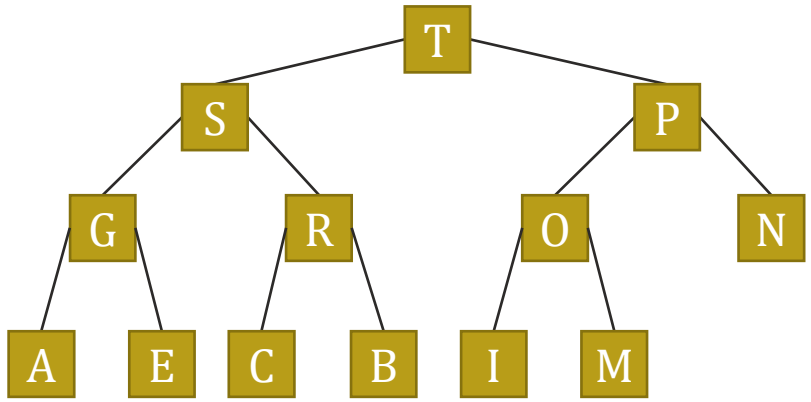
ADATSZERKEZETEK ÉS ALGORITMUSOK

Kupac, Heap

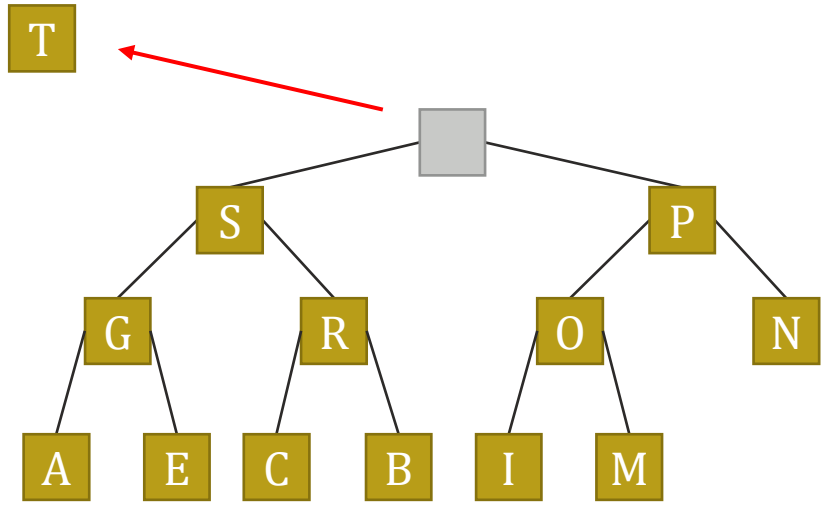
Kupac

- Egy majdnem teljes (bináris) fa heap tulajdonságú, ha
  - Üres, vagy
  - A gyökérben lévő kulcs nagyobb, mint mindkét gyerekében és mindkét részfája is heap tulajdonságú



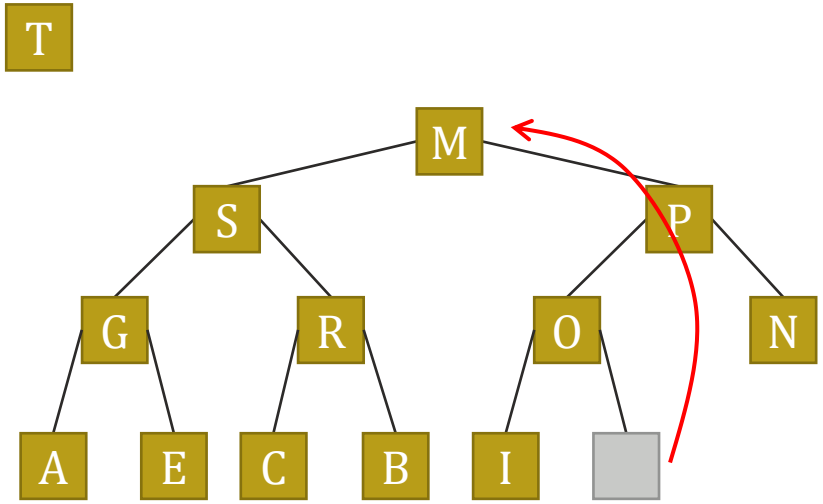
Kupac

- A kupacok használhatóak
  - A prioritásos sorok megvalósítására, mivel a gyökérben lévő elem a maximális és ez az, amit a delmax kitöröl
    - A törlés után helyre kell állítani a heap tulajdonságot
  - Rendezésre (később)



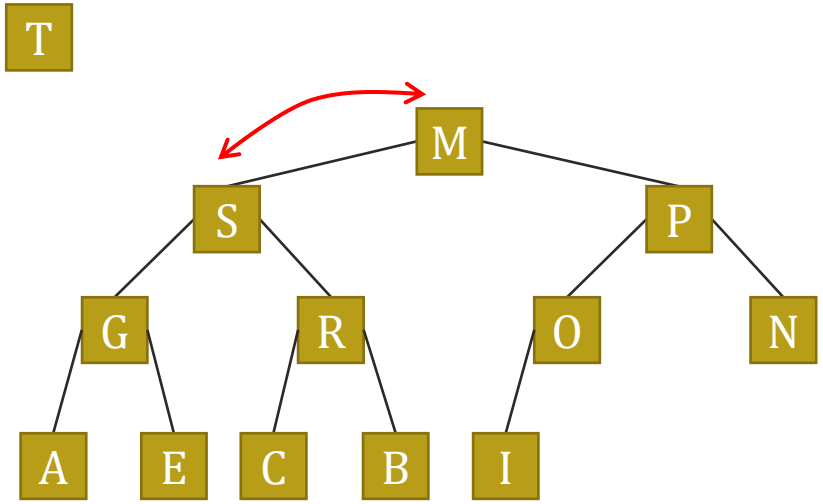
Kupac – helyreállítás

- Az első lépésében a majdnem teljes tulajdonságot állítjuk helyre
  - Vigyük fel a legutolsó elemet a gyökérbe
    - Ezzel a majdnem teljes tulajdonságot teljesítettük
    - Azért működik ez, mert az eredeti fa majdnem teljes volt



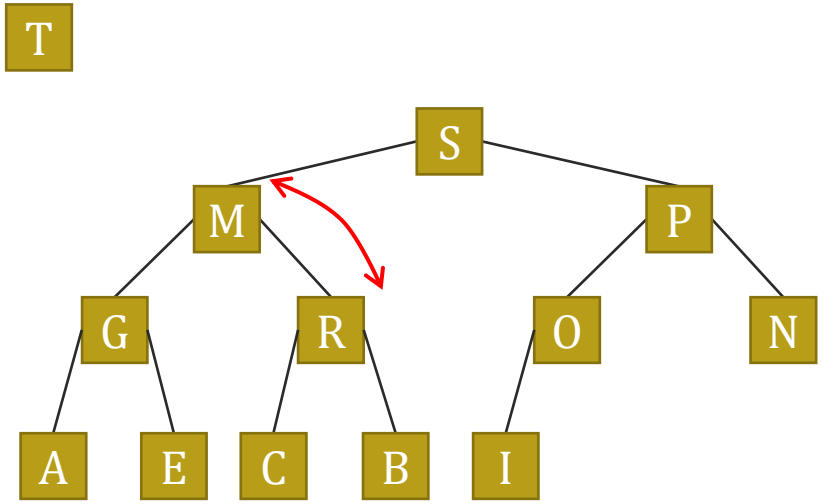
Kupac – helyreállítás

- A kapott majdnem teljes fa azonban nem kupac
  - Tehát a gyökér nem nagyobb a gyerekeinél
  - A helyreállításhoz cseréljük fel a gyökeret a nagyobb gyerekével



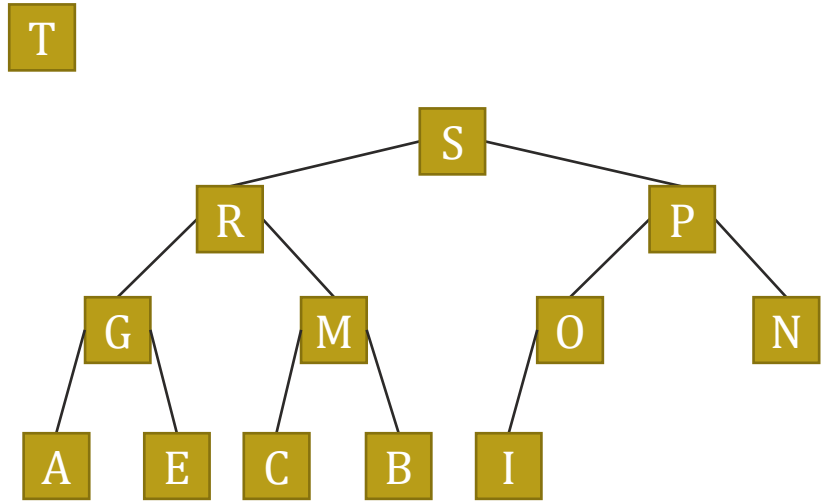
Kupac – helyreállítás

- A jobb részfa kupac, a bal részfa azonban nem teljesíti a kupac tulajdonságot
  - Ismételjük meg az előző lépést a bal részfára



Kupac – helyreállítás

- Ezt a lépést végrehajtva a kapott fa kupac

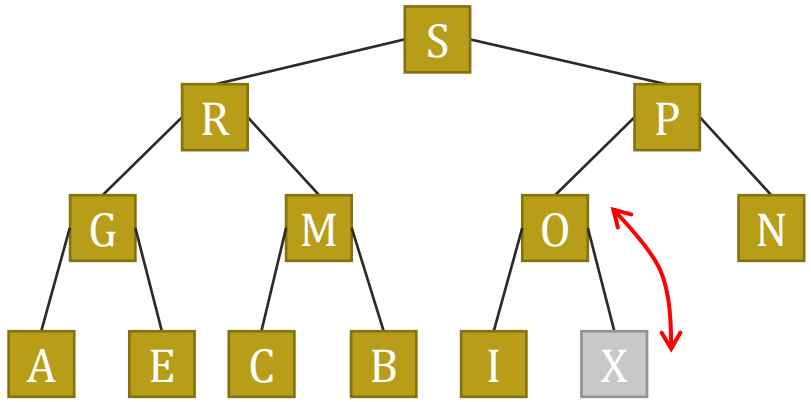


Kupac – Törlés ideje

- Gyökér eltávolítása $\mathcal{O}(1)$
- Utolsó elem a gyökérbe $\mathcal{O}(1)$
- Nagyobb gyerekkel csere $\mathcal{O}(h) = \mathcal{O}(\log_2 n)$
- Összesen (ignorálva a konstansokat) $\mathcal{O}(\log_2 n)$

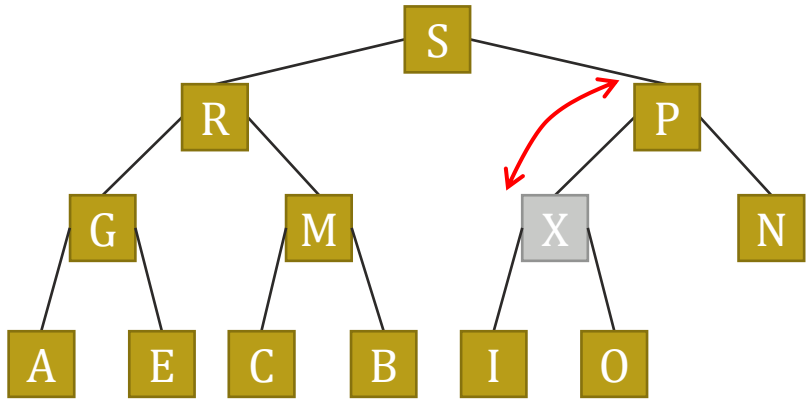
Kupac – elem beszúrása

- Adjunk egy elemet a kupachoz
  - Helyezzük a következő üres pozícióra a jobb szélén
    - Ez kell legyen a következő kitöltendő hely
  - Vigyük felfelé, amíg nagyobb, mint a szülei
  - Újra maximum h csere tehát a beszúrás is $\mathcal{O}(\log_2 n)$



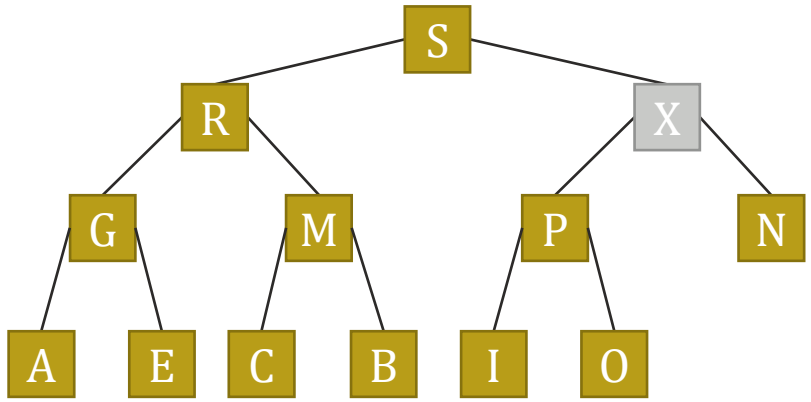
Kupac – elem beszúrása

- Adjunk egy elemet a kupachoz
  - Helyezzük a következő üres pozícióra a jobb szélén
    - Ez kell legyen a következő kitöltendő hely
  - Vigyük felfelé, amíg nagyobb, mint a szülei
  - Újra maximum h csere tehát a beszúrás is $\mathcal{O}(\log_2 n)$



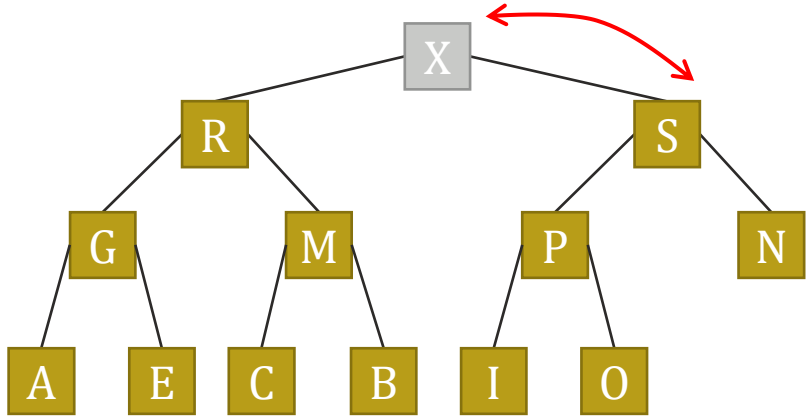
Kupac – elem beszúrása

- Adjunk egy elemet a kupachoz
  - Helyezzük a következő üres pozícióra a jobb szélén
    - Ez kell legyen a következő kitöltendő hely
  - Vigyük felfelé, amíg nagyobb, mint a szülei
  - Újra maximum h csere tehát a beszúrás is $\mathcal{O}(\log_2 n)$



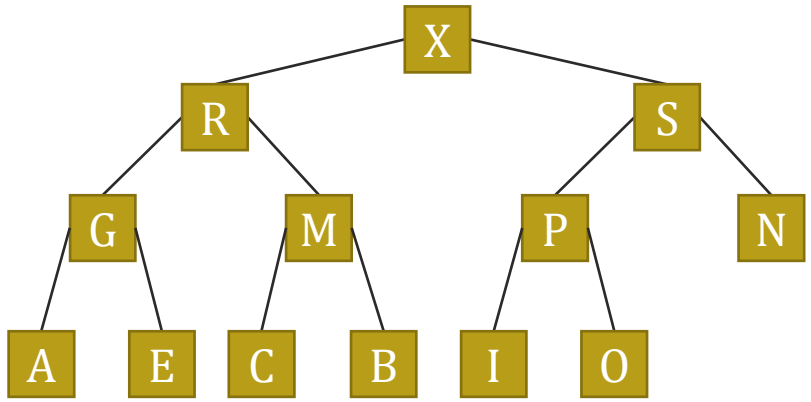
Kupac – elem beszúrása

- Adjunk egy elemet a kupachoz
  - Helyezzük a következő üres pozícióra a jobb szélén
    - Ez kell legyen a következő kitöltendő hely
  - Vigyük felfelé, amíg nagyobb, mint a szülei
  - Újra maximum h csere tehát a beszúrás is $\mathcal{O}(\log_2 n)$



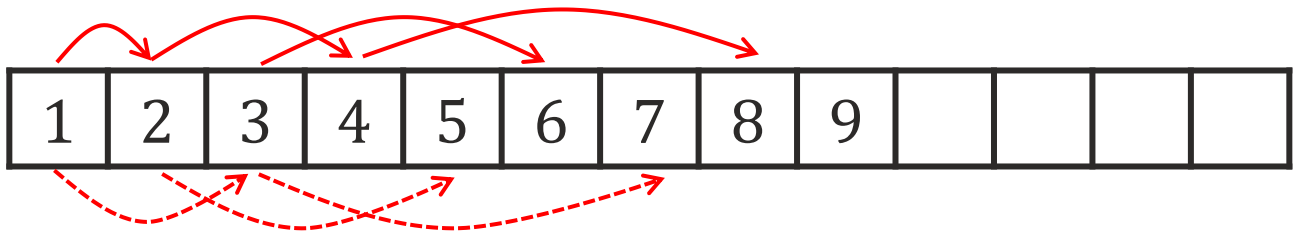
Kupac – elem beszúrása

- Adjunk egy elemet a kupachoz
  - Helyezzük a következő üres pozícióra a jobb szélén
    - Ez kell legyen a következő kitöltendő hely
  - Vigyük felfelé, amíg nagyobb, mint a szülei
  - Újra maximum h csere tehát a beszúrás is $\mathcal{O}(\log_2 n)$



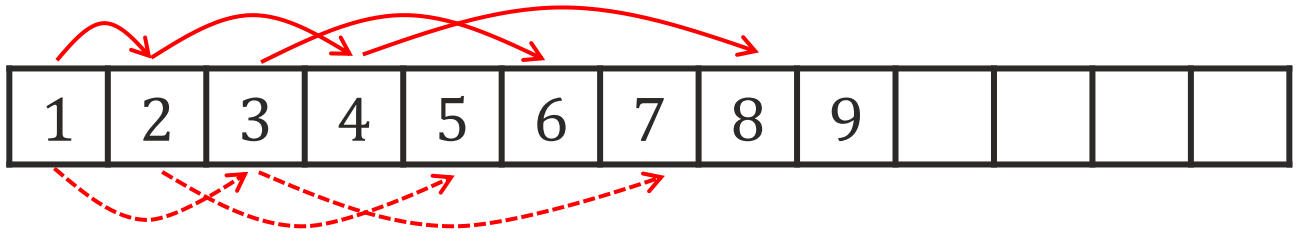
Kupacok – reprezentáció

- Dinamikusan allokált csomópontok és mutatók
  - mint bármilyen más láncolt lista vagy fa
- Használjunk egy tömböt és
  - használjuk ki a „majdnem teljes” tulajdonságot
  - a k csomópont gyerekei a $2k, 2k + 1$-nél vannak
  - a k szülője a $\frac{k}{2}$-nél van
  - ha $k > n$, akkor a csomópont nem létezik



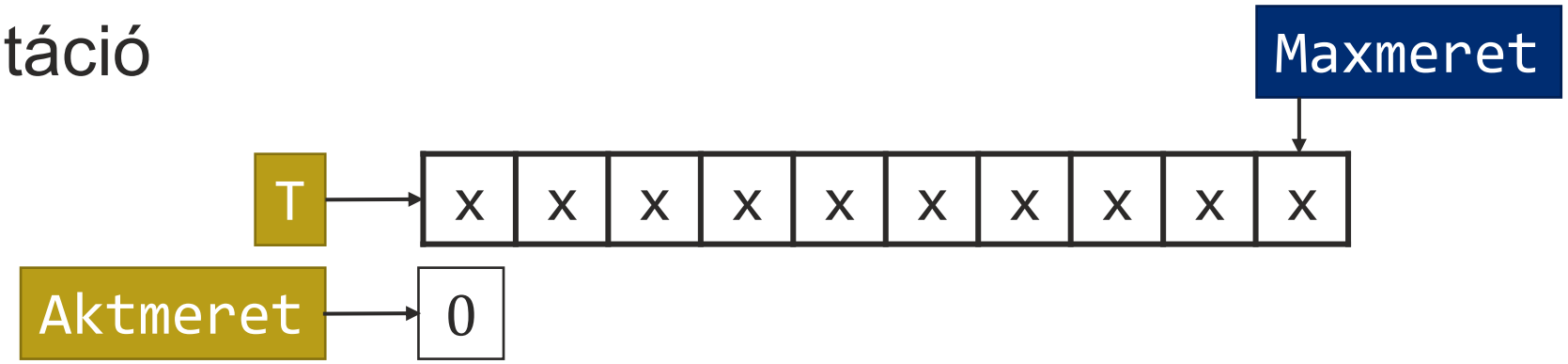
Kupac – hatékonyság

- A beszúró és törlő műveletek egy h magasságú fában
  - $h \leq \log_2 n$
  - Vagyis $\mathcal{O}(\log n)$ az időigénye



Kupac műveletei

- Reprezentáció



- Empty: $\rightarrow K$ az üres kupac létrehozása
  - A reprezentáló tömb létrehozása a `Maxmeretnek` megfelelően
  - `Aktmeret` beállítása nullára
- IsEmpty: $K \rightarrow L$ üres a kupac?
  - `return (Aktmeret = 0)`

Kupac műveletei

- Insert: $K \times N \rightarrow K$ elem betétele a kupacba

Aktmeret≠Maxmeret	
$\text{Aktmeret} \leftarrow \text{Aktmeret} + 1$ $T[\text{Aktmeret}] \leftarrow \text{ujelem}$ $\text{Szulo} \leftarrow \text{Aktmeret} / 2$ $\text{Gyerek} \leftarrow \text{Aktmeret}$	Tele!
$(Szulo \geq 1) \wedge (T[\text{Szulo}] < T[\text{Gyerek}])$	
$\text{Csere}(T[\text{Szulo}], T[\text{Gyerek}])$ $\text{Gyerek} \leftarrow \text{Szulo}$ $\text{Szulo} \leftarrow \text{Szulo}/2$	

Kupac műveletei

- Max: $K \to E$ maximális elem lekérdezése

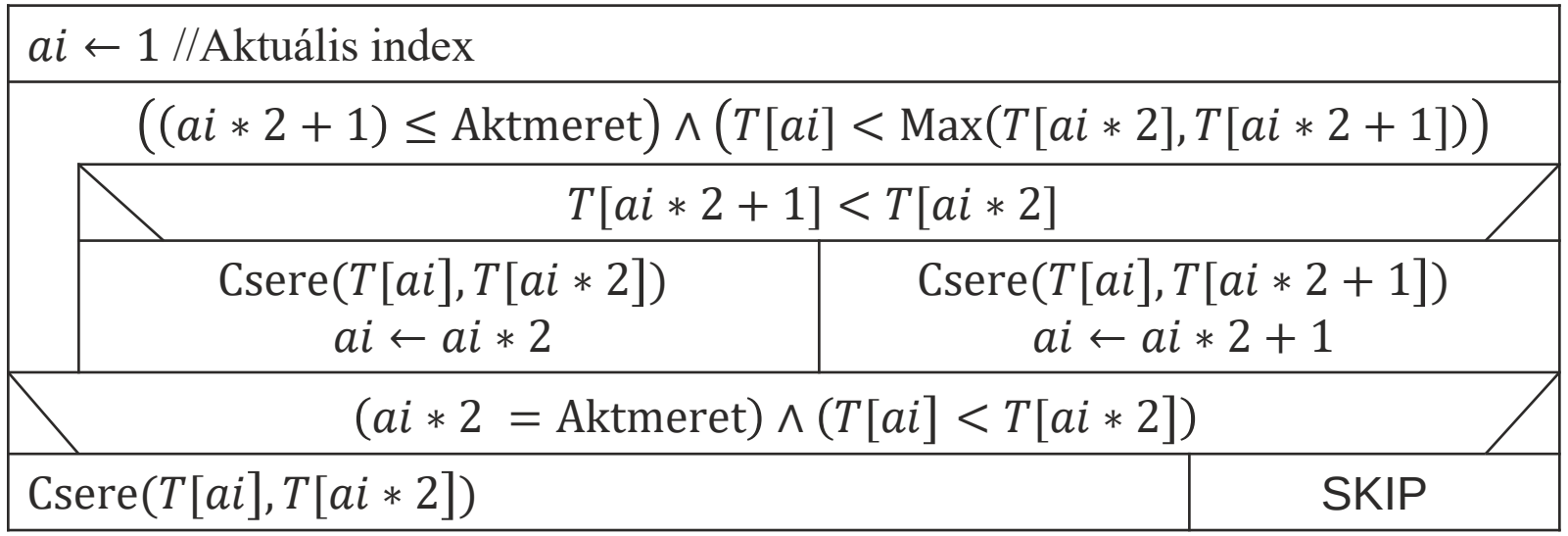
Kupac műveletei

- DelMax: $K \rightarrow K \times N$ maximális elem kivétele a kupacból
  - A maximális elem értékét a maxelem-ben kapjuk

Aktmeret≠ 0	
Maxelem← $T[1]$ $T[1] \leftarrow T[\text{Aktmeret}]$ $\text{Aktmeret} \leftarrow \text{Aktmeret} - 1$ Sullyeszt return Maxelem	Üres!

Kupac műveletei

- Sullyeszt: feltételezzük, hogy a kupacban két jó részfa van, de a $T[1]$-t megfelelően le kell süllyeszteni



Prioritásos sor megvalósítása kupaccal

- `Empty` (az üres prioritásos sor konstruktor – létrehozás)
  - `Empty` (üres kupac)
- `isempty` (üres a prioritásos sor?)
  - `IsEmpty` (üres a kupac?)
- `insert` (elem betétele a prioritásos sorba)
  - `Insert` (elem betétele a kupacba)
- `delmax` (maximális elem kivétele a prioritásos sorból)
  - `DelMax` (maximális elem kivétele a kupacból)
- `max` (maximális elem lekérdezése a prioritásos sorból)
  - `Max` (maximális elem lekérdezése a kupacból)

Bináris keresőfa

Következő téma