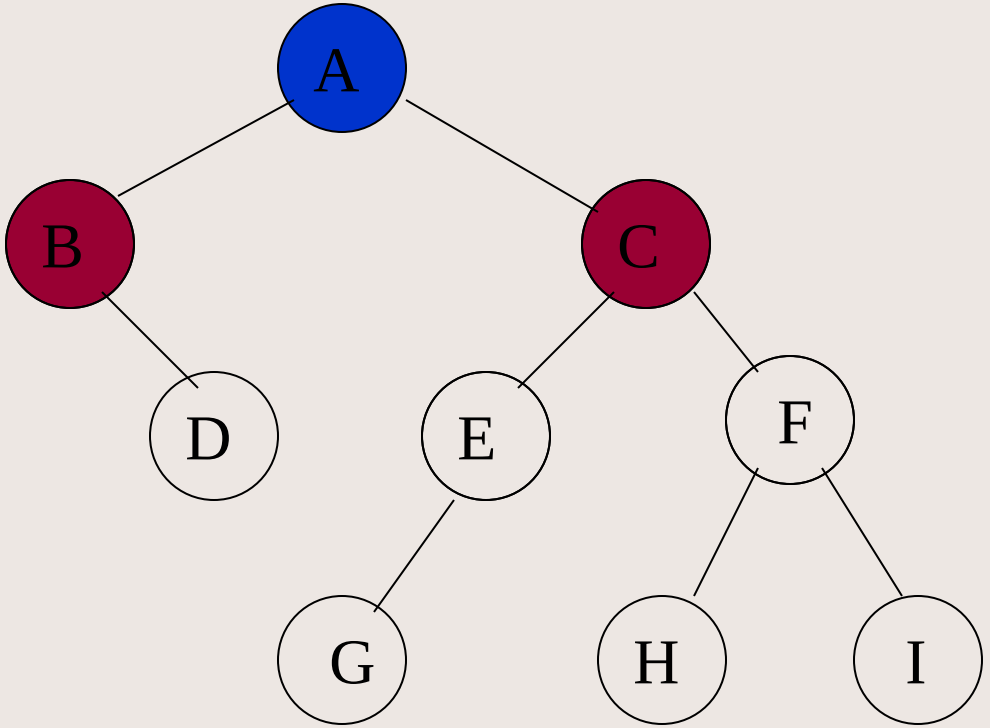
Termins - bērns

bērni



Termins - bērns

bērni

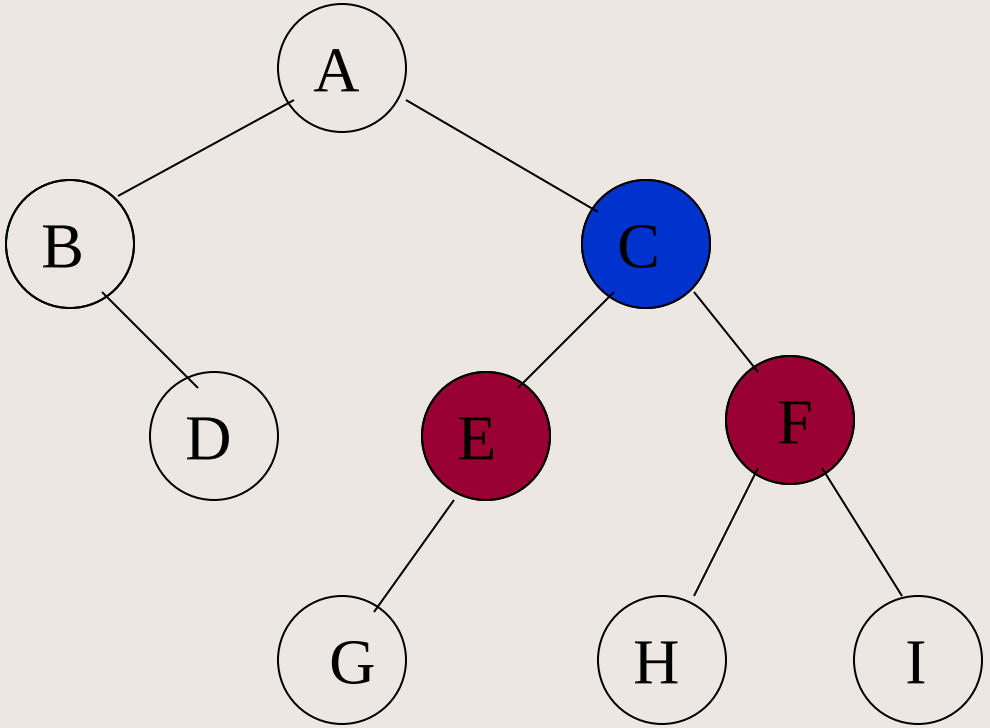


Kas ir mezgla A bērni?

Termins - bērns

bērni

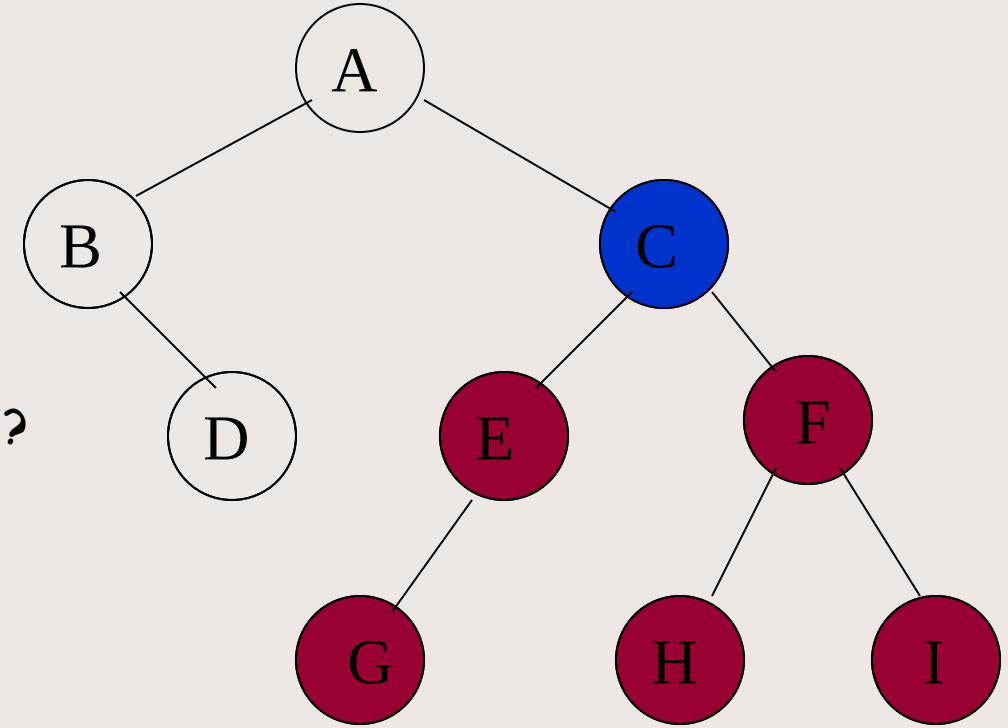
Kas ir mezgla C bērni?



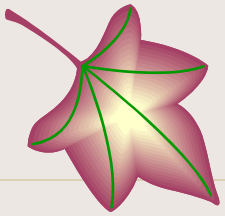
Termins - pēctecis

pēcteči

Kas ir mezgla C pēcteči?

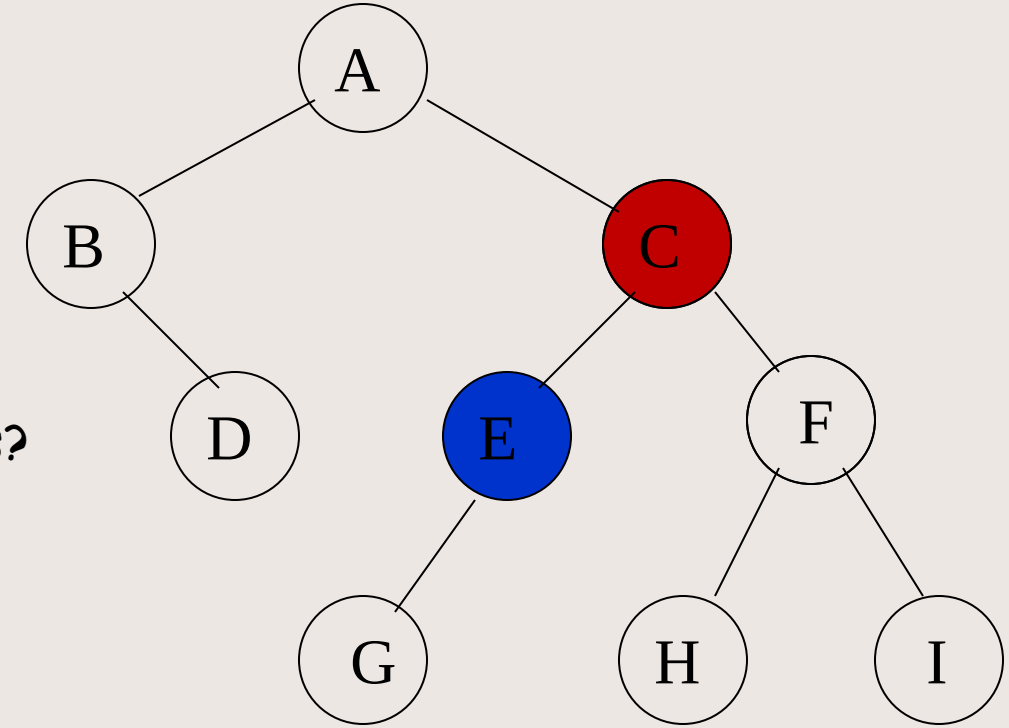


Termins - vecāks



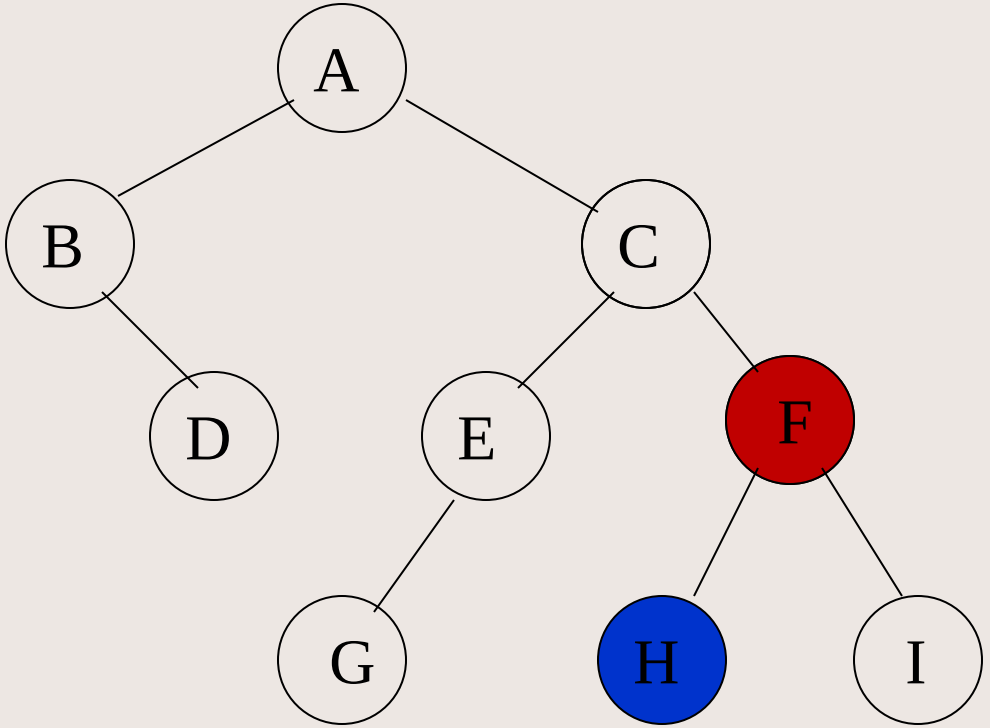
vecāki

Kas ir mezgla E vecāks?



Termins - vecāks

vecāki

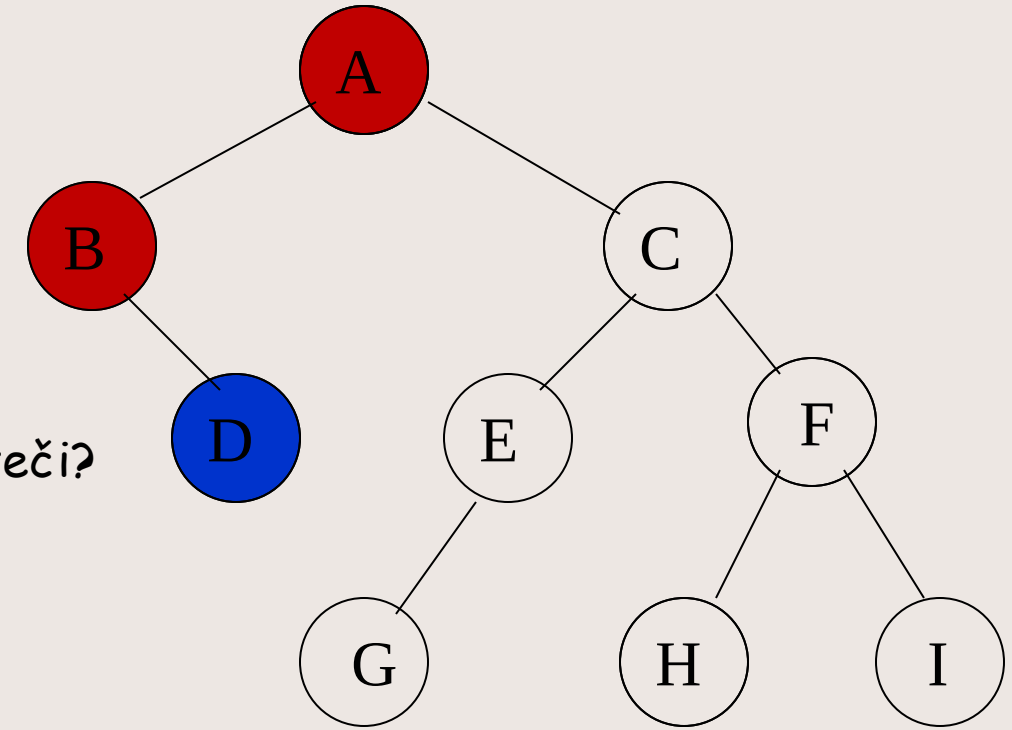


Kas ir mezgla H vecāks?

Termins - priekštecis

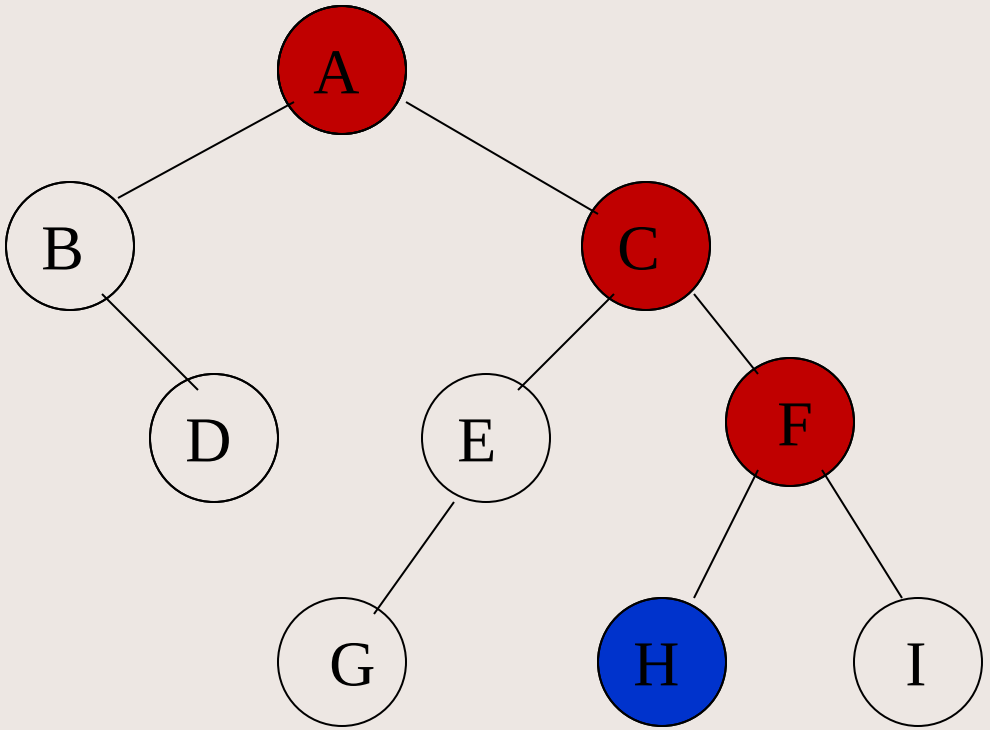
priekšteči

Kas ir mezgla D priekšteči?



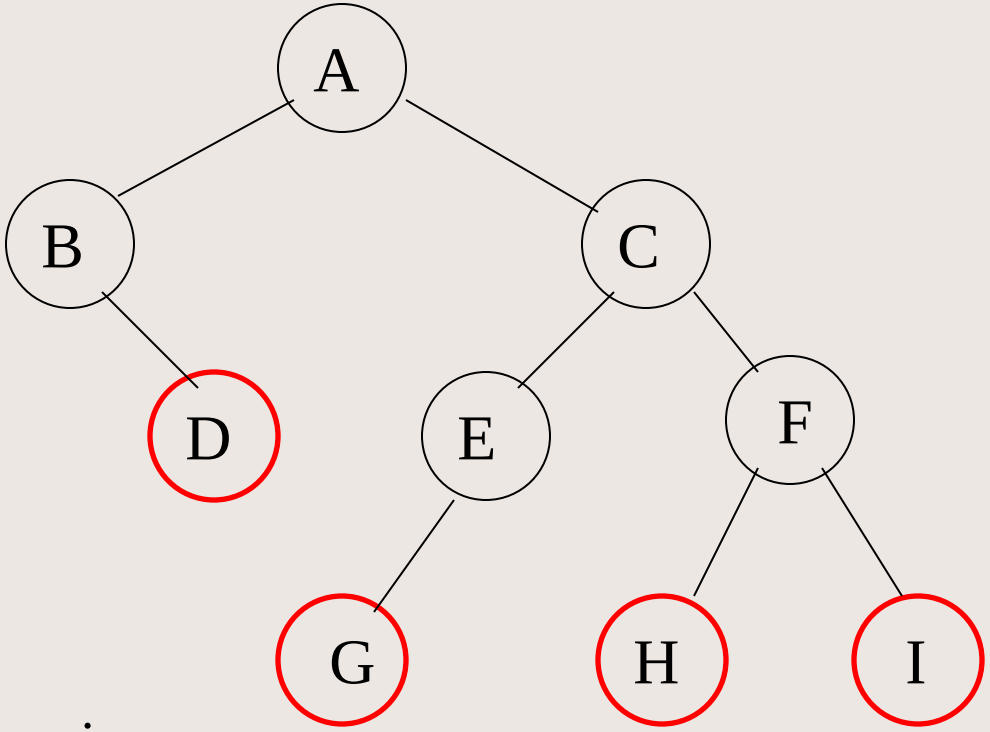
Termins - priekštecis

priekšteči



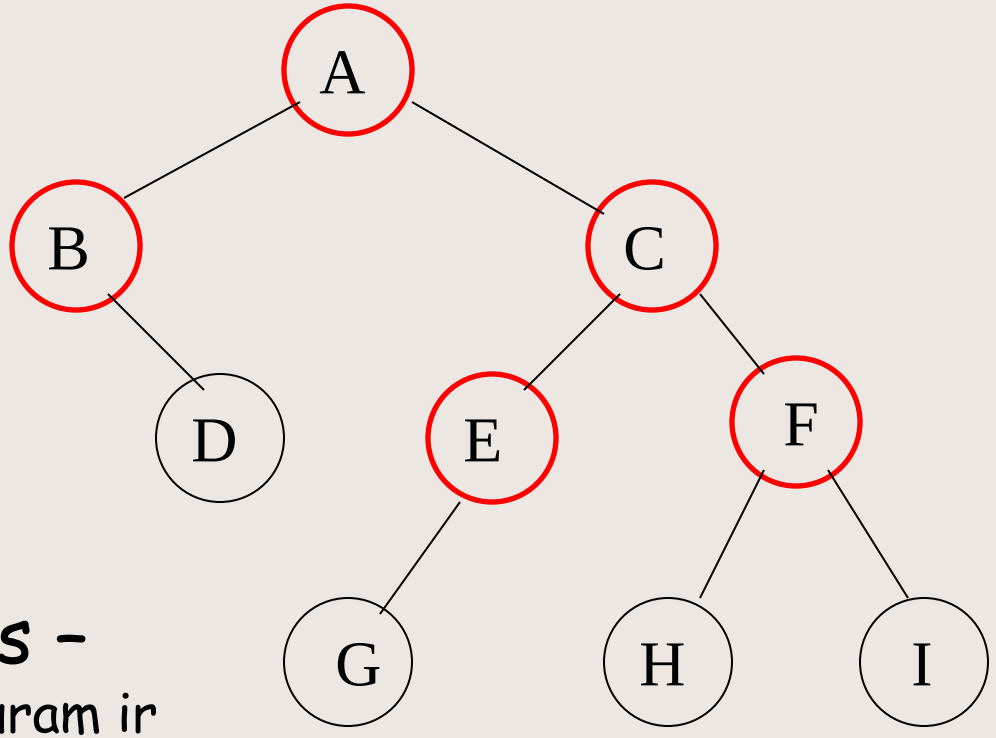
Kas ir mezgla H priekšteči?

Termins – lapa



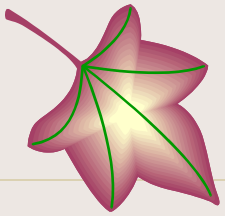
lapa –
Jebkurš mezgls, kuram ir divi tukši mezgli

Termins – iekšējs mezgls

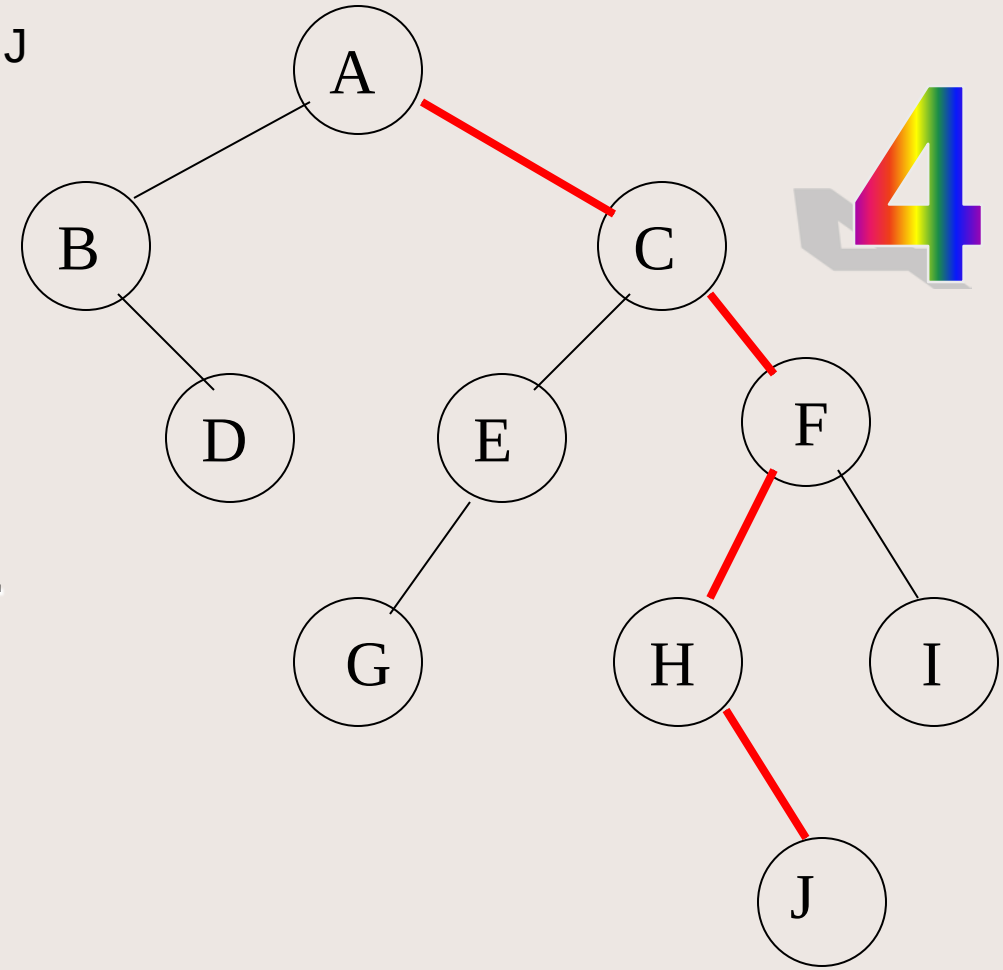


iekšējs mezgls –
Jebkurš mezgls, kuram ir vismaz viens netukšs bērns

Termins - ceļš



no A uz J



ceļš –

Ja n_1, n_2,…n_k ir mezglu secība tāda, ka n_i ir n_{i+1} vecāks, tad virkne ir **ceļš.**

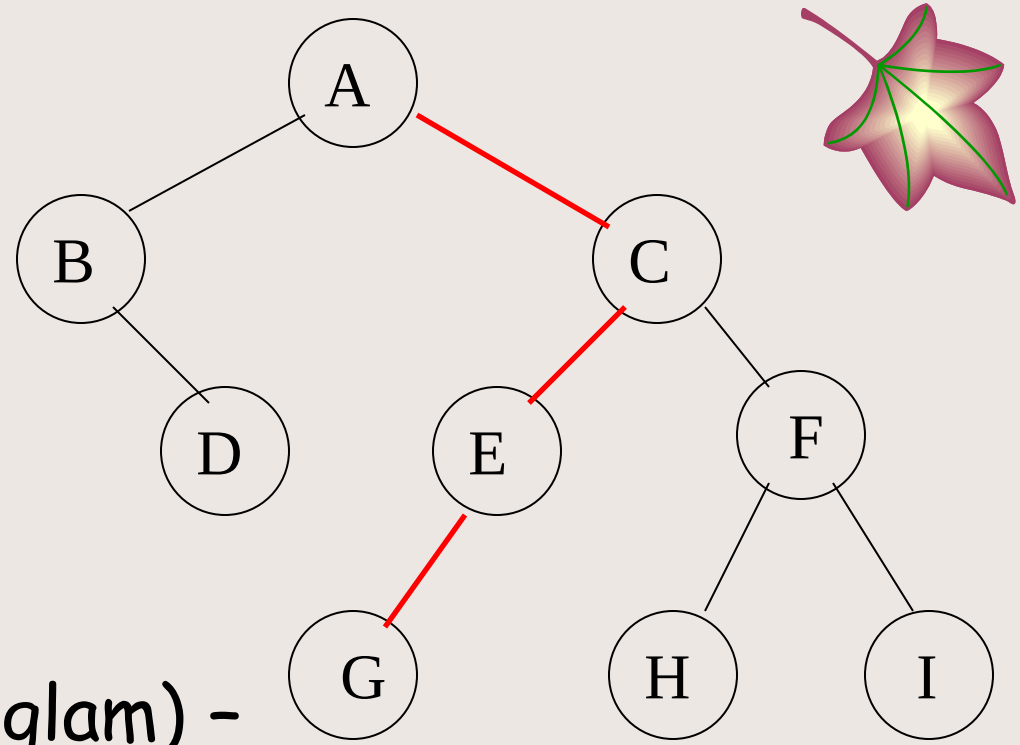
- Ceļa garums ir k-1.

Termins - dziļums

Dziļums -
Garums ceļam no saknes
līdz koka mezglam

Termins – augstums mezglam

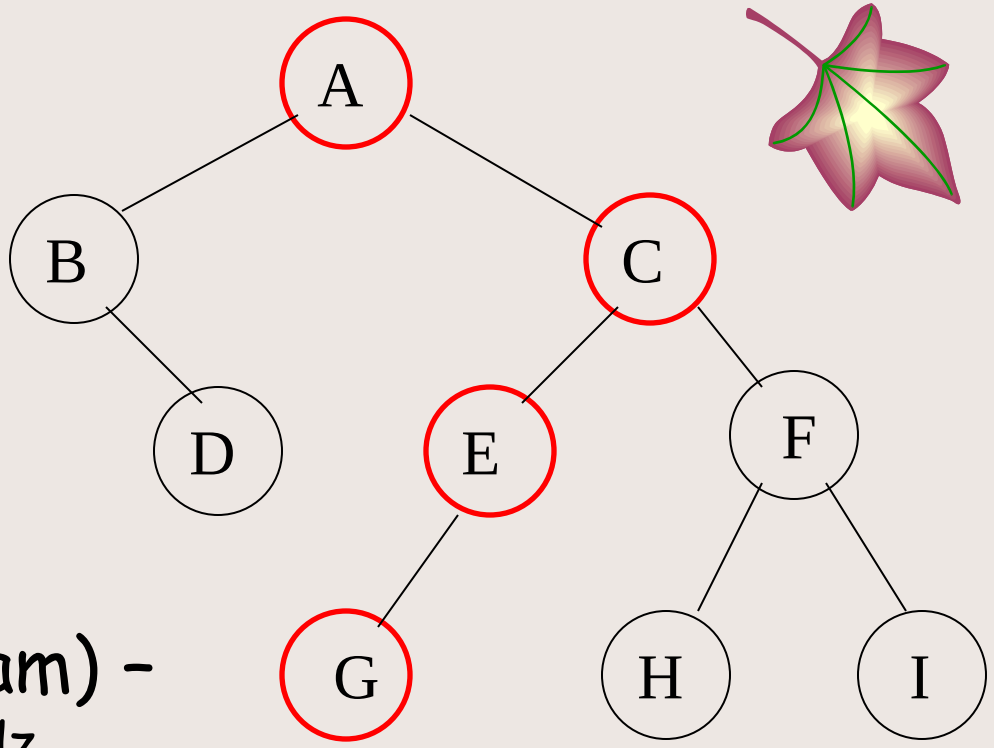
A - 3



augstums (mezglam) -
Garums ceļam no dotā mezgla
līdz dziļākajam mezglam (lapai)

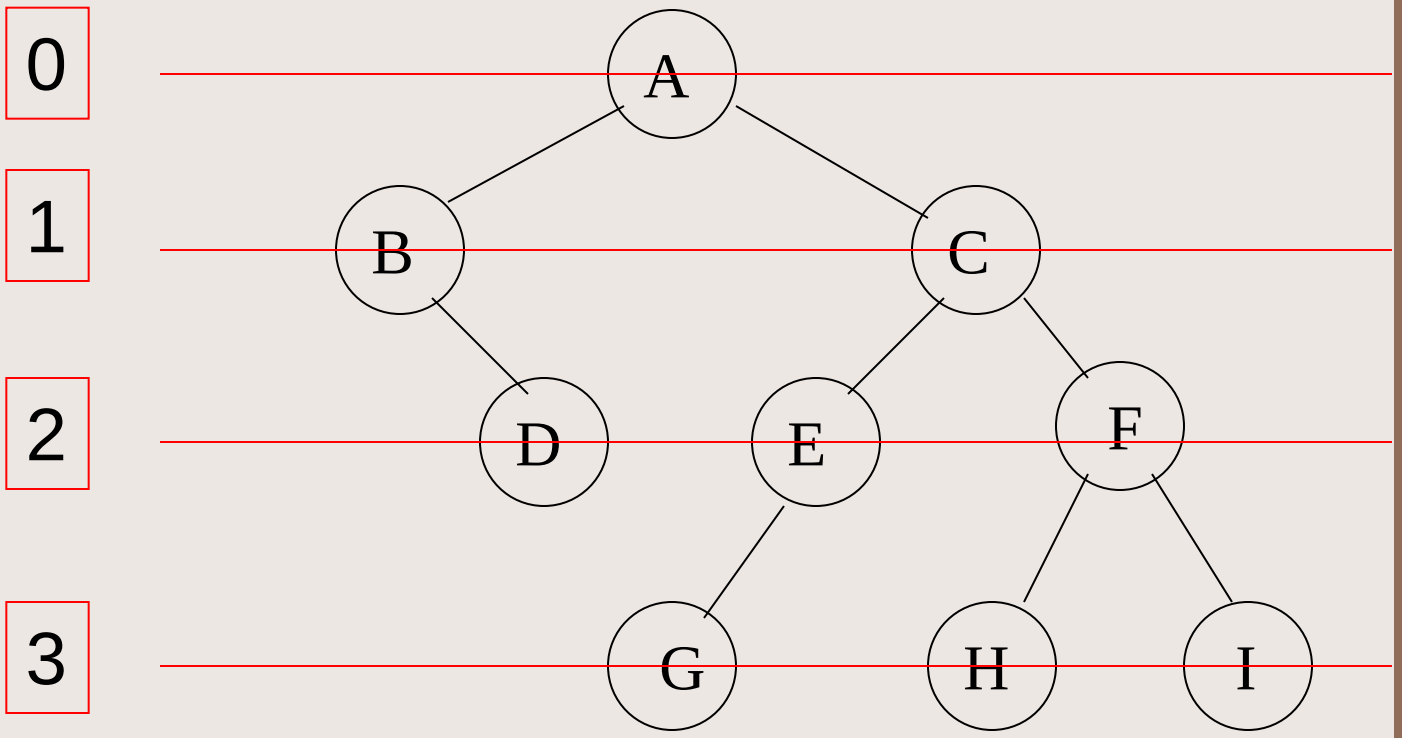
Termins – augstums kokam

3 (4?)



augstums (kokam) –
Dziļums no saknes līdz dziļākajam mezglam (lapai)
(reizēm + 1), ja tiek skaitīts mezglu daudzums ceļā!

Termins – līmenis



līmenis –
visi mezgli ar dziļumu d atrodas koka līmenī d

Reizēm numerāciju sāk ar 1 nevis 0!

Koku pamatveidi 1

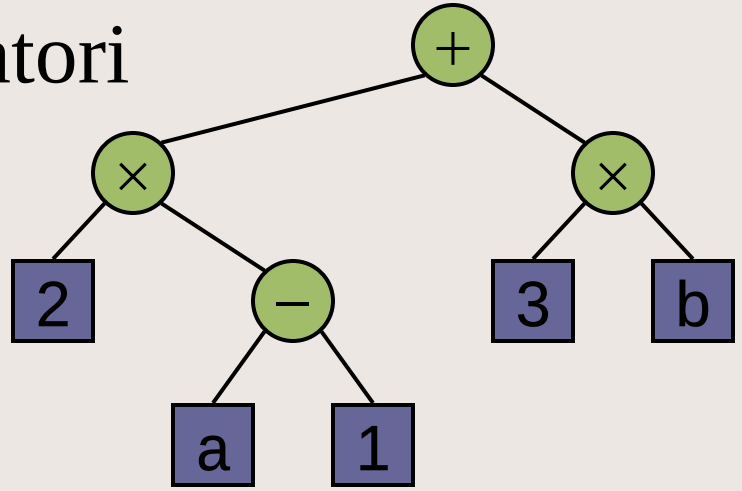
- **Sakārtots** koks (*ordered tree*) - koks kuram bērni ir sakārtoti noteiktā kārtībā, t.i. viennozīmīgi var noteikt, kurš ir pirmais, kurš - otrais, utt..
- **Binārs koks** (*binary tree*) ir sakārtots koks, kuram katram mezglam ir ne vairāk kā divi bērni. Katram bērnam ir viennozīmīgi nosakāms, vai tas ir kreisais (*left*) vai labais (*right*).
- **Tukšs binārs koks** ir koks, kuram nav mezglu. Mēdz apzīmēt ar Λ.

Koku pamatveidi 2

- **Pilns binārs koks** (*full binary tree*) - nav virsotņu, kurām ir tikai viens bērns.
- **Perfekts binārais koks** (*perfect binary tree*) – pilns binārs koks, kuram visas lapas ir ar vienādu dziļumu.
- **Pilnīgs binārs koks** (*complete binary tree*) - ir koks, ko var iegūt perfektam bināram kokam atmetot kādas lapas pēc kārtas no labās puses.

Aritmētikās izteiksmes koks

- Binārais koks bieži asociējas ar aritmētiskās izteiksmes attēlojumu
- Iekšējie mezgli: operatori
- Lapas : operandi

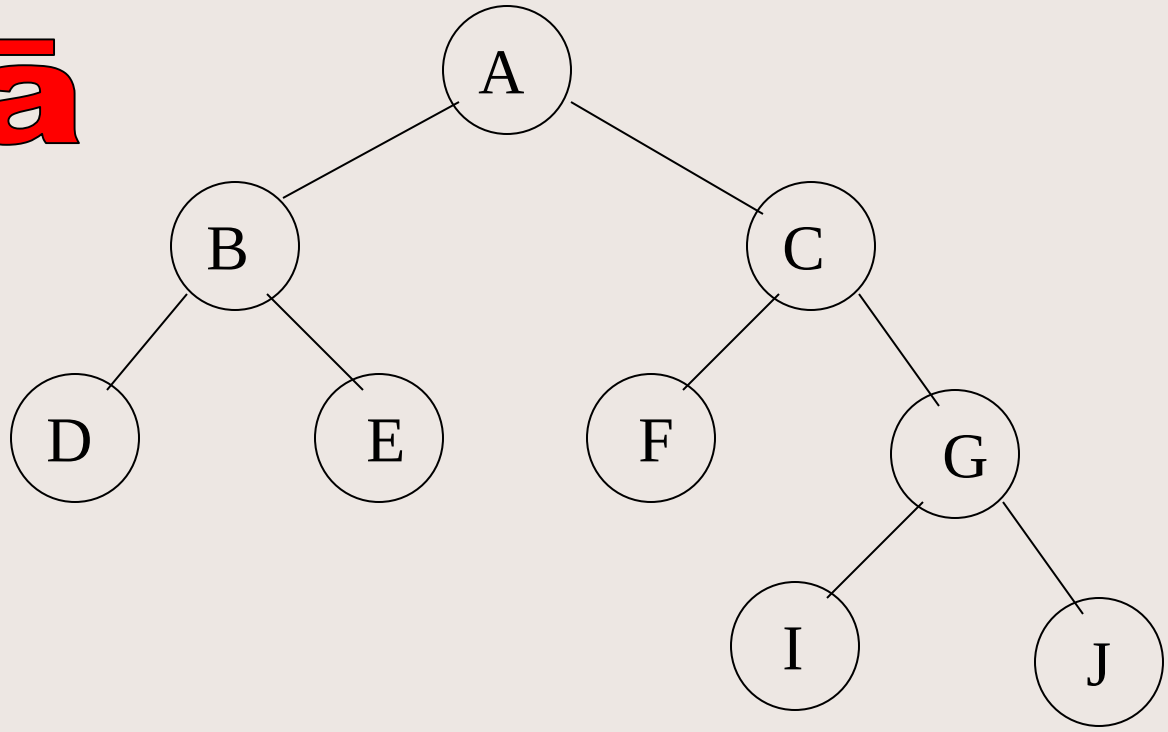


Piemērs: aritmētiskās izteiksmes koks izteiksmei

(2 * (a - 1) + (3 * b))

Termins – pilns binārs koks

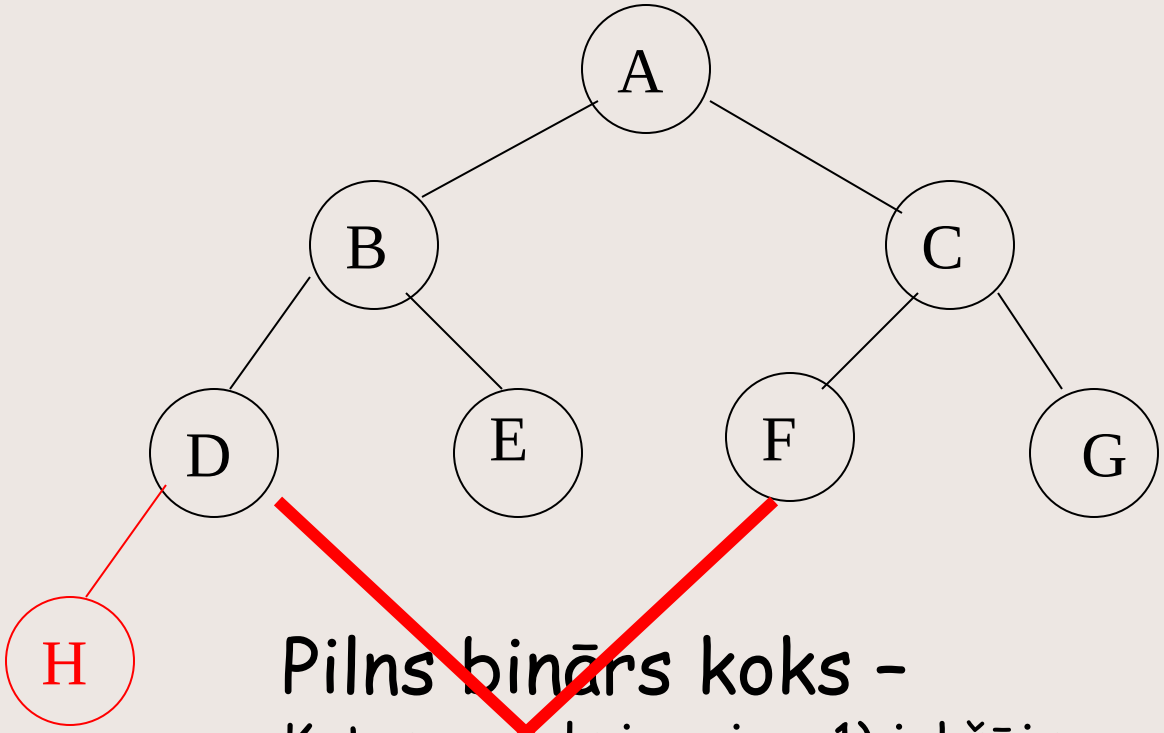
jā



Vai šis ir pilns binārs koks?

Termins – pilns binārs koks

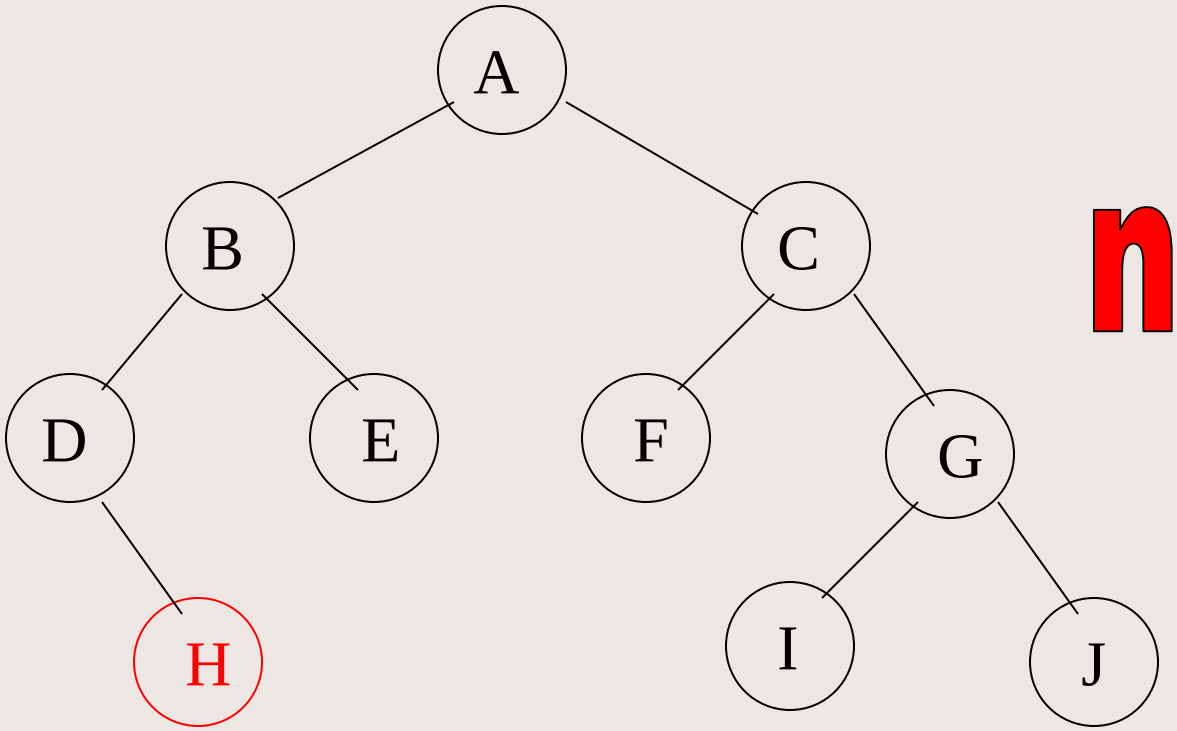
Nē



Vai šis ir pilns binārs koks?

Pilns binārs koks –
Katrs mezgls ir vai nu 1) iekšējs mezgls ar tieši diviem netukšiem bērniem vai 2) lapa

Termins – pilns binārs koks

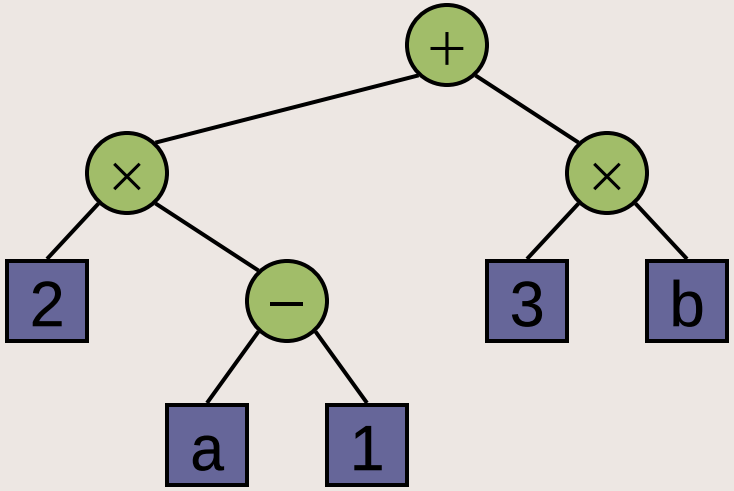


nē

Vai šis ir pilns binārs koks?

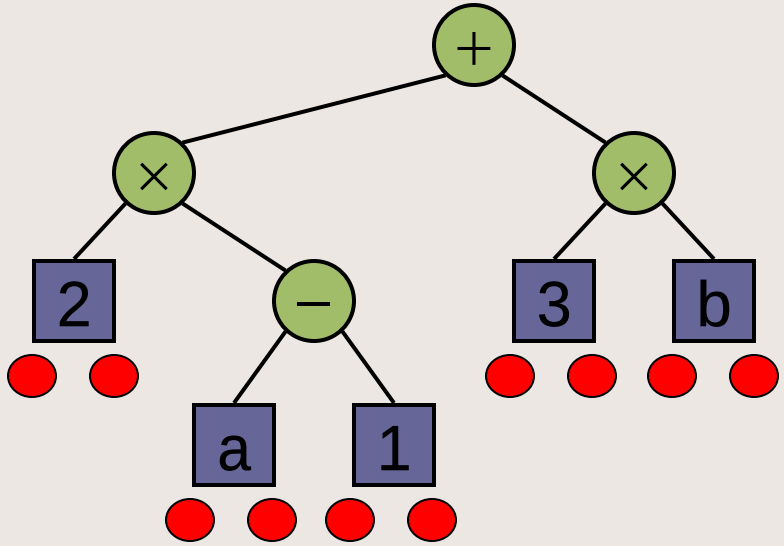
Pilna binära koka teorēma 1

Lapu skaits pilnā binārā kokā ir par vienu lielāks nekā iekšējo mezglu skaits

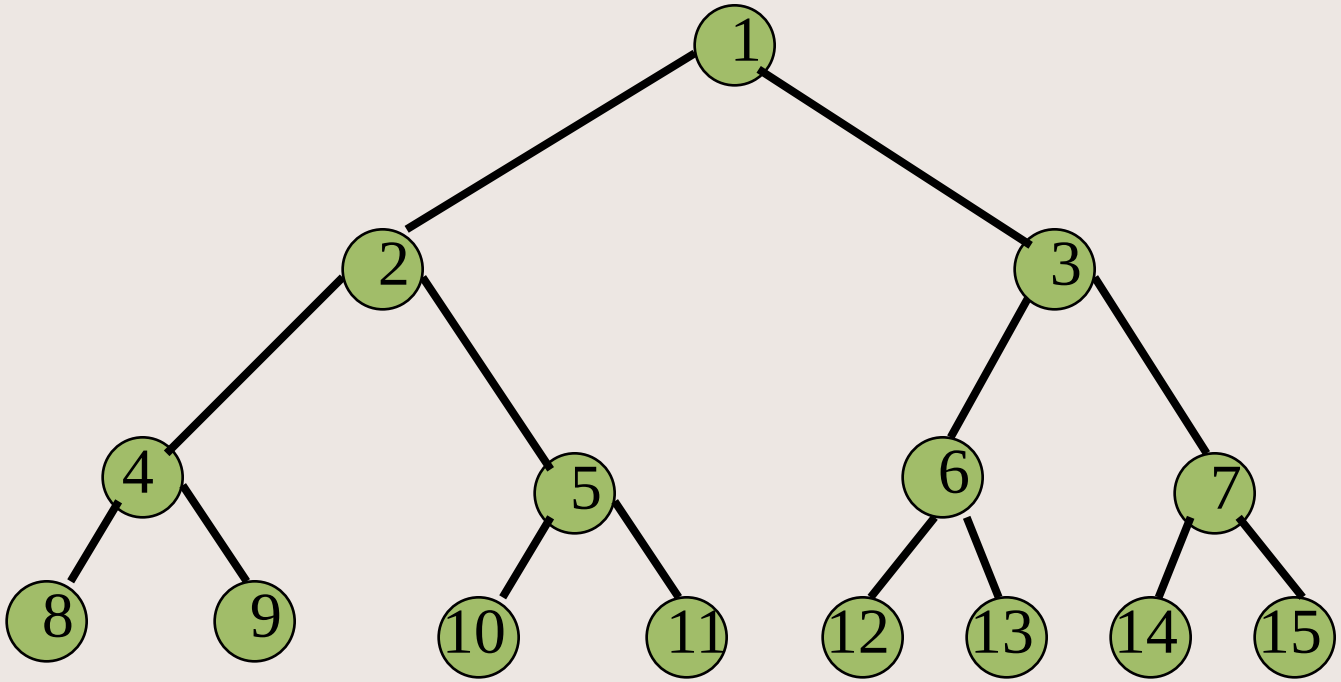


Pilna binära koka teorēma 2

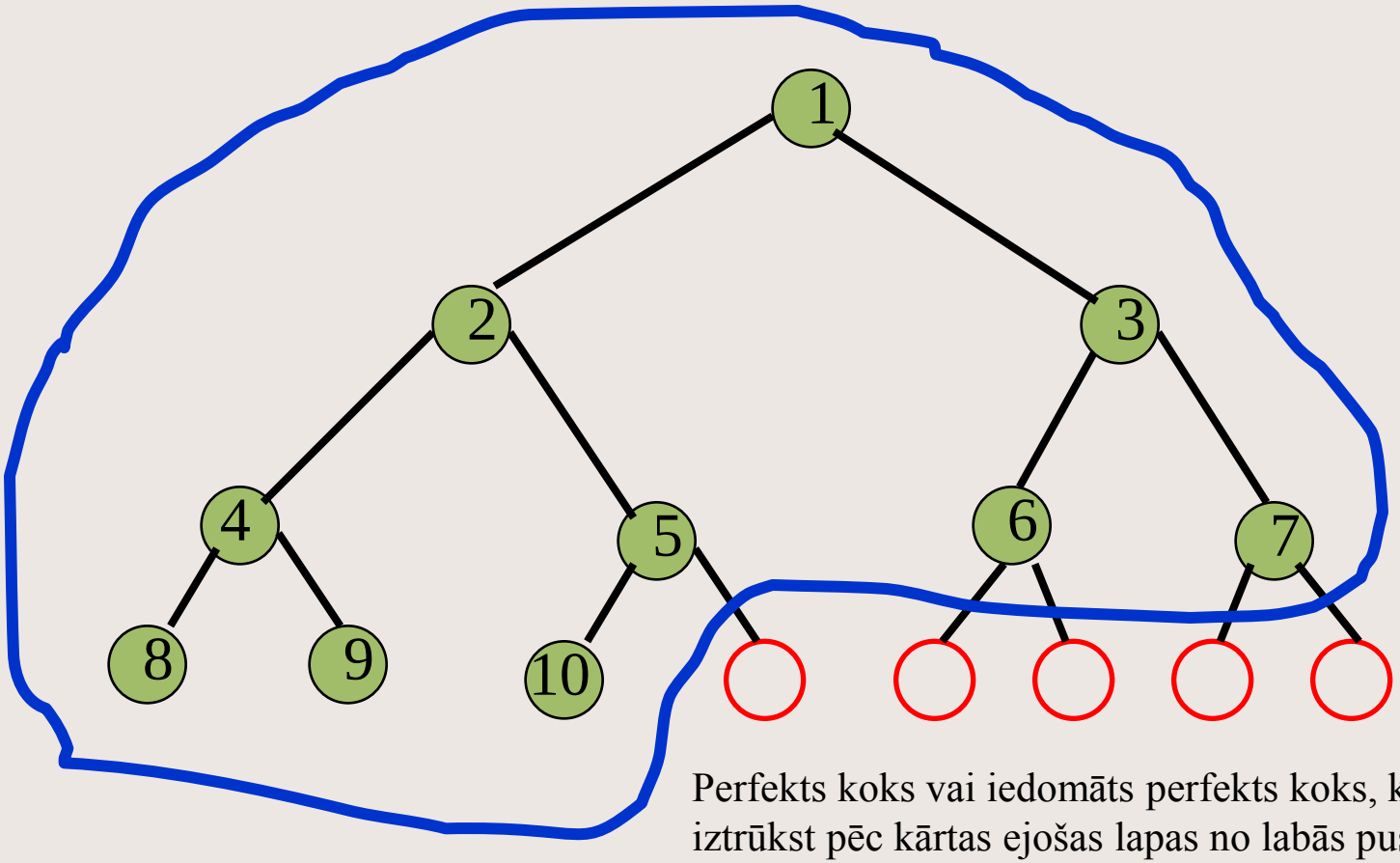
Tukšo apakškoku skaits pilnā netukšā binārā kokā ir par vienu lielāks nekā mezglu skaits



Termini – perfekts



Termini – pilnīgs binārs koks

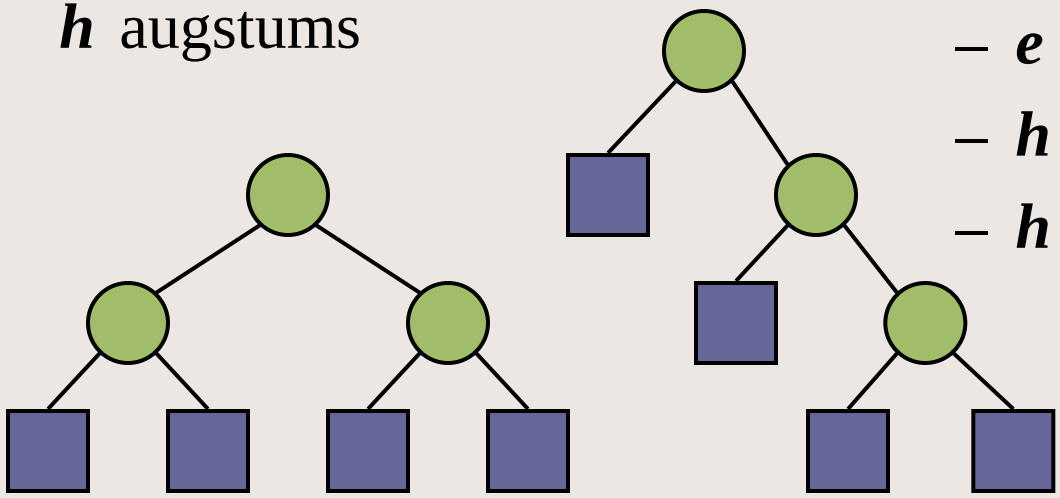


Perfekts koks vai iedomāts perfekts koks, kuram iztrūkst pēc kārtas ejošas lapas no labās puses

Binārā koka īpašības

Apzīmējumi

- n mezglu skaits
- e lapu skaits
- i iekšējo mezglu skaits
- h augstums



Īpašības:

- $e = i + 1$
- $n = 2e - 1$
- $h \leq i$
- $h \leq (n - 1)/2$
- $e \leq 2^h$
- $h \geq \log_2 e$
- $h \geq \log_2 (n + 1) - 1$

Pamatoperācijas ar kokiem 1

- ***Parent***(v): Atdod mezgla v vecāku vai Λ, ja v ir sakne.
- ***Children***(v): Atdod kopu ar mezgla v bērniem (kopa ir tukša, ja v ir lapa).
- ***FirstChild***(v): Atdod mezgla v pirmo bērnu vai Λ, ja v ir lapa.
- ***RightSibling***(v): Atdod labo kaimiņu mezglam v, vai Λ, ja v ir sakne vai ja v ir pats labējais bērns savam vecākam.
- ***LeftSibling***(v): Atdod kreiso kaimiņu mezglam v, vai Λ, ja v ir sakne vai ja v ir pats kreisējais bērns savam vecākam.

Pamatoperācijas ar kokiem 2

- ***LeftChild***(v), ***RightChild***(v): Atdod kreiso (labo) bērnu mezglam v (Λ, ja nav kreisā (labā) bērna).
- ***IsLeaf***(v): Atdod *true*, ja v ir lapa, *false* - ja v nav lapa.
- ***Depth***(v): Atdod mezgla v dziļumu kokā.
- ***Height***(v): Atdod mezgla v augstumu kokā.

Koka apstaiga

- Jebkuru koka mezglu apmeklēšanu kādā noteiktā secībā sauc par koka apstaigu
- Ar "mezgla apmeklēšanu" sapratīsim, ka tiek veikta kāda darbība ar informāciju, ko satur mezgls
- Mezgla apmeklēšanu programmās reprezentē funkcija *Visit()*.

Preorder apstaiga 1

- Mezgls tiek apmeklēts pirms tiek apmeklēts kāds no šī mezgla apakškokiem

procedure *Preorder*(**pointer** P):

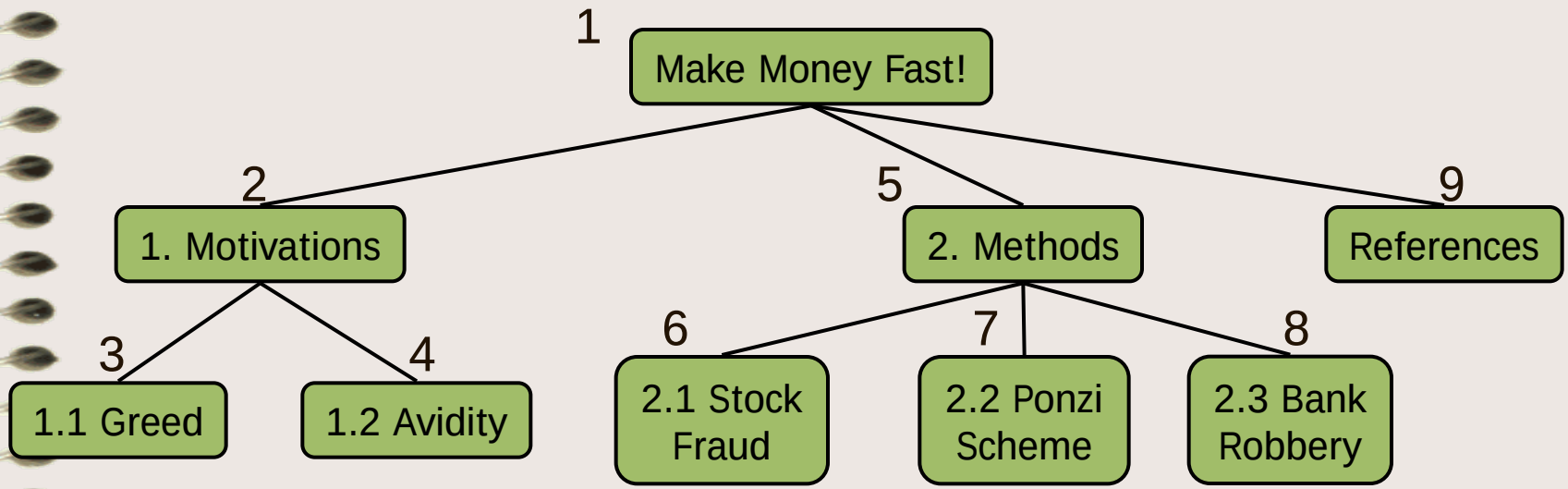
Visit(P)

foreach child Q **of** P, in order, **do**

Preorder(Q)

Preorder apstaiga 2

Pielietojums: strukturēta dokumenta izdrukāšana (pilns teksts vai satura rādītājs)



Postorder apstaiga 1

- Mezgls tiek apmeklēts tikai tad, kad ir jau apmeklēti visi šī mezgla bērni.

procedure *Postorder*(**pointer** P):

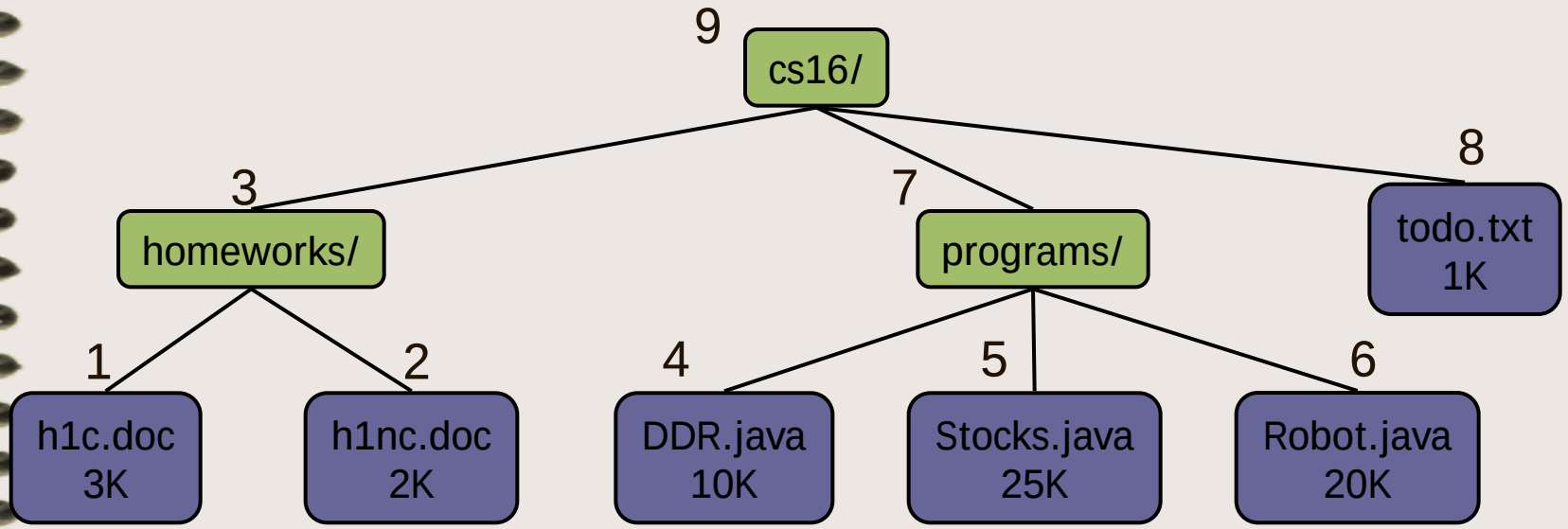
foreach child Q **of** P, in order, **do**

Postorder(Q)

Visit(P)

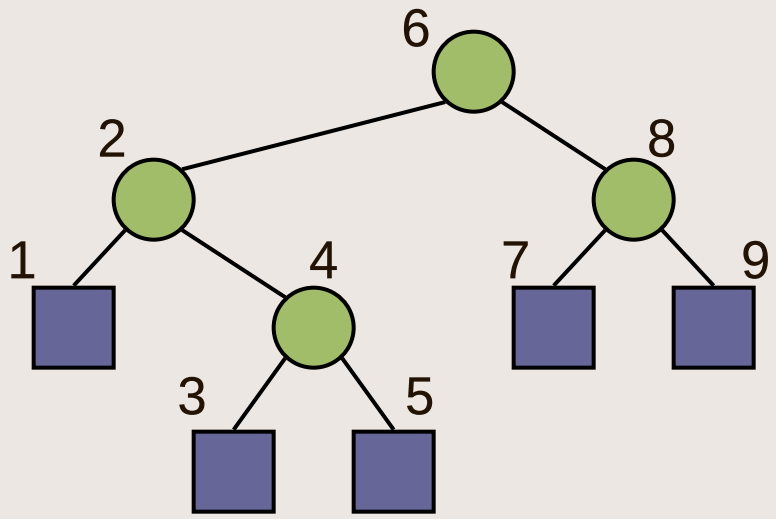
Postorder apstaiga 2

Pielietojums: direktorijas izmēra aprēķins (visi faili pašā direktorijā un visās tās apakšdirektorijās)



Inorder apstaiga 1

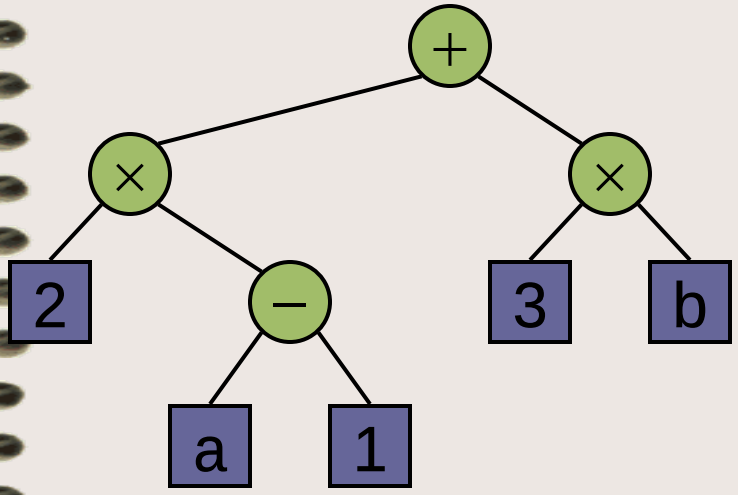
- Spēkā tikai bināriem kokiem. Mezgls tiek apmeklēts pēc tam, kad ir apmeklēts kreisais apakškoks, bet vēl nav apmeklēts labais apakškoks.



Inorder apstaiga 2

```
procedure Inorder(pointer P):
    {P ir binära koka sakne}
    if P=Λ then
        return
    else
        Inorder(LeftChild(P))
        Visit(P)
        Inorder(RightChild(P))
```

Aritmētiskās izteiksmes drukāšana




```
Algorithm printExpr (v)
    if isInternal (v)
        print(“(”)
        printExpr(leftChild (v))
    print(v.element ())
    if isInternal (v)
        printExpr(rightChild (v))
        print (“)”)
```

((2 * (a - 1)) + (3 * b))

Procedūra Traverse 1

procedure *Traverse*(**pointer** *P*):

{apstaigā koku, kura sakne ir P}

if $P \neq \Lambda$ **then**

PreVisit(*P*)

Traverse(LC(*P*))

InVisit(*P*)

Traverse(RC(*P*))

PostVisit(*P*)

Procedūra Traverse 2