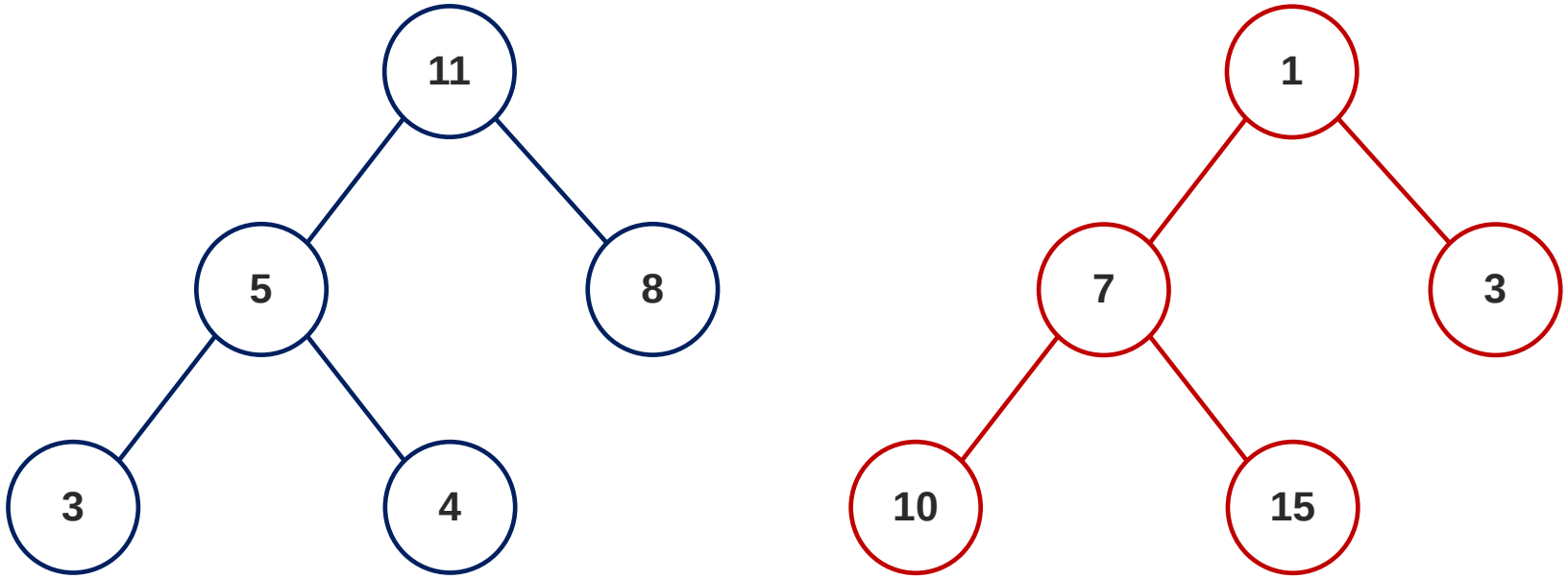
ADATSZERKEZETEK ÉS ALGORITMUSOK

Bináris Keresőfa, Bináris Kupac

Kupac

- Kupactulajdonság:
  - minden szülő kulcsa nagyobb (maximum-kupac) / kisebb (minimum-kupac), mint a gyerekei kulcsa – így a legnagyobb / legkisebb kulcsú elem a gyökér



Kupac műveletei

- Beszúrás:
  - Az újonnan beszúrt elemet a balra tömörítettség szabálya szerint a következő szabad helyre tesszük
  - Ezután ezt az elemet a megfelelő helyére juttatjuk a kupacban, így helyreállítjuk a kupactulajdonságot
- Gyökérelem törlése:
  - A gyökérelemet kivesszük a helyéről
  - A leendő gyökérelemet a helyére juttatjuk, így szintén helyreállítjuk a kupactulajdonságot
- Maximum / Minimum elem lekérdezés:
  - A gyökérelemet értékét adja vissza

Fák ábrázolása tömbben

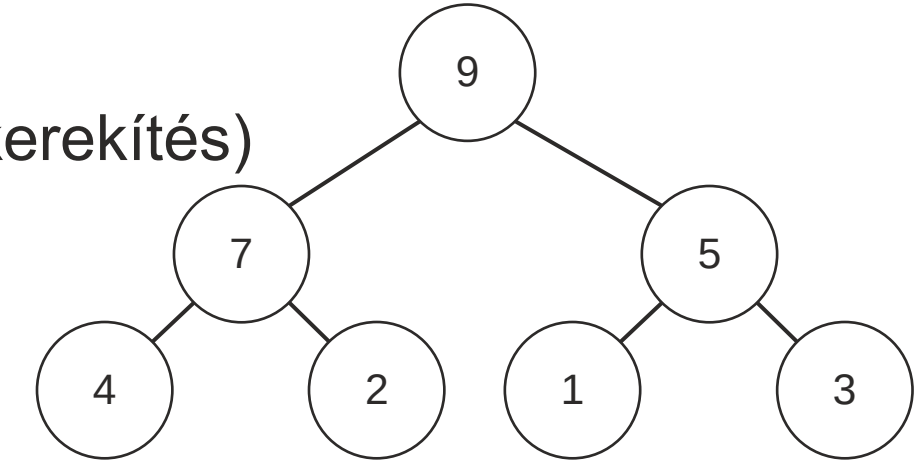
- Mivel a kupac majdnem teljes, valamint balra tömörített, ezért tömbben is lehet ábrázolni.
- A tömbben a kupac szintjei a gyökértől kezdve, egymás után jönnek, az egyes szintekben az elemek balról jobbra következnek: az első elem a gyökér, ezután jön a bal gyerek, majd a jobb gyerek, és így tovább…
- Így a reprezentáció nem lesz lyukas.
- A kupac ezen reprezentációja egyszerűbb, kisebb, ezáltal gyorsabb a gráfos ábrázolásnál, ezért ezt használják.

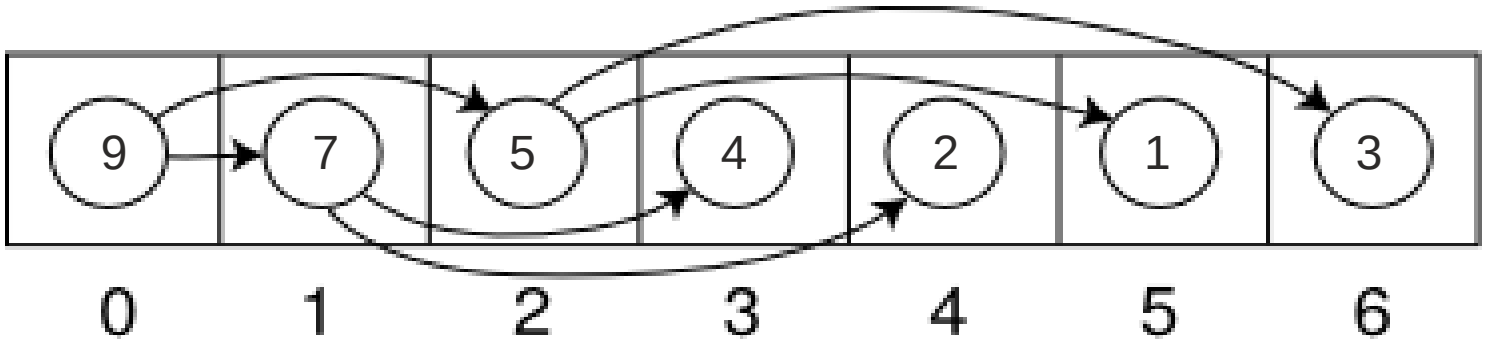
Leképezés a tömbbe

- A fa elemeit le kell képezni egy tömbbe.
  - Egyértelmű leképezésnek kell lennie, azaz oda-vissza működnie kell
- Egy indexfüggvényre van szükségünk.
  - A mátrixok sor- / oszlopfolytonos ábrázolásánál hasonló függvényeket használunk

Indexfüggvények

- Hasonlóan a tömb indexeléshez – 0-tól induló indexelés esetén
  - *i* csúcs bal gyereke: 2*i* + 1
  - *i* csúcs jobb gyereke: 2*i* + 2
  - *i* csúcs szülője: ⌊(*i* – 1) / 2 ⌋ (lefele kerekítés)





Bináris keresési fa

Bináris keresőfa műveletei

- Keresés
- Minimumkeresés
- Maximumkeresés
- Következő elem keresése
- Megelőző elem keresése
- Beszúrás
- Törlés

Bináris keresőfa műveletei I.

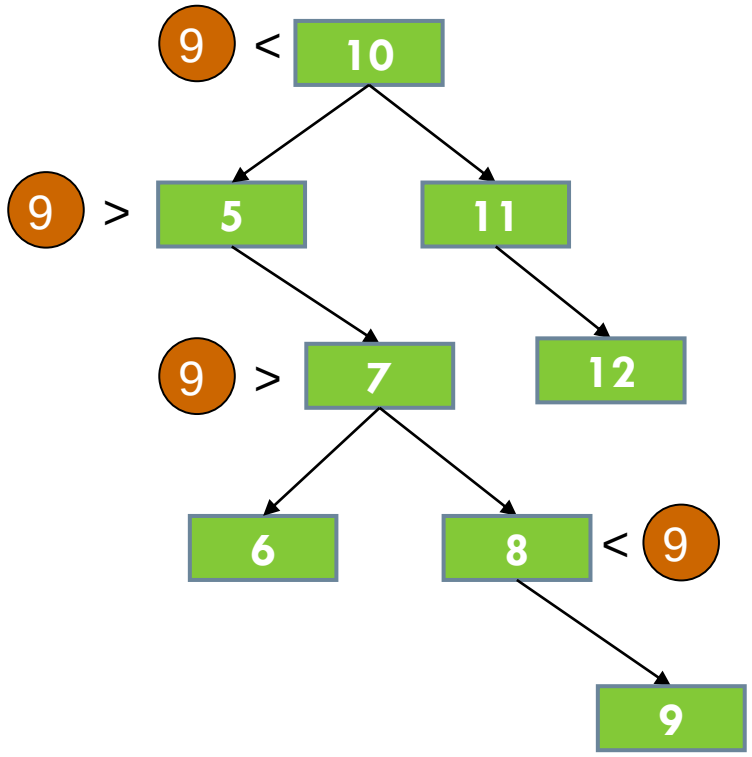
- **Beszúrás:** alapszabály:
  - nagyobb: jobbra lépünk
  - kisebb: balra lépünk
  - ha nincs gyereke ott, ahova lépnénk: beszúrunk
- **Keresés:** k kulcsú elemet keressük.
  - gyökértől indulunk
  - ha kisebb a keresett érték az aktuálisnál balra lépünk, ha nagyobb jobbra
- **Minimumkeresés részfában:**
  - addig lépünk balra, ameddig csak lehet.
  - Globális minimum: ha a gyökértől indulunk.
- **Maximumkeresés részfában:**
  - addig lépünk jobbra, ameddig csak lehet.
  - Globális maximum: ha a gyökérből indulunk.

Bináris keresőfa műveletei II.

- *Inorder bejárás esetén*
- **Következő elem:**
  - van jobb részfája: a részfa minimuma
  - nincs jobb részfája: lépjünk felfelé a fában. Az első elem aminek ő a bal részfájában van (tehát az első aminél kisebb) a rákövetkező elem.
- **Megelőző elem:**
  - van bal részfája: a részfa maximuma
  - nincs bal részfája: lépjünk felfelé a fában. Az első elem aminek ő a jobb részfájában van a keresett (megelőző értékű) elem.

Beszúrás

- A bináris keresőfába a 9 kulcsú csúcsot szúrjuk be.
- Feltételezzük (vagy ellenőrizzük), hogy a fában még nincs 9 kulcsú csúcs!
- Először megkeressük a helyét, majd beláncoljuk.



Ennek a csúcsnak már nincs jobb gyereke.
Következésképpen megtaláltuk a szülőt!

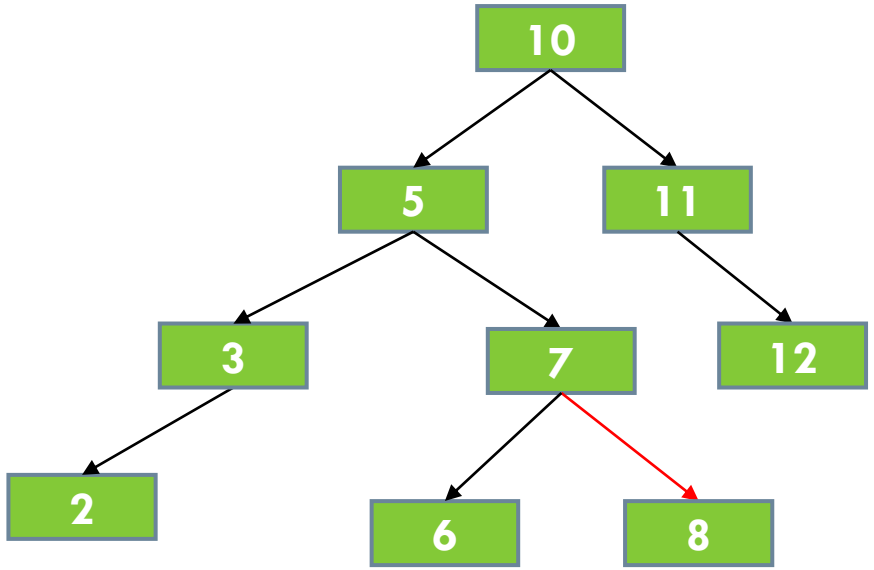
Mivel 9>8, ezért a 9 kulcsú csúcs a 8 kulcsú jobb gyereke lesz.

Törlés

- A bináris keresőfából a *p* kulcsú csúcsot töröljük.
- Lehetőségek:
  1. A *p* kulcsú csúcsnak még nincs gyereke: szülőjének mutatóját `nullptr`-re állítjuk.
  2. A *p* kulcsú csúcsnak egy gyereke van: a szülője és a gyermeke között építünk ki kapcsolatot.
  3. A *p* kulcsú csúcsnak két gyereke van: átszervezzük a fát: kivágjuk azt a legközelebbi rákövetkezőjét, aminek nincs bal gyereke, így 1., vagy 2. típusú törlés, majd ennek tartalmát beírjuk a *p* kulcs helyére.

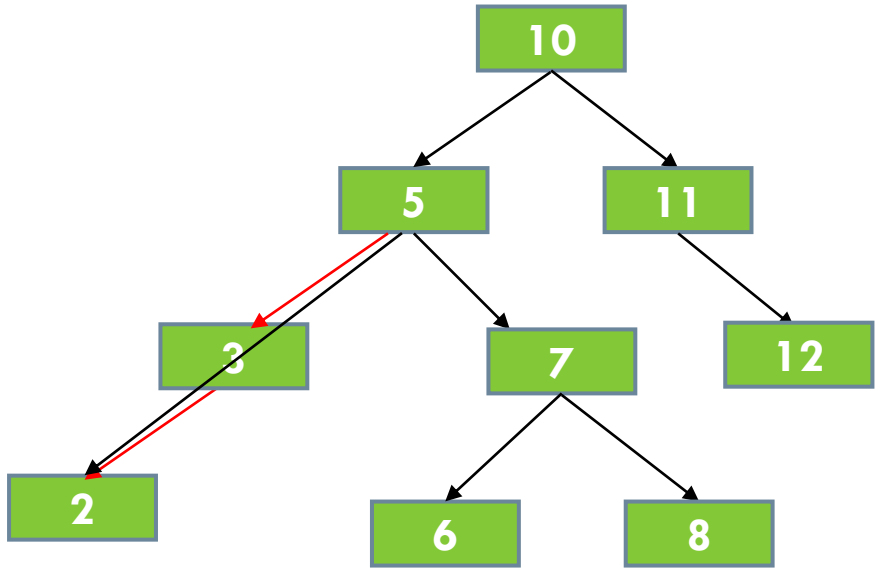
Törlés – 1. eset

- Töröljük a fából a **8** kulcsú csúcsot. Ezen csúcsnak nincs gyereke, tehát szülőjének megfelelő mutatóját `nullptr`-re állítjuk.



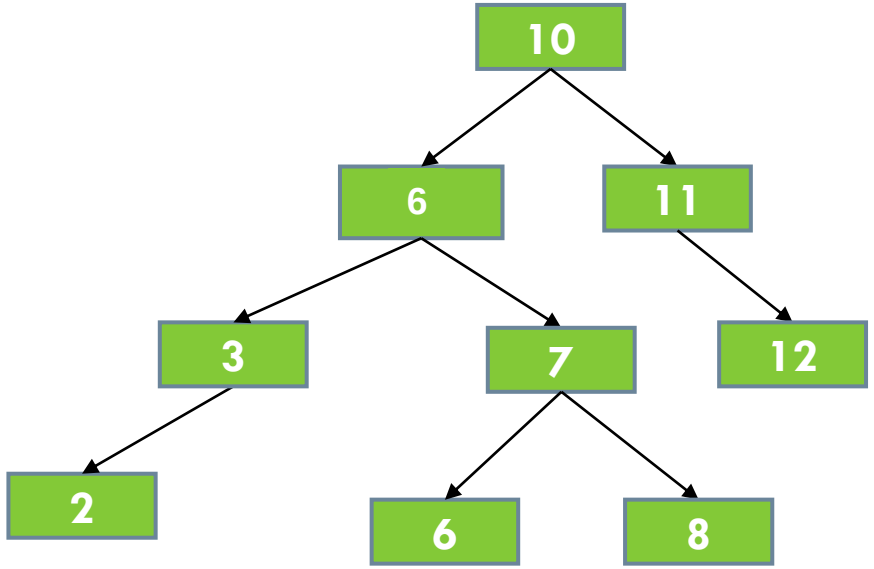
Törlés – 2. eset

- Töröljük a fából a **3** kulcsú csúcsot. Ezen csúcsnak egy gyereke van, tehát szülő és gyermeke között építünk ki kapcsolatot.



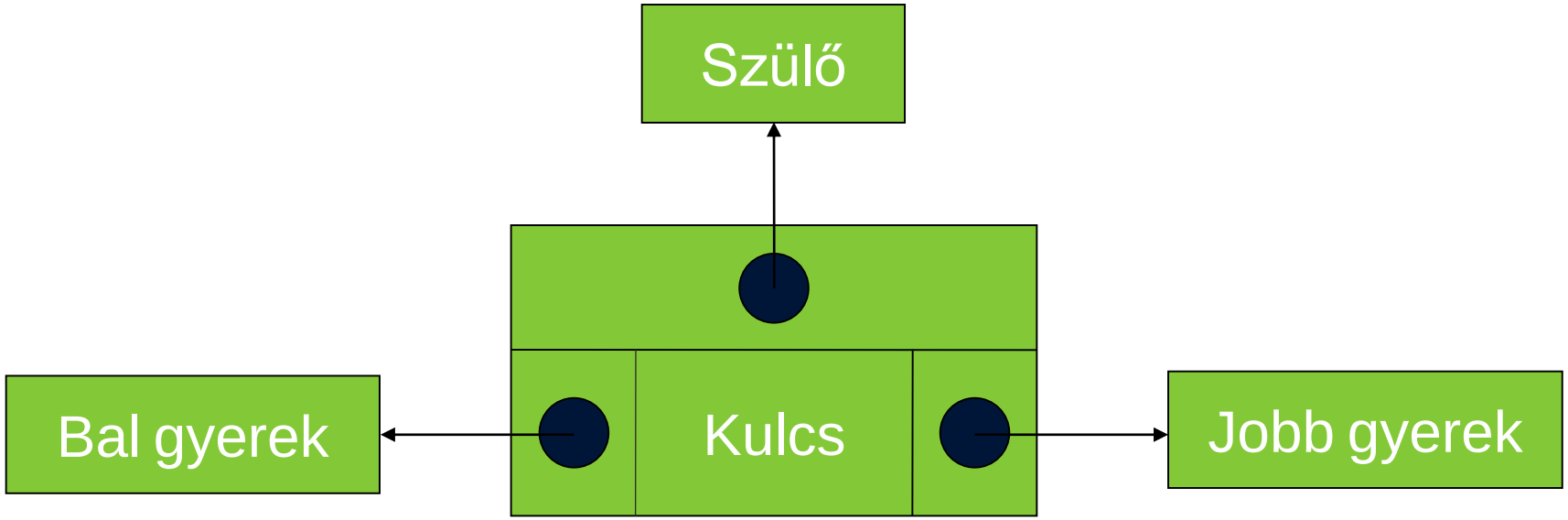
Törlés – 3. eset

- Töröljük a fából a 5 kulcsú csúcsot. Ezen csúcsnak két gyereke van, tehát megkeressük a rákövetkezőjét, ez a 6 kulcsú csúcs, és ennek nincs bal gyermeke. Így a 6 értéket beírjuk az 5 értékű csúcsba, és töröljük a 6-ost.



Reprezentáció

- A bináris keresőfák reprezentációja során is láncolt ábrázolást használunk.
- Az adatelemeket a következő módon reprezentáljuk:



C++-ban

Beágyazott elem osztály és külön kitüntetett gyökérelem:

```
template<class T>
class Bs_tree {
    // Belső csúcs struktúra
    struct Node {
        Node *parent;
        Node *left, *right;
        T key;
        Node(const T& k) : parent(nullptr), left(nullptr), right(nullptr), key(k){}
        Node(const T& k, Node *p) : parent(p), left(nullptr), right(nullptr), key(k){}
    };
    Node *root;
};
```

Fontos!
T olyan kell legyen, amire tudunk értelmezni „rendező operátort”!

Megírandó műveletek:

Amiket kívülről látunk (public):

```
size_t size();
bool isempty();
bool find(T k);
void insert(T k);
void remove(T k);
T min();
T max();
ostream& InOrder(ostream& o);
ostream& PreOrder(ostream& o);
ostream& PostOrder(ostream& o);
```

Amiket kívülről nem látunk (private):

```
Node* _next(Node* p);
Node* _prev(Node* p);
Node* _min(Node* p);
Node* _max(Node* p);
void _remove(Node* p);
size_t _size(Node *x);
ostream& _inorder(Node* i, ostream& o);
ostream& _preorder(Node* i, ostream& o);
ostream& _postorder(Node* i, ostream& o);
```

Kupac megvalósítás

Következő téma