Bináris fa adatszerkezet

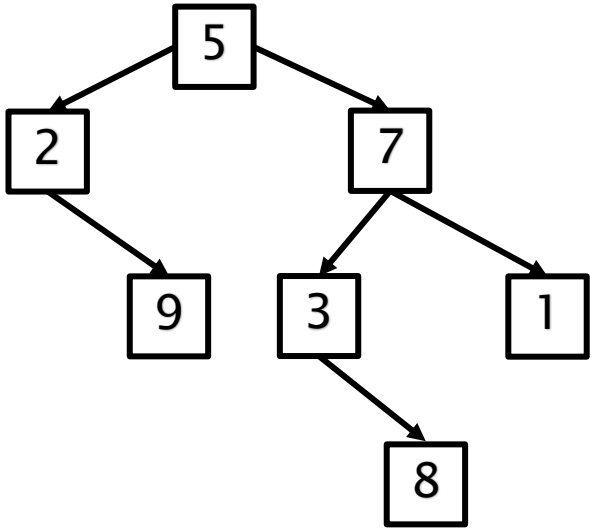
Dr. Szeghalmy Szilvia
Debreceni Egyetem, Informatikai Kar

A bináris fa

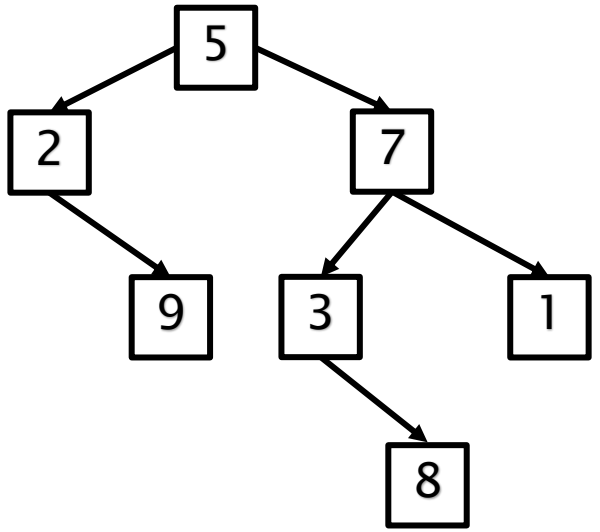
- Fa: Összefüggő, kör- és hurokmentes, irányított gráf
  - Van egy kitüntetett elem, melynek nincs megelőző (szülő) eleme: gyökér elem
  - Minden más elemnek pontosan egy szülő eleme van
  - Bináris => minden elemnek legfeljebb két leszármazottja (gyerekeleme) **lehet**

- Tulajdonságok
  - Homogén
  - **Hierarchikus**
  - Dinamikus

- Reprezentáció: **szétszórt** / folytonos



Előadáson áttekintett fogalmak



Az ábra alapján adjuk meg a válaszokat:

- gyökérelem: 5
- levélelemek: 9, 8, 1
- közbenső elemek: 2, 7, 3

- 5-ös elem bal oldali részfájának elemei: 2, 9
- 2. szinten lévő elemek (a gyökér a 0.): 9, 3, 1
- a fa magassága: 4
- a leghosszabb út a fában: 5, 7, 3, 8
- mely elemet, hova kellene tenni, hogy a fa minimális magasságú legyen?

  a 8-as elemet a 2-es baljára
- milyen sorrendben érinti az elemeket a
  - preorder bejárás (gy, b, j): 5 2 9 7 3 8 1
  - inorder bejárás (b, gy, j): 2 9 5 3 8 7 1
  - postorder bejárás (b, j, gy): 9 2 8 3 1 7 5

Művelet: Létrehozás

- Összetett elem:
  - adatrész: problémafüggő, az adatrész is lehet összetett
  - a gyerekelemek címe vagy annak hiányában None érték
- Üres fát hozunk létre és bővítjük

```
class Faelem:
    def __init__(self, adat):
        self.adat = adat      # az adat rész típusa a problémától függ
        self.bal = None       # a bal oldali gyerek címe (vagy None)
        self.jobb = None

# üres fa létrehozása
gy = None
```

Műveletek: elérés

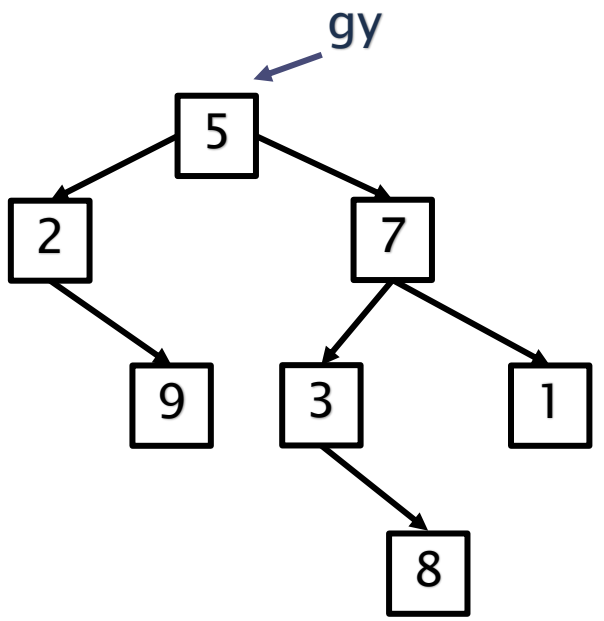
```
# LÉTREHOZÁS:

class Faelem:
   def __init__(self,adat):
       self.adat = adat
       self.bal = None
       self.jobb = None

gy = None
```

- **Elérés**: a gyökértől az elemig vezető út megadásával

- Pl.:

elem	**elérés**
5:	gy
2:	gy.bal
9:	gy.bal.jobb
7:	gy.jobb
3:	gy.jobb.bal
8:	gy.jobb.bal.jobb
1:	gy.jobb.jobb

Ha az adatrészre van szükség: gy.**adat**, gy.bal.**adat, …**

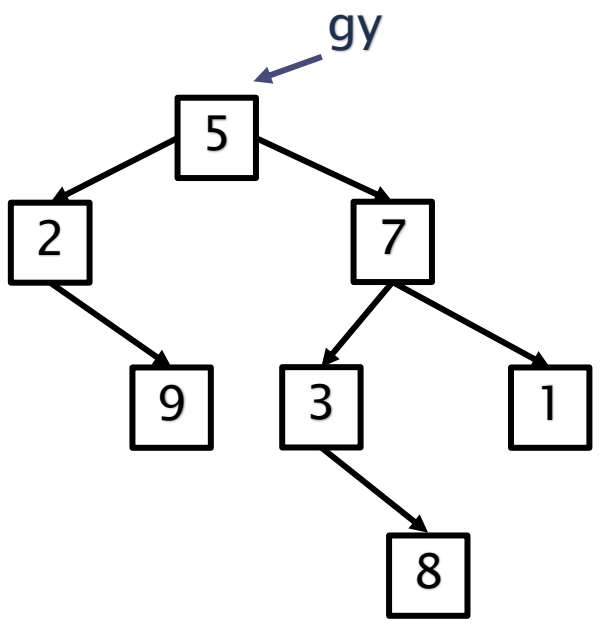


Műveletek: bejárás

- **Preorder: írjuk ki az elemeket preorder sorrendben.**

```
def preorder(gy):
    if not gy:
        return
    print(gy.adat)
    preorder(gy.bal)
    preorder(gy.jobb)

def preorder2(gy):
    if gy:
        print(gy.adat)
        preorder2(gy.bal)
        preorder2(gy.jobb)
```

```
# LÉTREHOZÁS:

class Faelem:
    def __init__(self,adat):
        self.adat = adat
        self.bal = None
        self.jobb = None

gy = None
```

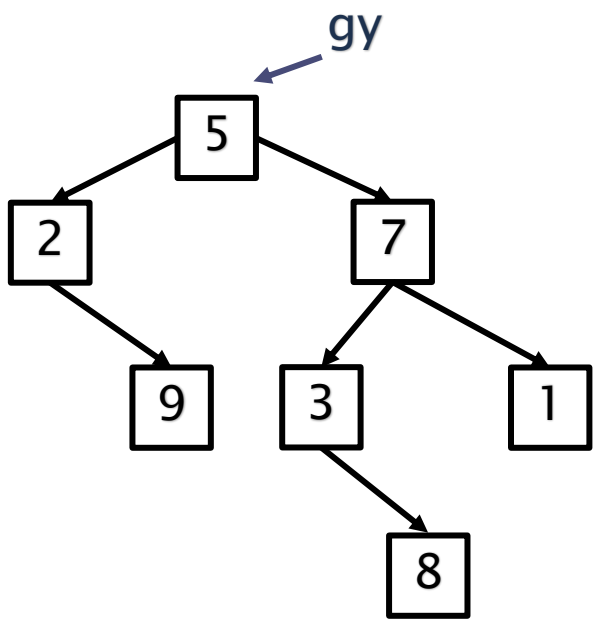


Műveletek: bejárás

- **Inorder: írjuk ki az elemeket inorder sorrendben.**

```
def inorder(gy):
    if not gy:
        return
    inorder(gy.bal)
    print(gy.adat)
    inorder(gy.jobb)

def inorder2(gy):
    if gy:
        inorder2(gy.bal)
        print(gy.adat)
        inorder2(gy.jobb)
```

```
# LÉTREHOZÁS:

class Faelem:
    def __init__(self,adat):
        self.adat = adat
        self.bal = None
        self.jobb = None

gy = None
```

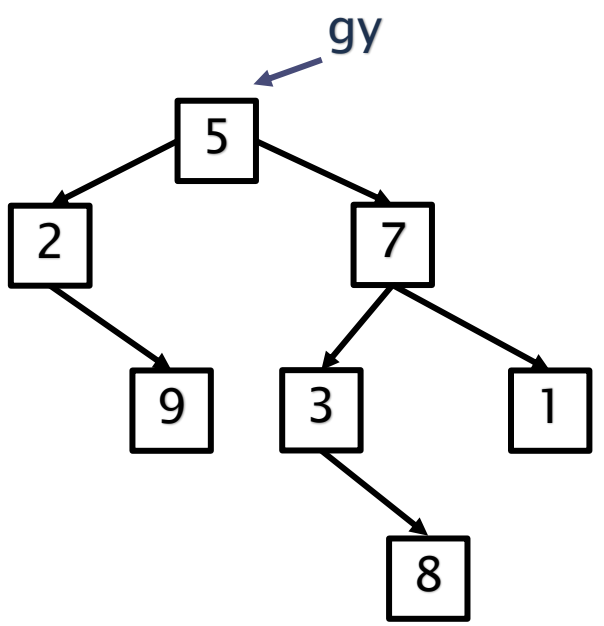


Műveletek: bejárás

- **Postorder: írjuk ki az elemeket postorder sorrendben.**

```
def postorder(gy):
    if not gy:
        return
    postorder(gy.bal)
    postorder(gy.jobb)
    print(gy.adat)

def postorder2(gy):
    if gy:
        postorder2(gy.bal)
        postorder2(gy.jobb)
        print(gy.adat)
```

```
# LÉTREHOZÁS:

class Faelem:
    def __init__(self,adat):
        self.adat = adat
        self.bal = None
        self.jobb = None

gy = None
```

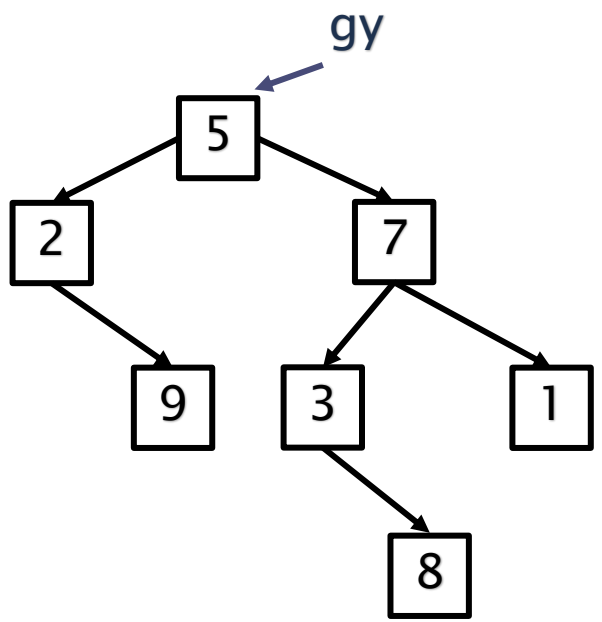


Műveletek: keresés

- Keresd meg a fában a *mit* paraméterben kapott értékű elemet és térj vele vissza. Ha nincs ilyen elem, None értékkel térj vissza.

```
def keres(gy, mit):  # gy: Faelem, mit: int
    if not gy:
        return None
    if gy.adat == mit:
        return gy
    x = keres(gy.bal, mit)
    if x:
        return x
    return keres(gy.jobb, mit)
```

```
# LÉTREHOZÁS:

class Faelem:
    def __init__(self,adat):
        self.adat = adat
        self.bal = None
        self.jobb = None

gy = None
```



Műveletek: csere

```
# LÉTREHOZÁS:

class Faelem:
   def __init__(self,orsz):
       self.orszag = orsz
       self.eset = 0
       self.bal = None
       self.jobb = None

gy = None
```

- Adatrész felülírása

Egy fában a koronavírusos esetek számát tároljuk országonként. Írj függvényt, mely a paraméterben kapott ország (orsz) esetszámát megnöveli a szintén paraméterben kapott értékkel (uj_eset).

A megoldáshoz használd a *keres(gy, orsz)* függvényt, mely visszaadja az adott országot tartalmazó elemet, vagy annak hiányában None értéket.

```
def modosit(gy, orsz, uj_eset):   #gy: Faelem, orsz: str, uj_eset: int
    e = keres(gy, orsz)
    if e:
        e.eset = uj_eset
```

Műveletek: csere

```
# LÉTREHOZÁS:

class Faelem:
   def __init__(self,orsz):
       self.orszag = orsz
       self.eset = 0
       self.bal = None
       self.jobb = None

gy = None
```

Egy fában a koronavírusos esetek számát tároljuk országonként.

A tudósok becslése szerint tízszer annyi megbetegedés van, mit amennyit a tesztekkel kimutatnak.

Írj függvényt, mely ennek megfelelően minden esetszámot felszoroz 10-zel.

```
def becsult_esetszam(gy):
    if not gy:
       return
    gy.adat *= 10
    becsult_esetszam(gy.bal)
    becsult_esetszam(gy.jobb)
```

```
def becsult_esetszam2(gy):
    if gy:
        gy.adat *= 10
        becsult_esetszam2(gy.bal)
        becsult_esetszam2(gy.jobb)
```

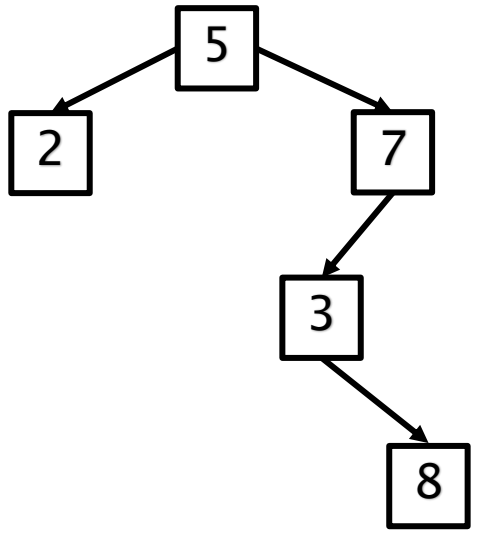
Műveletek: bővítés / felépítés

```
# LÉTREHOZÁS:

class Faelem:
   def __init__(self,adat):
       self.adat = adat
       self.bal = None
       self.jobb = None

gy = None
```

- Az új elemeket levélelemként szúrjuk be a fába.

Mielőtt függvényt írnánk rá, építsük fel az ábrán látható fát az elérés műveletre alapozva.

```
gy = Faelem(5)
gy.bal = Faelem(2)
gy.jobb = Faelem(7)
gy.jobb.bal = Faelem(3)
gy.jobb.bal.jobb = Faelem(8)
```



Műveletek: bővítés / felépítés

Írjunk függvényt, mely a paraméterben kapott adatot beszúrja a fába.

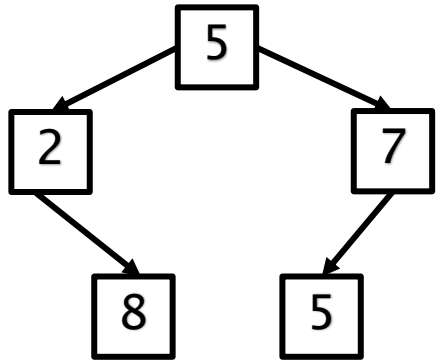
A függvény véletlenszerűen döntse el, hogy mely részfába kerüljön az elem.

A döntést *a random.randint(0,1)* függvényhívás eredménye alapján hozd meg.

(Megj.: a véletlen segít minket abban, hogy az elemek viszonylag egyenletesen oszoljanak el a részfák között)

```
# LÉTREHOZÁS:

class Faelem:
   def __init__(self,adat):
       self.adat = adat
       self.bal = None
       self.jobb = None

gy = None
```

```
import random
def beszur(gy, adat): # gy: Faelem; mit: int
   if not gy:
      return Faelem(adat)
   if random.randint(0,1):
      gy.bal = beszur(gy.bal, adat)
   else:
      gy.jobb = beszur(gy.jobb, adat)
   return gy
```

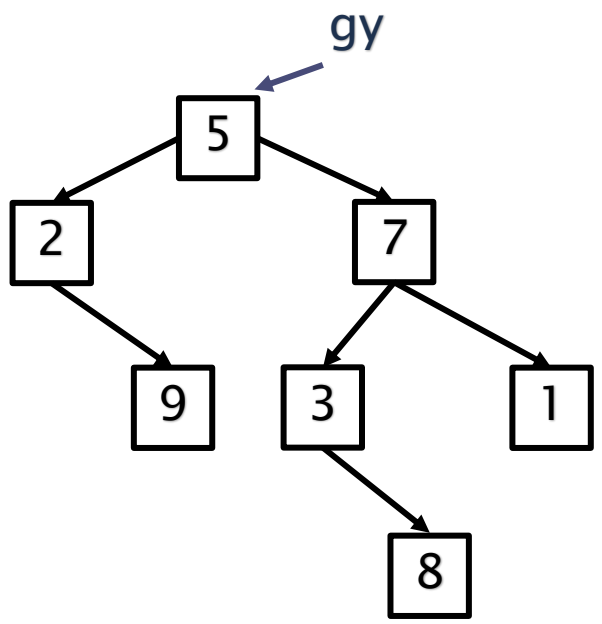


Műveletek: felszabadítás

```
# LÉTREHOZÁS:

class Faelem:
   def __init__(self,adat):
       self.adat = adat
       self.bal = None
       self.jobb = None

gy = None
```

- Postorder sorrendben töröljük az összes elemet a fából.
- *Megj.: A Python automatikusan eltávolítja a memóriából azon elemeket, melyek már nem érhetőek el, de lássuk az algoritmust.*

```
def felszabadit(gy):  # Faelem
    if not gy:
        return None
    gy.bal = felszabadit(gy.bal)
    gy.jobb = felszabadit(gy.jobb)
    del gy #felszabadítjuk a memóriaterületet amely nyelvben szükséges
    return None
```



Töröld az ábrán látott fa jobb oldali részfáját a *felszabadit(gy)* függvény segítségével.

```
gy.jobb = felszabadit(gy.jobb)
```

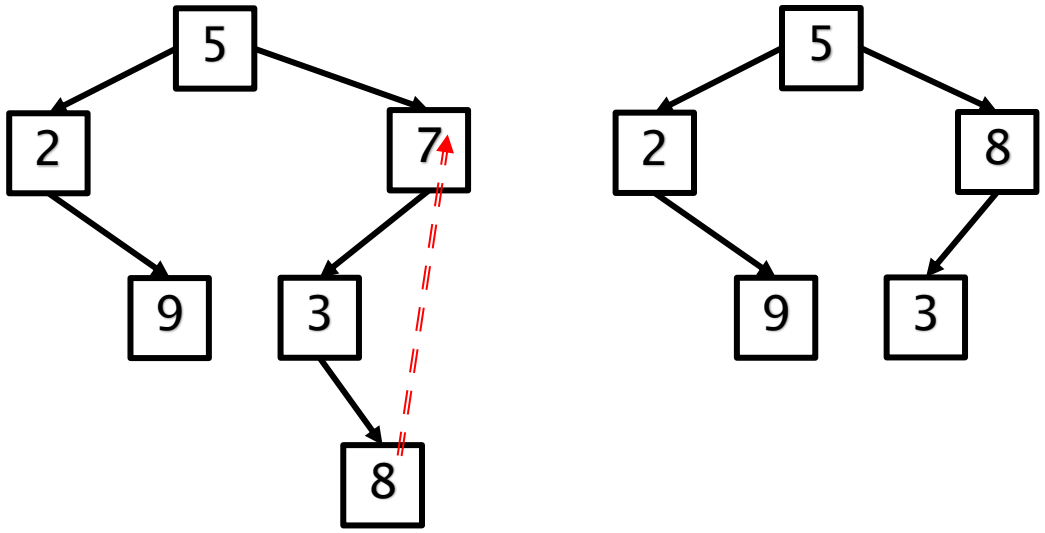
Műveletek: törlés (részletek később)

```
# LÉTREHOZÁS:

class Faelem:
   def __init__(self,adat):
       self.adat = adat
       self.bal = None
       self.jobb = None

gy = None
```

- Csak levélelemet törlünk ki a fából
- Mit tegyünk, ha a törlendő nem levél?
  - Keresünk egy levélelemet
  - Felülírjuk vele a törlendőt
  - Töröljük a levélelemet

Pl: töröljük a 7-est

Levélelemek: 9, 8 a 8-ast választva:




```
def levelkeres(gy):
    if not gy:
        return None
    if (gy.bal is None) and (gy.jobb is None):
        return gy
    lvl = levelkeres(gy.bal)
    if not lvl:
       lvl = levelkeres(gy.jobb)
   return lvl
```

Feladatok

- Írj függvényt, mely **megjeleníti** hány %-a gyógyult meg a betegeknek az egyes országokban (az ország kulcs, tehát egyedi)! A fában csak azon országok szerepelnek, ahonnan jelentettek eseteket.

```
# LÉTREHOZÁS:

class Faelem:
   def __init__(self,orsz):
       self.orszag = orsz
       self.eset = 0
       self.gyogyult = 0
       self.bal = None
       self.jobb = None

gy = None
```

```
def gyogyult(gy):
    if not gy:
       return
    print(f"{gy.orszag:20}{gy.gyogyult/gy.eset:0.2f}")
    gyogyult(gy.bal)
    gyogyult(gy.jobb)
```

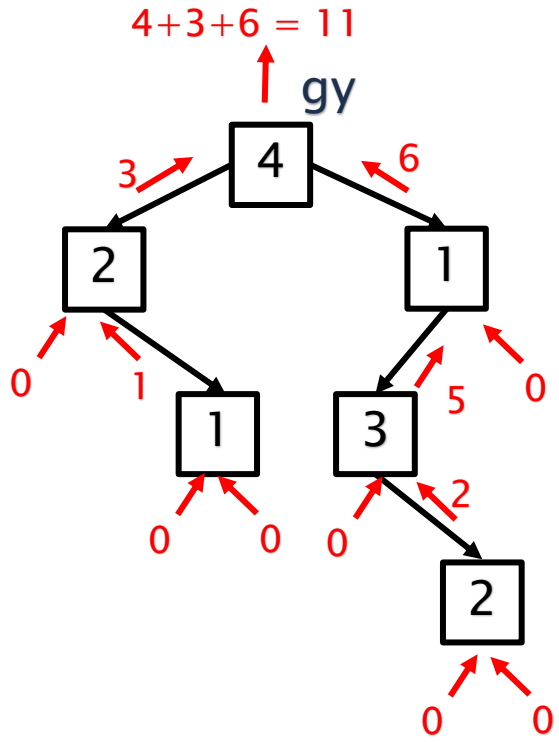
```
def gyogyult2(gy):
    if gy:
        print(f"{gy.orszag:20}{gy.gyogyult/gy.eset:0.2f}")
        gyogyult2(gy.bal)
        gyogyult2(gy.jobb)
```

Feladatok

- Írj függvényt, mely meghatározza hogy hány koronavírusos eset van összesen az országokban!
  - ha nincs (rész)fa, 0
  - tudd meg a bal oldali részfára az esetszámot
  - tudd meg a jobb oldali részféra az esetszámot
  - add vissza a kettő összegét + a gyökérben lévő esetszámot

```
def osszes_eset(gy):
    if not gy:
       return 0
    ossz = gy.adat
    b_ossz = osszes_eset(gy.bal)
    j_ossz = osszes_eset(gy.jobb)
    return ossz + b_ossz + j_ossz
```

```
# LÉTREHOZÁS:

class Faelem:
   def __init__(self,orsz):
       self.orszag = orsz
       self.eset = 0
       self.bal = None
       self.jobb = None

gy = None
```

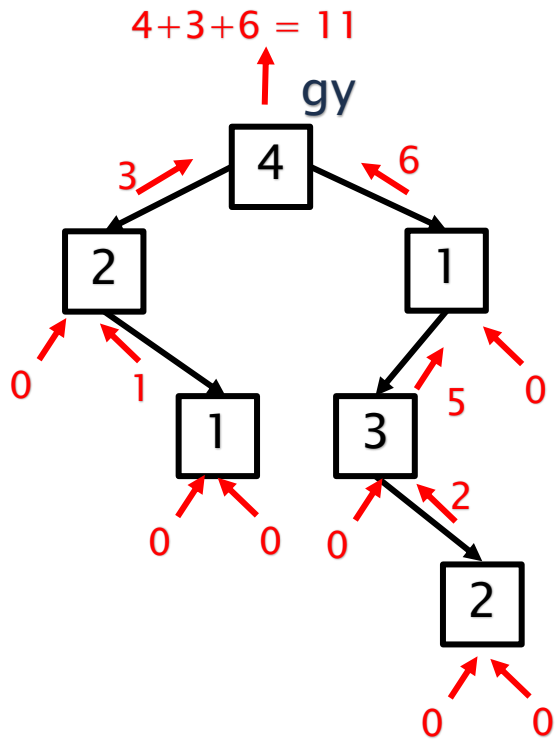


Feladatok

- Írj függvényt, mely meghatározza hogy hány koronavírusos eset van összesen az országokban!
  - ha nincs (rész)fa, 0
  - tudd meg a bal oldali részfára az esetszámot
  - tudd meg a jobb oldali részféra az esetszámot
  - add vissza a kettő összegét + a gyökérben lévő esetszámot

```
def osszes_eset(gy):
    if not gy:
       return 0
    return gy.adat + osszes_eset(gy.bal) + osszes_eset(gy.jobb)
```

```
# LÉTREHOZÁS:

class Faelem:
   def __init__(self,orsz):
       self.orszag = orsz
       self.eset = 0
       self.bal = None
       self.jobb = None

gy = None
```

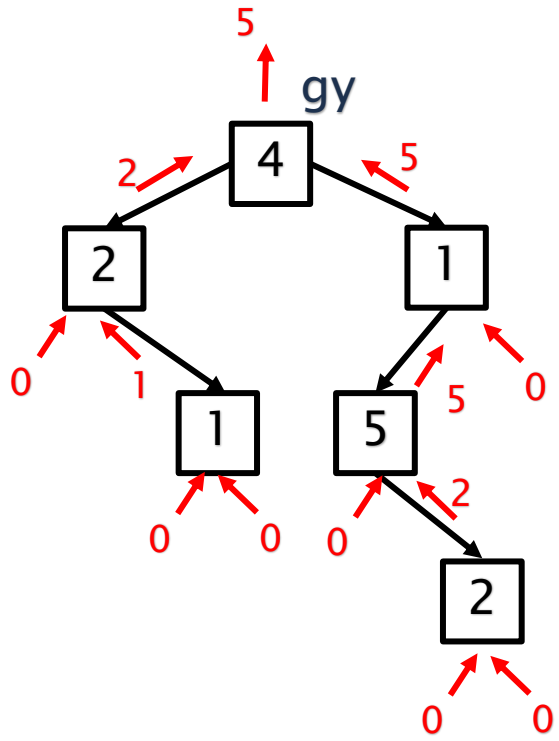


Feladatok

- Írj függvényt, mely meghatározza az esetszám maximumát!
  - ha nincs (rész)fa, 0
  - tudd meg a bal oldali részfára az esetszám maximumát
  - tudd meg a jobb oldali részféra az esetszám maximumát
  - add vissza a gyökérben lévő és a részfák maximumát

```
def max_eset(gy):
    if not gy:
        return 0
    max = gy.adat
    b_max = max_eset(gy.bal)
    j_max = max_eset(gy.jobb)
    if b_max > max:
        max = b_max
    if j_max > max:
        max = j_max
    return max
```

```
# LÉTREHOZÁS:

class Faelem:
    def __init__(self,orsz):
        self.orszag = orsz
        self.eset = 0
        self.bal = None
        self.jobb = None

gy = None
```



Feladatok

- Írj függvényt, mely meghatározza hogy hány országban van 1000 felett az esetszám!

```
# LÉTREHOZÁS:

class Faelem:
   def __init__(self,orsz):
       self.orszag = orsz
       self.eset = 0
       self.bal = None
       self.jobb = None

gy = None
```

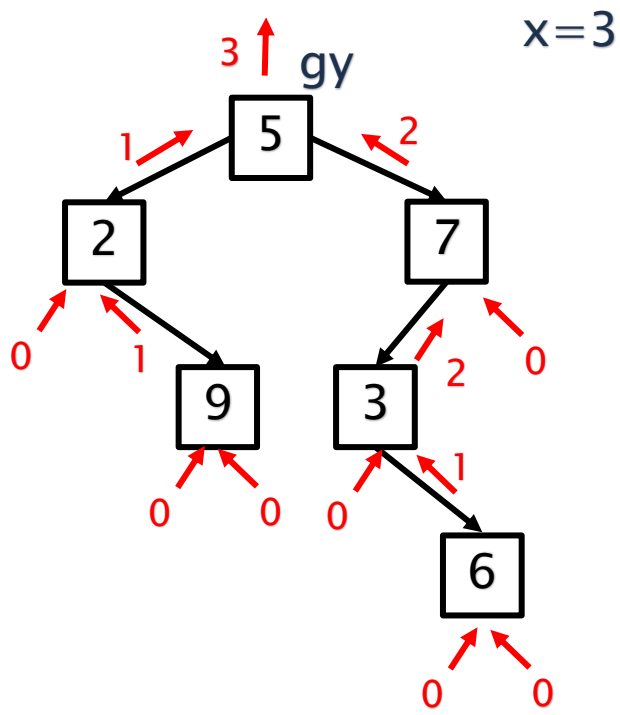
```
def sokk(gy):
    if not gy:
       return 0
    db = 0
    if gy.eset > 1000:
       db = 1
    b_db = sokk(gy.bal)
    j_db = sokk(gy.jobb)
    return db + b_db + j_db


def sokk2(gy):
    if not gy:
       return 0
    return (1 if gy.eset > 1000 else 0) + sokk2(gy.bal) + sokk2(gy.jobb)
```

Feladatok

- Számold meg, hogy hány olyan érték van a fában, mely osztható az *x* paraméterben kapott értékkel (*x* != 0)

```
def oszthato(gy, x):
   if not gy:
        return 0   # a feladat által kért visszatérési érték
   # vizsgáljuk meg az adott csúcsot
   db = 0
   if gy.adat % x == 0:
       db = 1
   # és kérdezzük le, hogy a részfákban hány ilyen elem van
   db += oszthato(gy.bal, x)
   db += oszthato(gy.jobb, x)
   return db
```

```
# LÉTREHOZÁS:

class Faelem:
   def __init__(self,adat):
       self.adat = adat
       self.bal = None
       self.jobb = None

gy = None
```

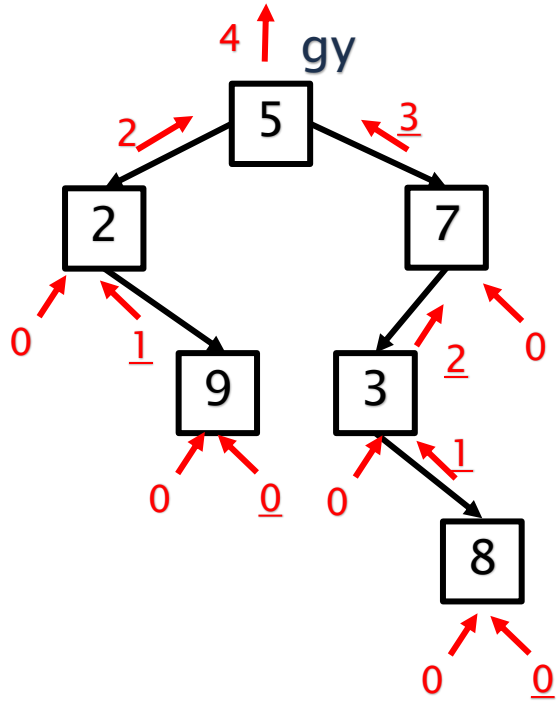


Feladatok

- Írj függvényt, mely meghatározza egy fa magasságát.
  - ha nincs (rész)fa, 0 a magasság
  - tudd meg a bal oldali részfa magasságát
  - tudd meg a jobb oldali részfa magasságát
  - add vissza a nagyobbat +1-et (mert gyökér is egy szint)

```
def magassag(gy):
    if not gy:
        return 0
    b = magassag(gy.bal)
    j = magassag(gy.jobb)
    if b > j:
        return b+1
    return j+1
```

```
# LÉTREHOZÁS:

class Faelem:
   def __init__(self,adat):
       self.adat = adat
       self.bal = None
       self.jobb = None

gy = None
```



Feladatok

- Írj függvényt, mely megadja mely országban van a legtöbb koronavírusos eset.
  A függvény értékpárt adjon vissza: ország, eset
- Üresfára visszaadandó érték: "", 0

```
# LÉTREHOZÁS:

class Faelem:
   def __init__(self,orsz):
       self.orszag = orsz
       self.eset = 0
       self.bal = None
       self.jobb = None

gy = None
```

```
def fertozott(gy):
    if not gy:
        return "", 0   # a feladat által kért visszatérési érték
    orsz, max = gy.orszag, gy.eset
    b_orsz, b_max = fertozott(gy.bal)
    j_orsz, j_max = fertozott(gy.jobb)
    if b_max > max:
       orsz, max = b_orsz, b_max
    if j_max > max:
       orsz, max = j_orsz, j_max
    return orsz, max
```