DUOMENŲ STRUKTŪROS IR ALGORITMAI

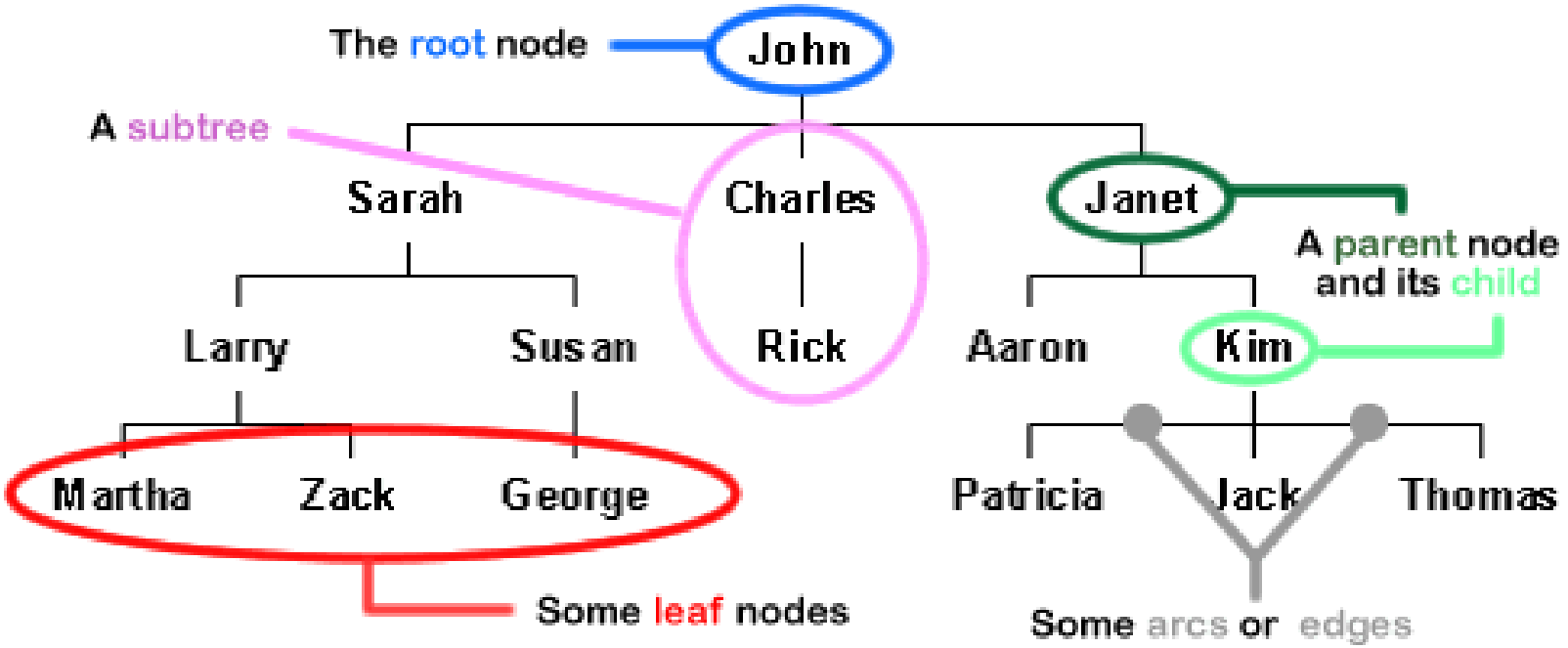
Hierarchinės duomenų struktūros:

Dvejetainiai medžiai

Praeitos paskaitos santrauka

- Tiesinės dinaminės duomenų struktūros:
  - Sąrašai
  - Stekas (dėklas)
  - Eilė
  - Dekas
- Tiesinių duomenų struktūrų pavyzdžiai su masyvais
- Tiesinių duomenų struktūrų pavyzdžiai su sąrašais

Medžio struktūra

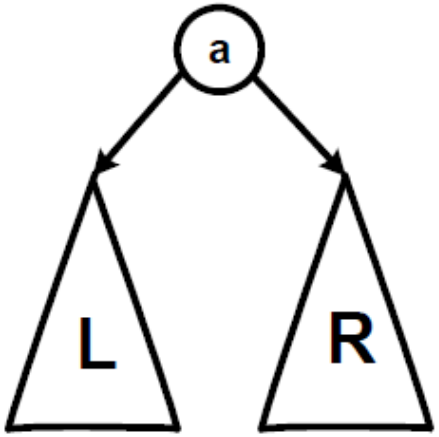


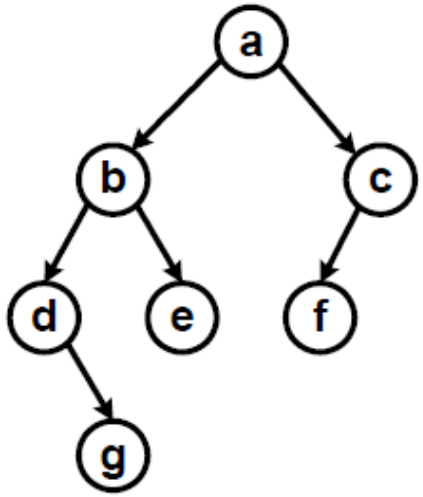
Medžiai – tai netiesinės duomenų struktūros, kurių pagrindinis privalumas labai greita paieška.

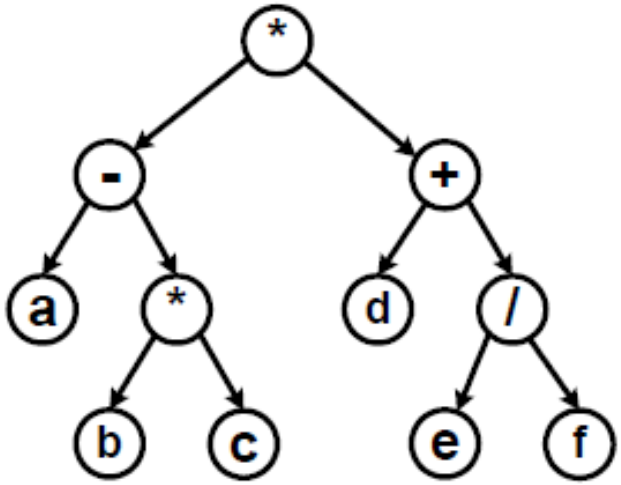
Terminai ir medžio savybės

- Medis – tai ciklų neturintis orientuotas grafas.
- Medį sudaro baigtinis elementų skaičius, kurie vadinami **viršūnėmis**.
- Medžio viršūnė, į kurią neįeina jokia kita viršūnė, vadinama **šaknimi**.
- Į kiekvieną viršūnę, išskyrus šaknį, įeina ne daugiau nei viena viršūnė.
- Medžio dalis, išeinanti iš bet kurios viršūnės, vadinama **šaka**.

Pavyzdžiai







$$(a - b * c) * (d + e/f)$$

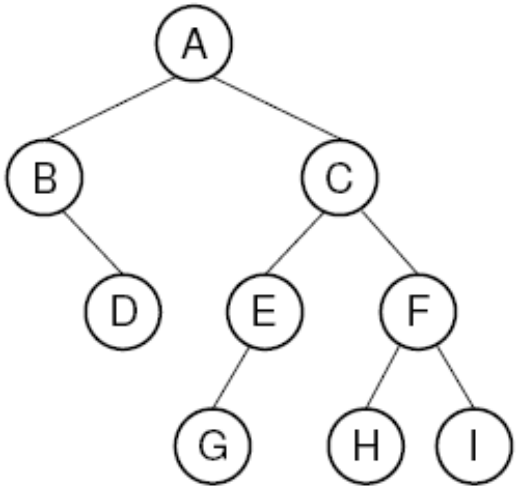
Medžio savybės

- Jei n_1, n_2, ..., n_k yra medžio viršūnės, ir n_i yra viršūnės n_{i+1} tėvas, tada kelias nuo viršūnės n_1 iki n_k bus vadinamas k - 1 keliu. Iš šaknies į kiekvieną viršūnę veda vienintelis kelias.
- Viršūnės, nutolusios nuo šaknies atstumu *k*, vadinamos *k-ojo* lygmens viršūnėmis.
- Viršūnės, kurios neturi dukterinių viršūnių, vadinamos **lapais**.
- **Medžio aukštis** yra vienetu didesnis už gylį iki giliausios viršūnės.
- **Šaknies gylis** lygus 0, šaknis yra nuliniame lygyje.
- Medžiai gali būti: dvejetainiai, trejetainiai, ketvirtainiai ir t.t.

Dvejetainis medis

Dvejetainiu vadinamas orientuotas medis, kuriame į kiekvieną viršūnę, išskyrus šaknį, įeina viena briauna, o išeina ne daugiau kaip dvi.

Dvejetainis medis turi ne daugiau nei du viršūnes-vaikus.



Medžio aukštis - 4
Viršūnės H gylis - 3
Šaknis - A
Lapai – D G H I

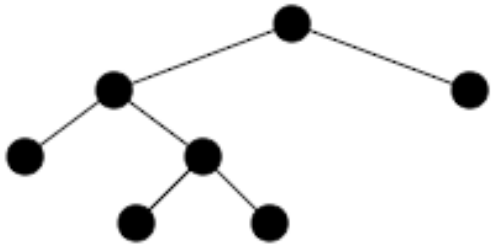
Pilnas ir subalansuotas medis

Pilnu (*full*) medžiu vadiname tokį medį, kuriame kiekviena viršūnė turi du vaikus arba yra lapas.

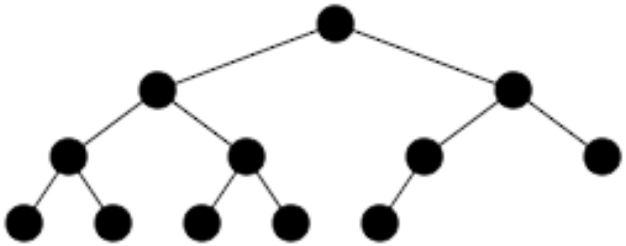
Subalansuotas medis – tai toks medis, kuriame viršūnės pildomos iš kairės į dešinę tol, kol užpildomas visas lygis ir tik tuomet pereinama į sekantį lygį.

Subalansuoto medžio visi lygiai išskyrus paskutinį yra pilnai užpildyti. Paskutinis lygis pildomas iš kairės į dešinę.

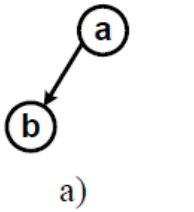
Dvejetainis medis



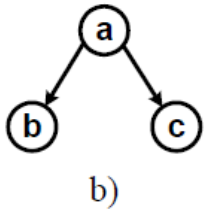
Pilnas (nesubalansuotas) medis.



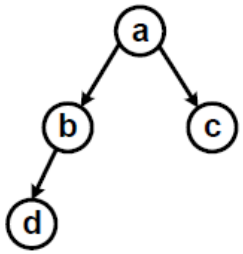
Subalansuotas (nepilnas) medis.



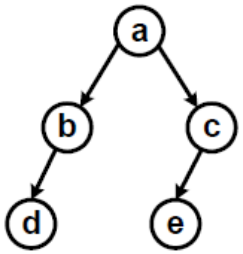
a)



b)



c)



d)

Subalansuoti medžiai

Dvejetainio medžio formavimas (1)

Panagrinėkime algoritmą, dvejetainio medžio formavimui.

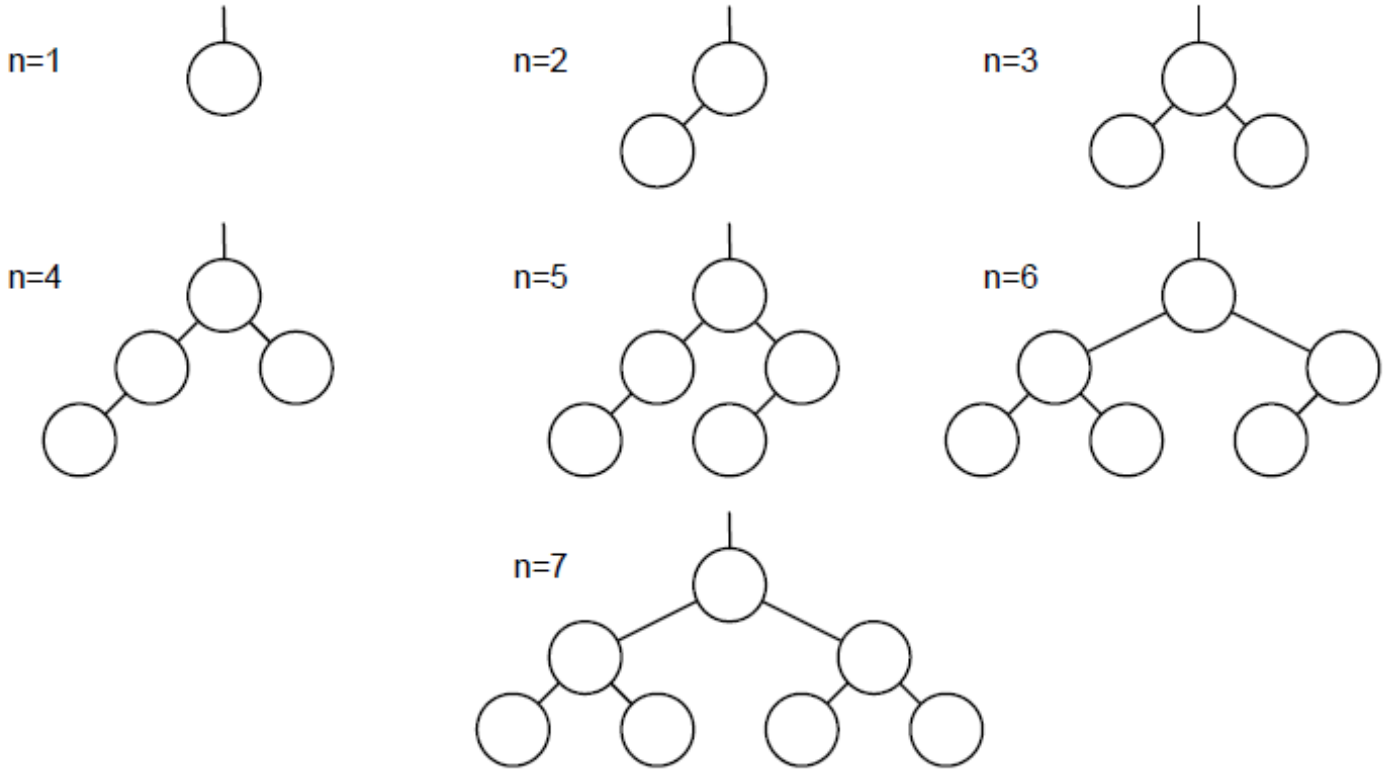
Tarkime iš duomenų $n_1, n_2, \ldots, n_k$ reikia sudaryti dvejetainio medžio struktūrą.

Dvejetainio medžio formavimo algoritmas (1):

1. n_1 – įrašomas šaknyje.
2. n_2 lyginamas su n_1. Jei $n_2 < n_1$, tuomet jungiamas prie kairės šakos, priešingu atveju – prie dešinės šakos.
3. Antras žingsnis kartojamas su kitais n_i elementais, pradedant lyginimą nuo šaknies ir tęsiant tol, kol elementas tampa lapu.

Dvejetainio medžio formavimas (2)

Mažiausio aukščio dvejetainio medžio formavimo algoritmas (2). Viršūnės išdėstomos taip, kad būtų užpildytas maksimaliai visas žemiausias lygis pradedant pildyti vienodai nuo kairės kiekvienos viršūnės šakos, po to pereinant prie dešiniosios šakos.



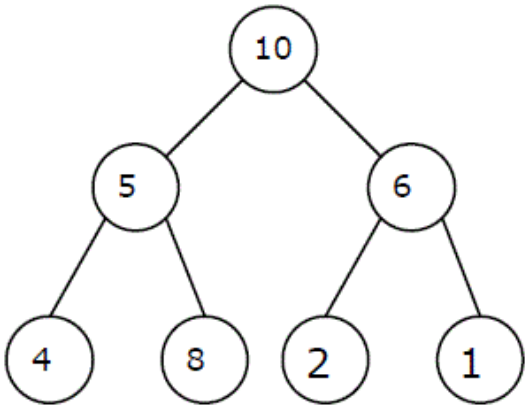
Pilnas subalansuotas medis

Kiek viršūnių ***N*** turi pilnas subalansuotas medis, kurio gylis ***h*** **?**

$$N = 1 + 2^1 + 2^2 + ... + 2^h$$

Arba $N = 2^{h+1} - 1$

kur $h = (\log_2 N)$

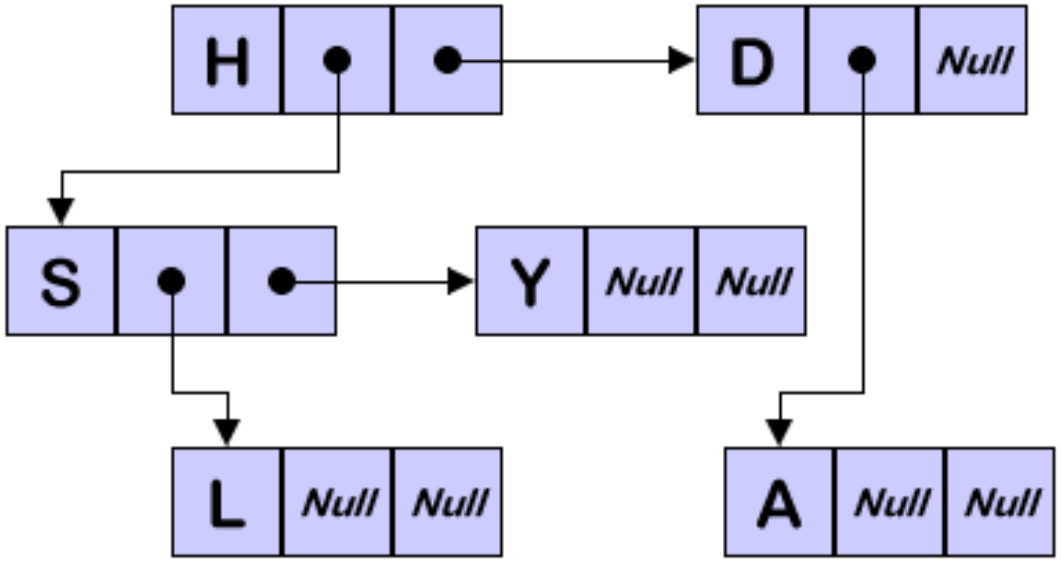


Norint surasti bet kurią viršūnę tokiame dvejetainiame medyje, blogiausiu atveju tai užims ***h***+1 arba $(\log_2 N)$ žingsnių.

Dvejetainio medžio elementas

Dvejetainio medžio viršūnės elementas susideda iš duomenų ir dviejų rodyklių.

Dvejetainis medis



Atminties perteklinis poreikis

Realizuojant medį, reikalingas atminties kiekis S:

$$S = N(2P + D)$$

kur N – viršūnių skaičius, P – atminties kiekis, skirtas rodyklei, D – atminties kiekis duomenims.

Perteklinis atmintis kiekis, realizuojant medį:

$$S_{pert} = 2*P*N.$$

Naudingos ir visos atmintis santykis:

$$S_{naud} / S_{visas} = D / (2P + D).$$

Viršūnės elementas

```
struct node {
        int data;
        struct node *left;
        struct node *right;
  };
```

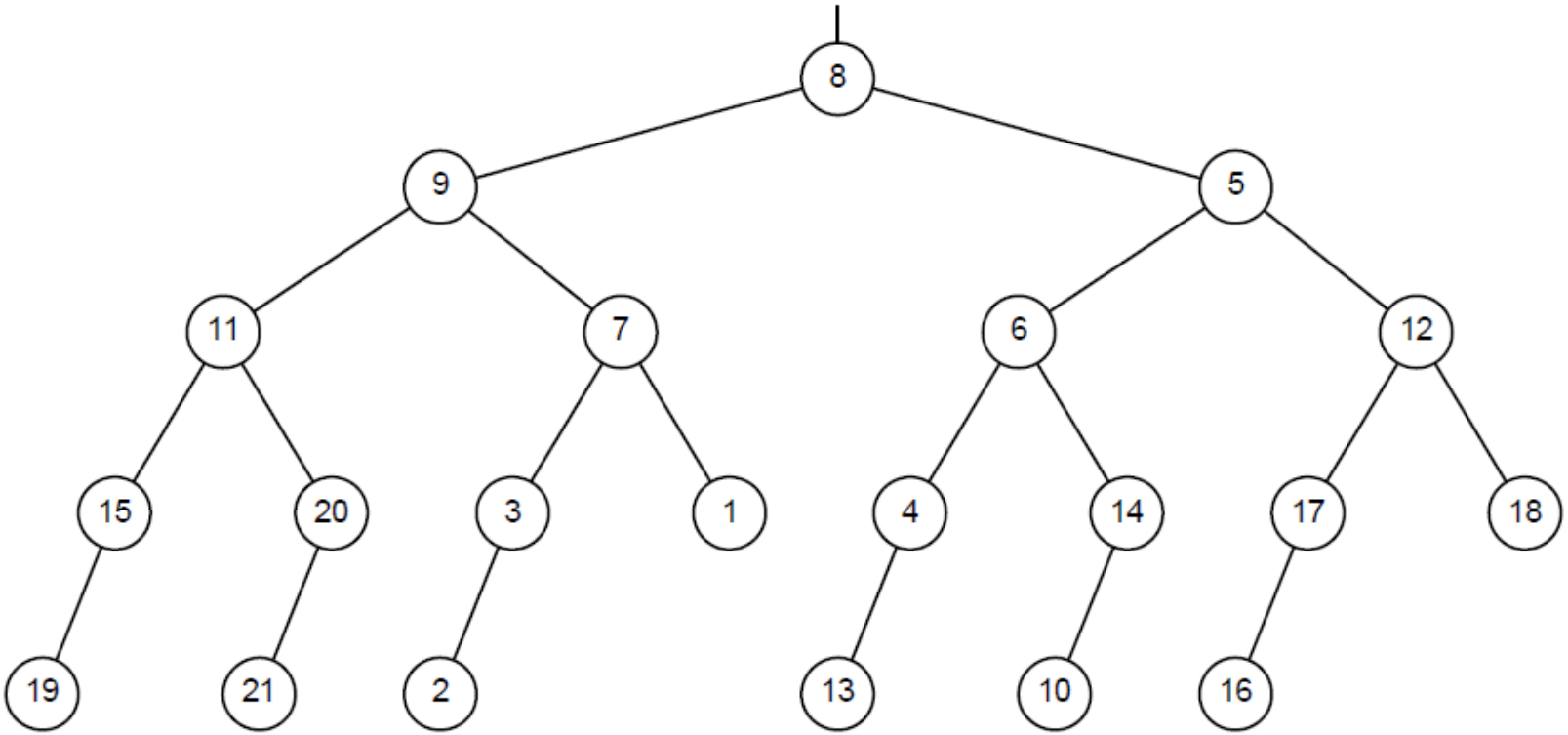
```
class node {
        private: int data;
                 node *left;
                 node *right;
        public:
                  node(int skaicius)
                  { data = skaicius;
                    left = right = NULL; }      };
```

Medžio pildymas

```
node* Tree (int N)            // N – viršūnių skaičius
{  int x;
       if ( N == 0 )
            return (NULL);
       nL = N/2; nR = N - nL -1;
           x = read();
           node* Node= new(node);
           (*Node).data = x;
           (*Node).left = Tree(nL);
           (*Node).right = Tree(nR);
  return (Node);
}
```

Pavyzdys

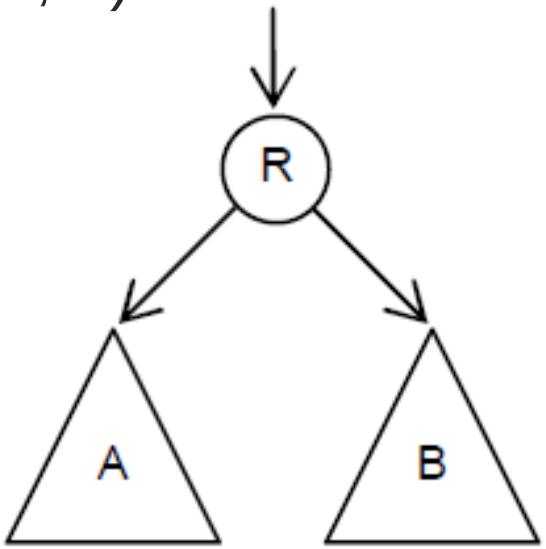
Iš 21 skaičiaus aibės suformuojamas dvejetainis medis :
(8 9 11 15 19 20 21 7 3 2 1 5 6 4 13 14 10 12 17 16 18)



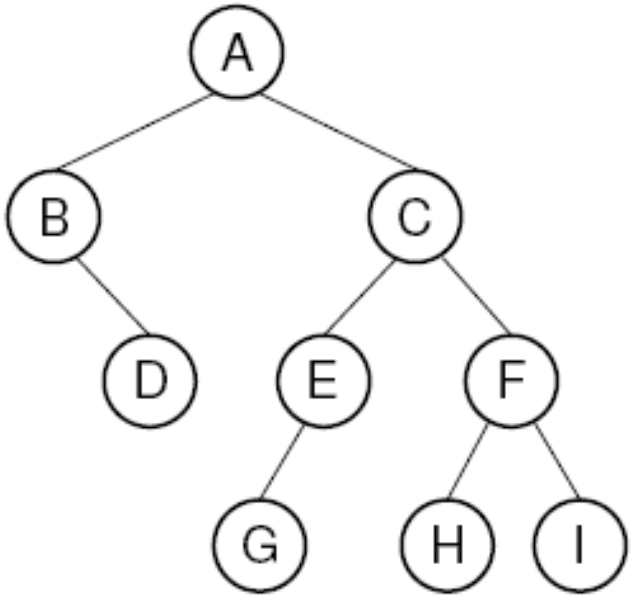
Medžio apėjimo būdai

Nuskaitant medžio struktūros viršūnės, reikia tai padaryti tik po vieną kartą. Viršūnės ir dviejų jos vaikų apėjimo tvarka gali būti:

- Tiesioginė (R, A, B)
- Atvirkštinė (A, B, R)
- Vidinė (A, R, B)



Pavyzdys

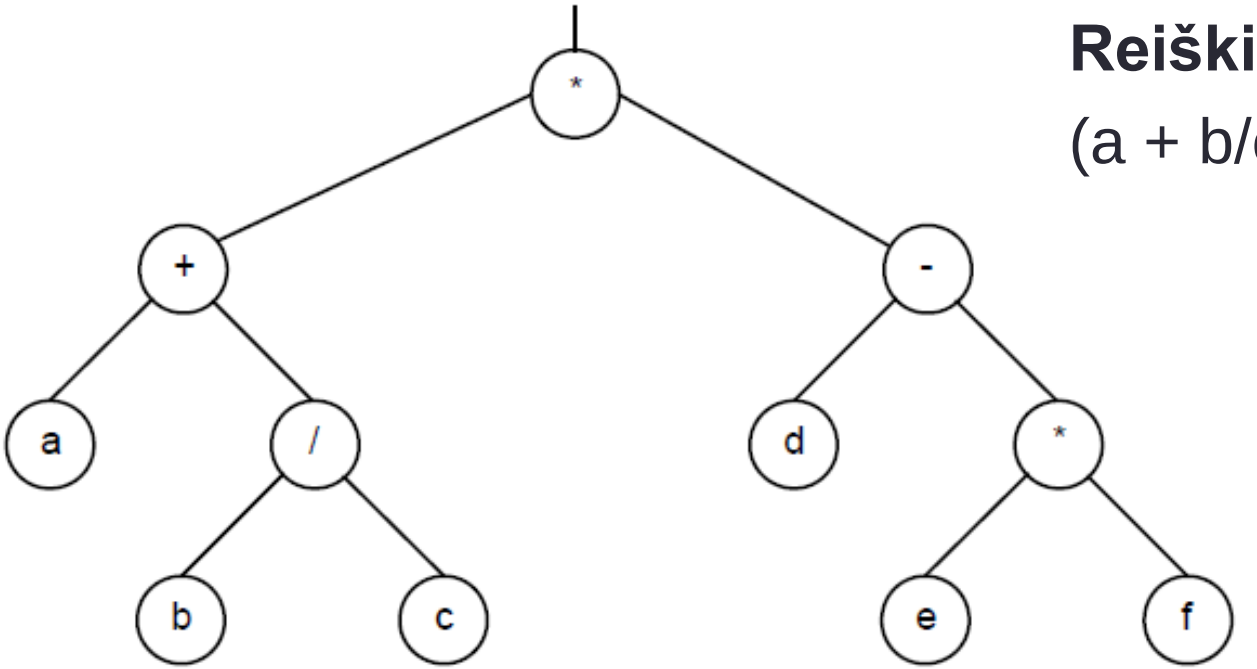


Tiesioginis apėjimas: **A B D C E G F H I**

Atvirkšinis apėjimas: **D B G E H I F C A**

Vidinis apėjimas: **D B A G E C H F I**

Pavyzdys



Reiškinys:

(a + b/c) * (d – e*f)

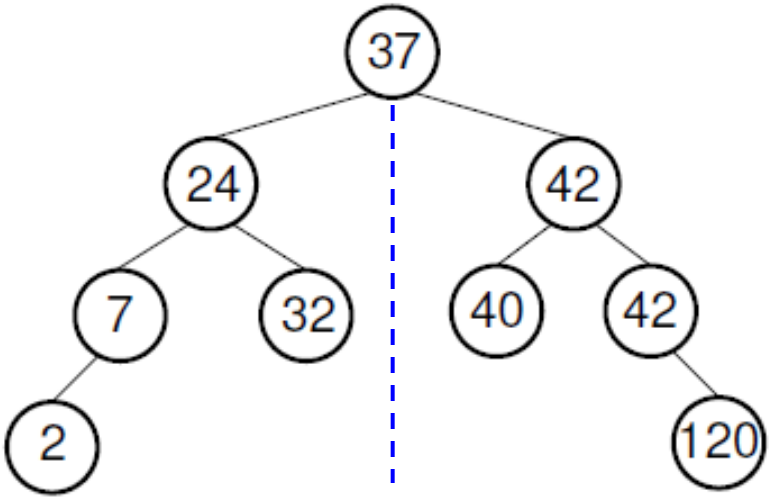
1. Tiesioginis: *** + a / b c - d * e f**
2. Vidinis: **a + b / c * d - e * f**
3. Atvirkštinis: **a b c / + d e f * - ***

Dvejetainis paieškos medis

Dvejetainis paieškos medis tenkina tokias sąlygas:

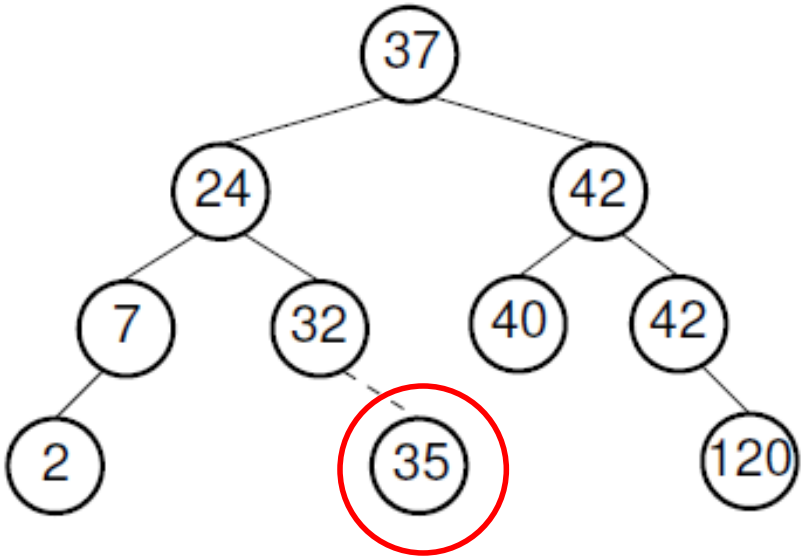
- Visos viršūnės esančios kairėje šakoje nuo šaknies turi mažesnes reikšmes nei šaknis.
- Visos viršūnės esančios dešinėje šakoje nuo šaknies turi didesnes arba lygias reikšmes šakniai.

Visos reikšmės mažesnės nei 37



Visos reikšmės didesnės nei 37

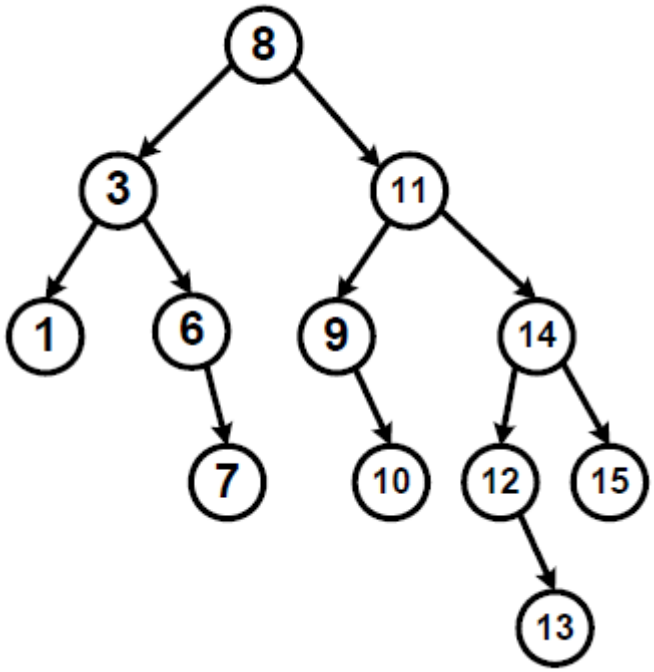
Dvejetainis paieškos medis



Įterpiama nauja viršūnė su reikšme 35.

Dvejetainio paieškos medžio formavimas

Pradiniai duomenys: 8, 11, 9, 3, 1, 14, 6, 12, 10, 7, 13, 15



Formavimo principas:

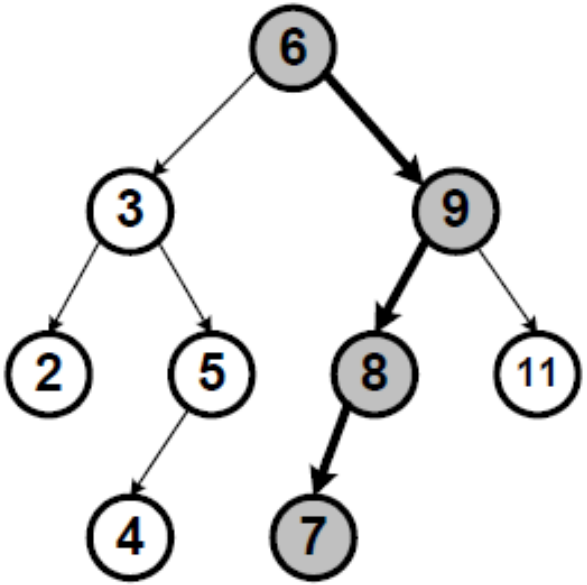
1. Pirmas elementas p_1 – šaknis.

2. Kitų elementų viršūnės šaka pasirenkama pagal nelygybės $p_i < p_k$ atsakymą:
   - teisybė – kairė šaka,
   - neteisybė – dešinė šaka.

Lyginimas pradedamas nuo šaknies.

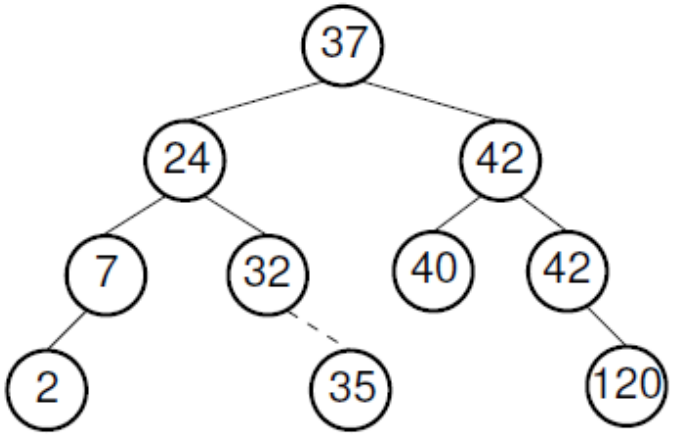
Viršūnės paieška

```
node* find_node (node* tree, int find_data)
{
     while ( (tree != NULL) && ((*tree).data != find_data) )
    {
            if ( (*tree).data < find_data )
                  tree = (*tree).right;
            else
                  tree = (*tree).left;
     }
   return tree;
}
```

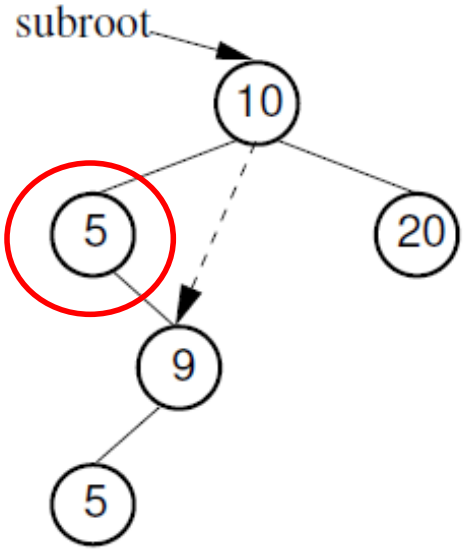


Pridedama nauja viršūnė

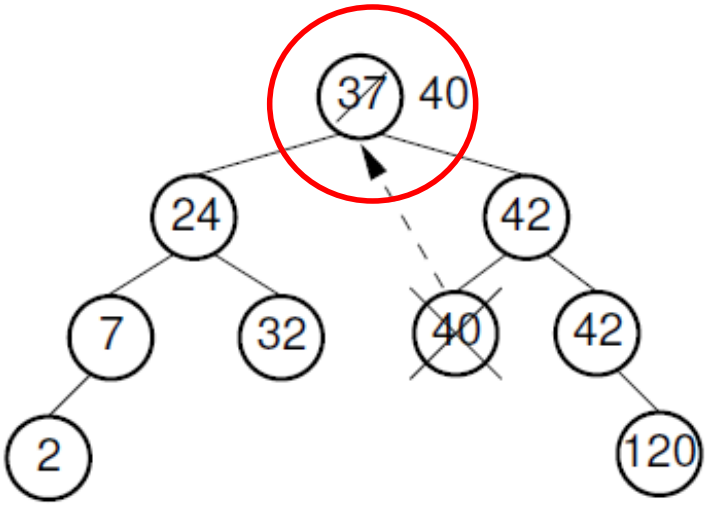
```
insert_node (node* tree, node* v)
{
        while ( tree != NULL )
        {
                if ( (*v).data < (*tree).data )
                        tree = (*tree).left;
                else
                        tree = (*tree).right;
        }
  tree = v;
  (*v).left = NULL;
  (*v).right = NULL;
}
```



Viršūnės šalinimas



Jei trinama šakos viršūnė, ji keičiama į tos viršūnės dešinės šakos vaiką.



Jei trinama šaknis, jos reikšmė keičiama į dešinės šakos mažiausią reikšmę turinčią viršūnę.

Viršūnės šalinimas

```
deleteNode (node* v, int  a) {
node* p, q;
v = find (v, a);
if ( v != NULL ) {                          // viršūnė nerasta
    q = v;
if ( (*v).left == NULL )  {                 // ne daugiau nei vienas vaikas
    v = (*v).right;
}  }
else
    if ( (*v).right == NULL ) {             // tik kairės šakos vaikas
       v = (*v).left;
}
 else {                                     // jei v turi du vaikus, ieškoma kita viršūnė
     p = (*v).right;
       while ( (*p).left != NULL )
             p = (*p).left;
             v = p;
             p = (*p).right;    (*v).left = (*q).left;
            (*v).right = (*q).right;   }   delete q; }
```