Université de La Rochelle

Universiteti i Prishtinës

Zgjedhja e strukturave të të dhënave në Java

K. Bertet, F. Berisha

Plani

- Të dhënat FIFO (First In First Out) të menagjuara nga një Queue
- Të dhënat LIFO (First In First Out) të menagjuara nga një Stack
- Të dhënat e indeksuara (indeks, çelës, id)...
  - ... të menagjuara nga një listë
  - ... të menagjuara nga një pemë binare
  - ... të menagjuara nga një hash tabelë

E dhëna

Atribute (gjendja)

- `int id`
- `String label`
- ……

Metoda (sjellja)

- `Data(int id, String label)`
- `public boolean equals(int id)`
- `public int compareTo(Data d)`
- `public int getId()`
- `public String toString()`

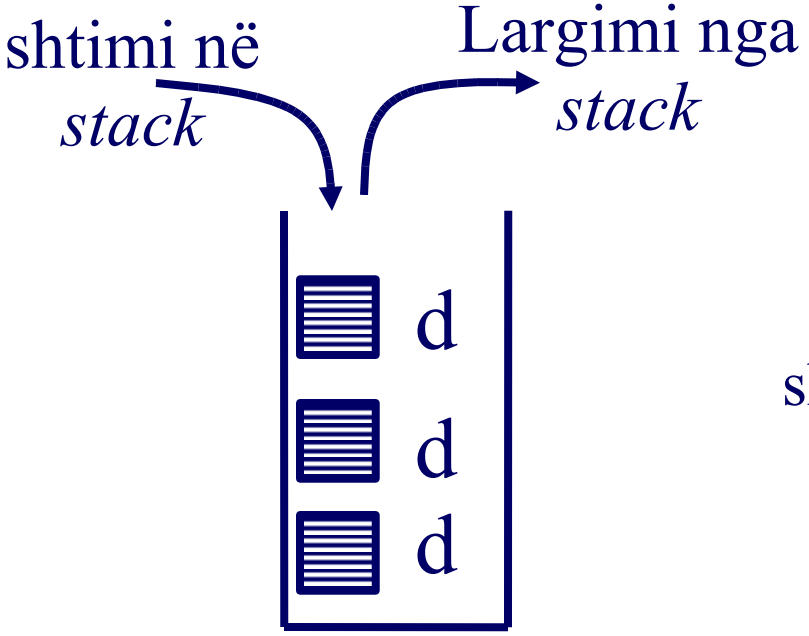
Të dhënat FIFO dhe LIFO

Stack (mullarët) dhe queue (rradhët) janë struktura të dhënash sekuenciale (*lista*) sjellja e të cilave definohet me:

- **shtim** te njëri skaj
- **largim** te njëri skaj
- pa kërkim

⇒ Këto dy operacione varen nga renditja e arritjes së të dhënave, e jo nga vlera e tyre (*id).*

Të dhënat FIFO dhe LIFO

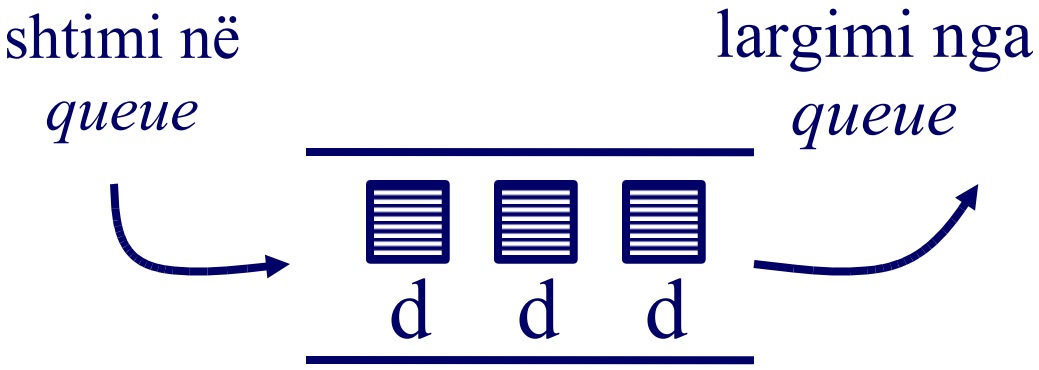


Stack (mullar)

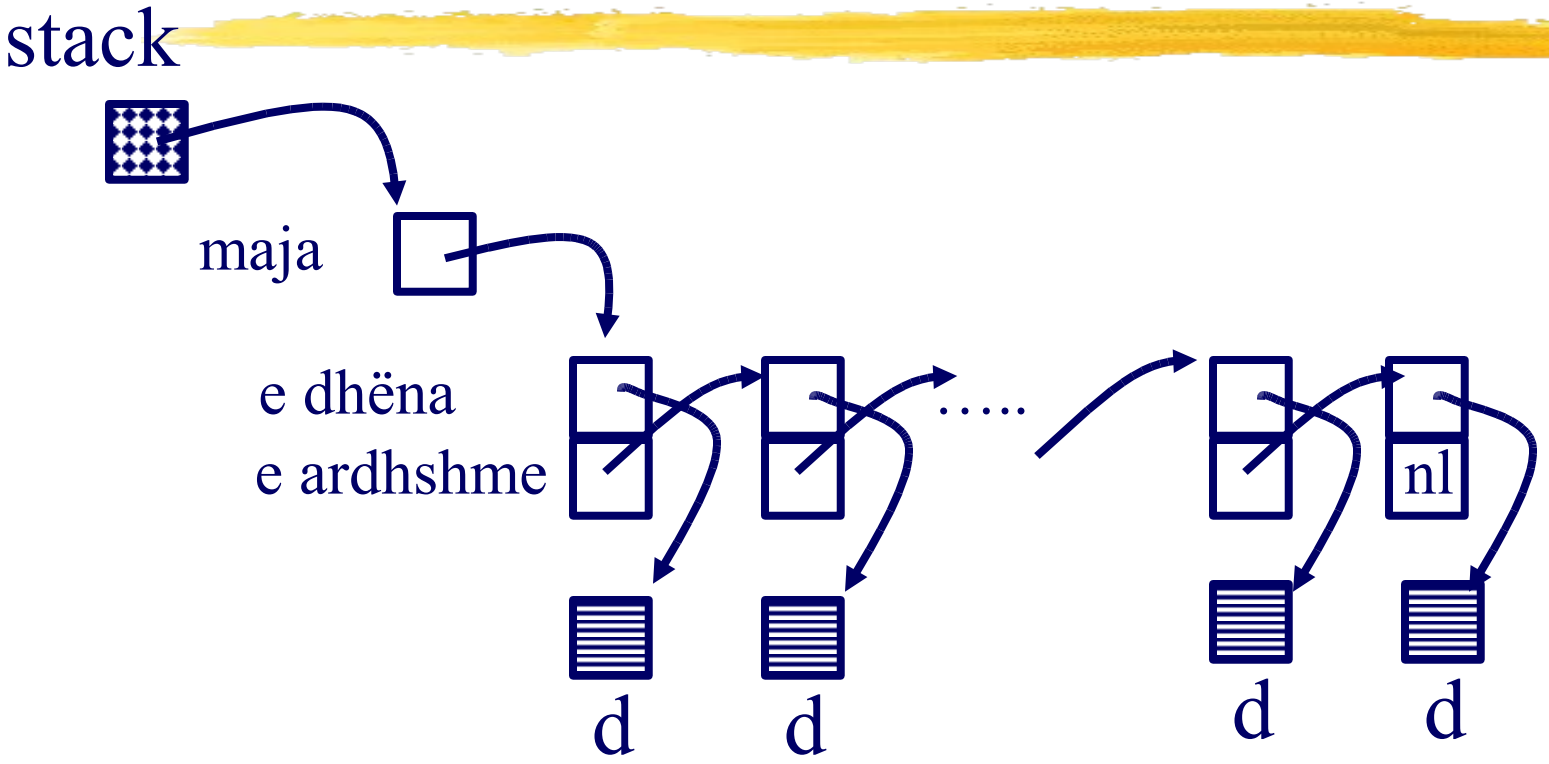
LIFO: *Last In, First Out*

Queue (rradhë)

FIFO: *First In, First Out*



Stack: paraqitja



Stack: metodat

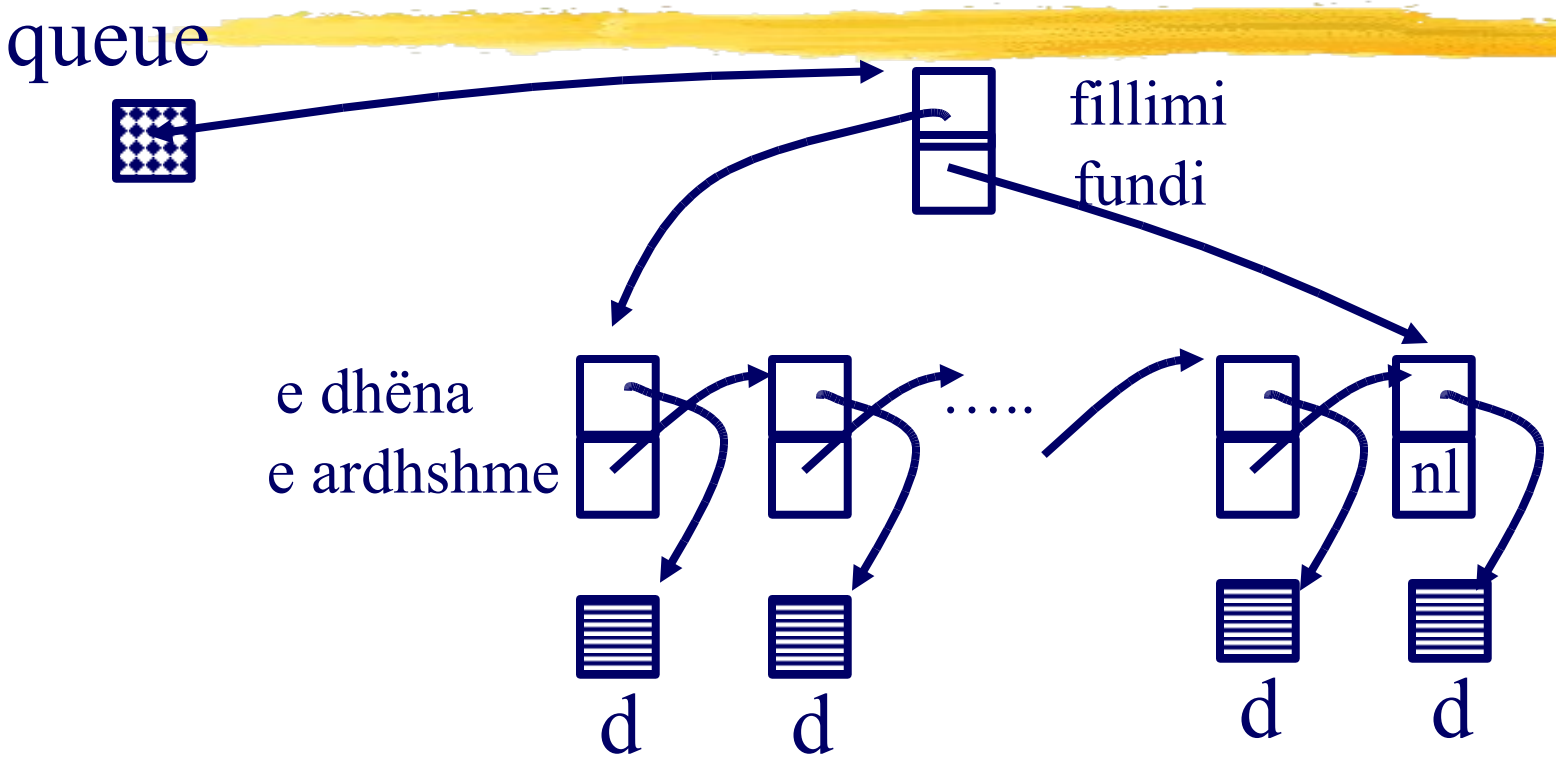
`public void push(Data d):` $O(1)$

`public Data pop():` $O(1)$

`public int size():` $O(1)$

(`public String toString():` $O(n)$)

Queue: paraqitja



Queue: metodat

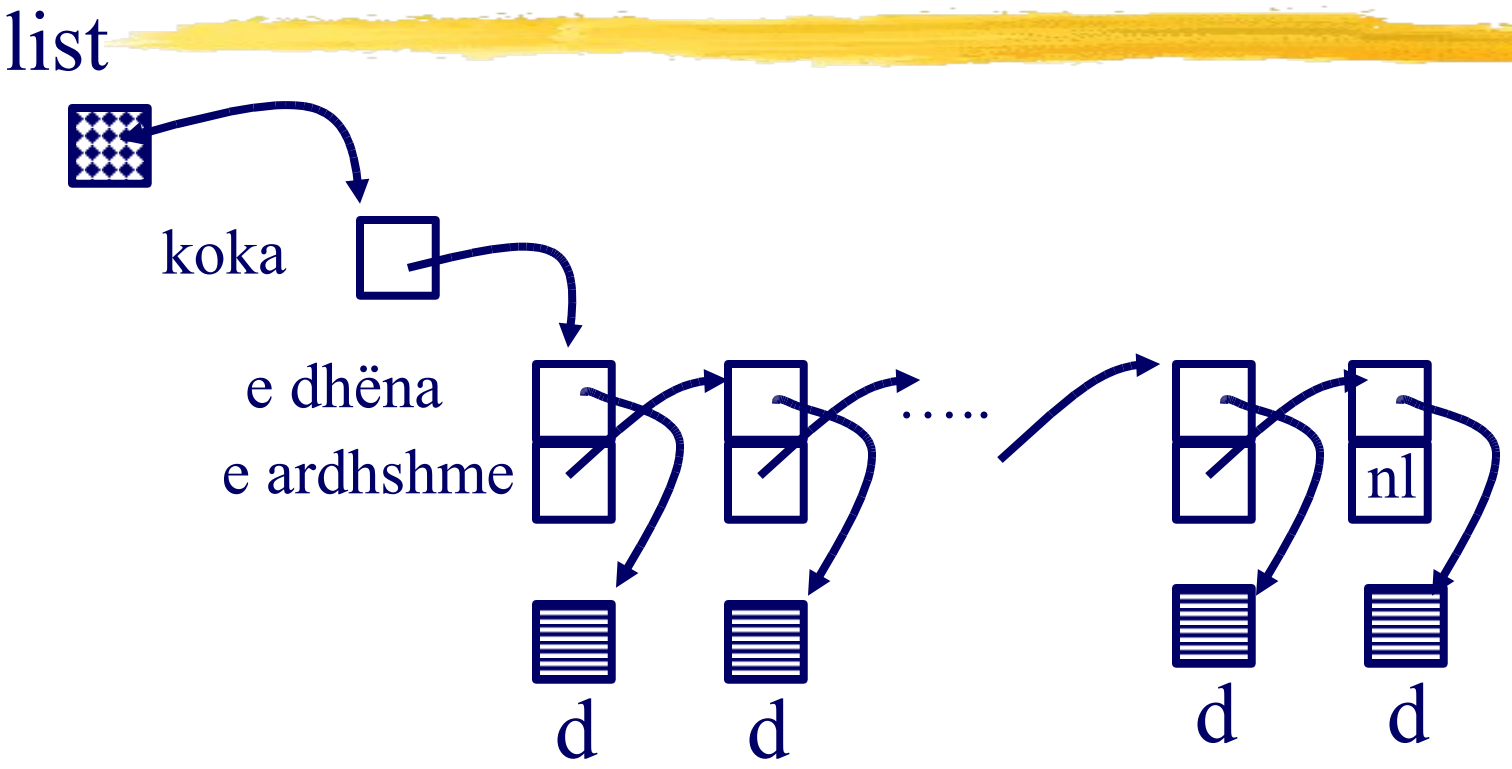
`public void add(Data d):`	O(1)
`public Data poll():`	O(1)
`public int size():`	O(1)
`( public String toString ():`	O(n))

Të dhënat e indeksuara

• Listat, hash tabelat dhe pemët binare janë struktura të dhënash të cilat lejojnë menagjimin e koleksione të dhënash të indeksuara (me anë të një indeksi).

• Sjellja e tyre definohet me *shtimin, largimin* dhe *kërkimin* e një të dhëne përmes indeksit të saj.

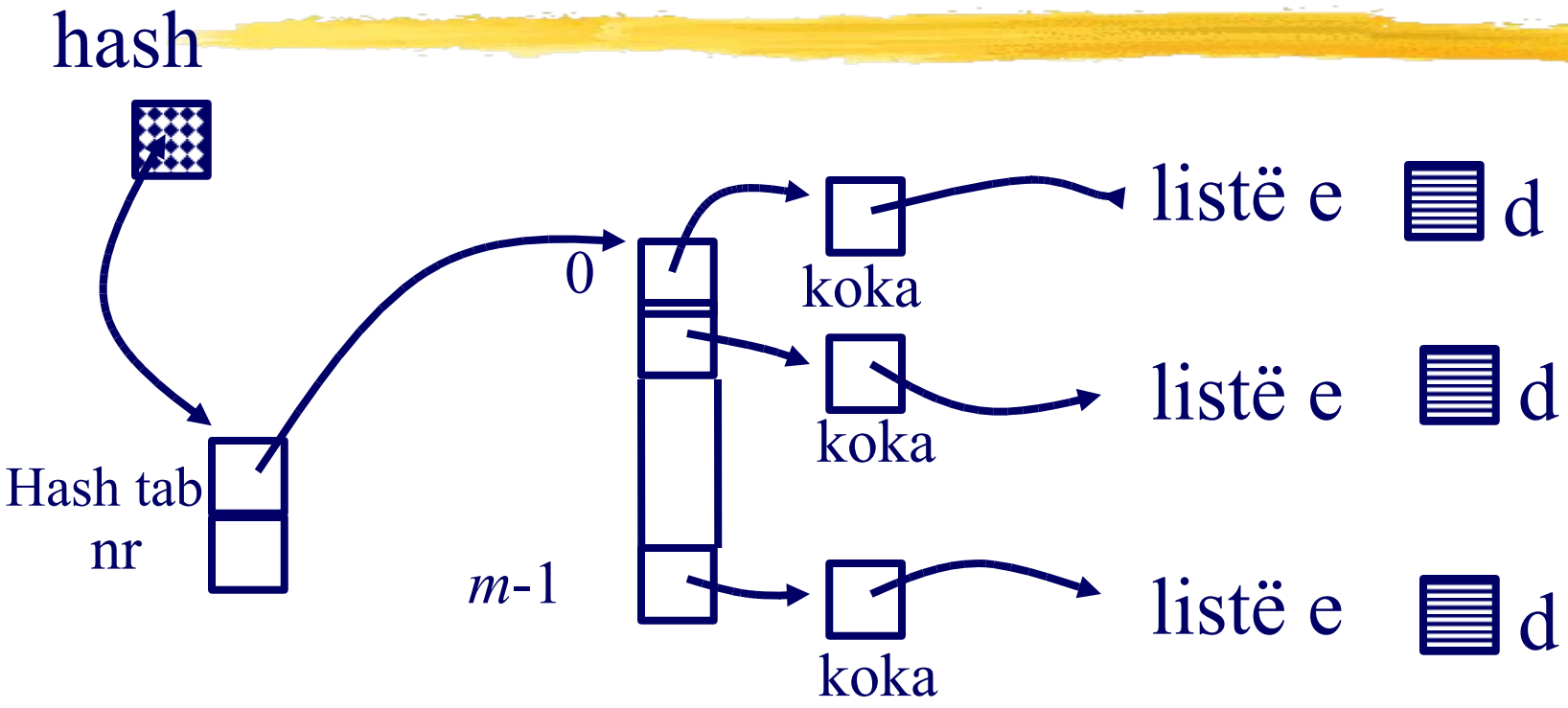
List: paraqitja



List: metodat

`public void add(Data d):`	O(1)
`public Data get(index):`	O(1)
`public Data remove(index):`	O(n) ose O(1)
`public int size():`	O(1)
`public String toString():`	O(n)

Hash tabelë: paraqitja



Hash tabelë: përshkrim

• Një hash funksion (funksion i cungimit) *h* i korrespondon çdo të dhëne (*id*) një dhe vetëm një listë:

$h(id)$ = nr i listës (ndërmjet 0 dhe m-1)

• Shembull hash funksioni:

$$h(id) = id \bmod (m-1)$$

• Hash funksione të sofistikuara garantojnë një shpërndarje të mirë në mesatare të *n* të dhënash në *m* lista:

gjatësia maksimale e listës: $L \approx n/m$

Hash tabelë: përshkrim

• Kur numri n i të dhënave rritet, mund të rritet numri m i listave për të garantuar:

$$min < n/m < max$$

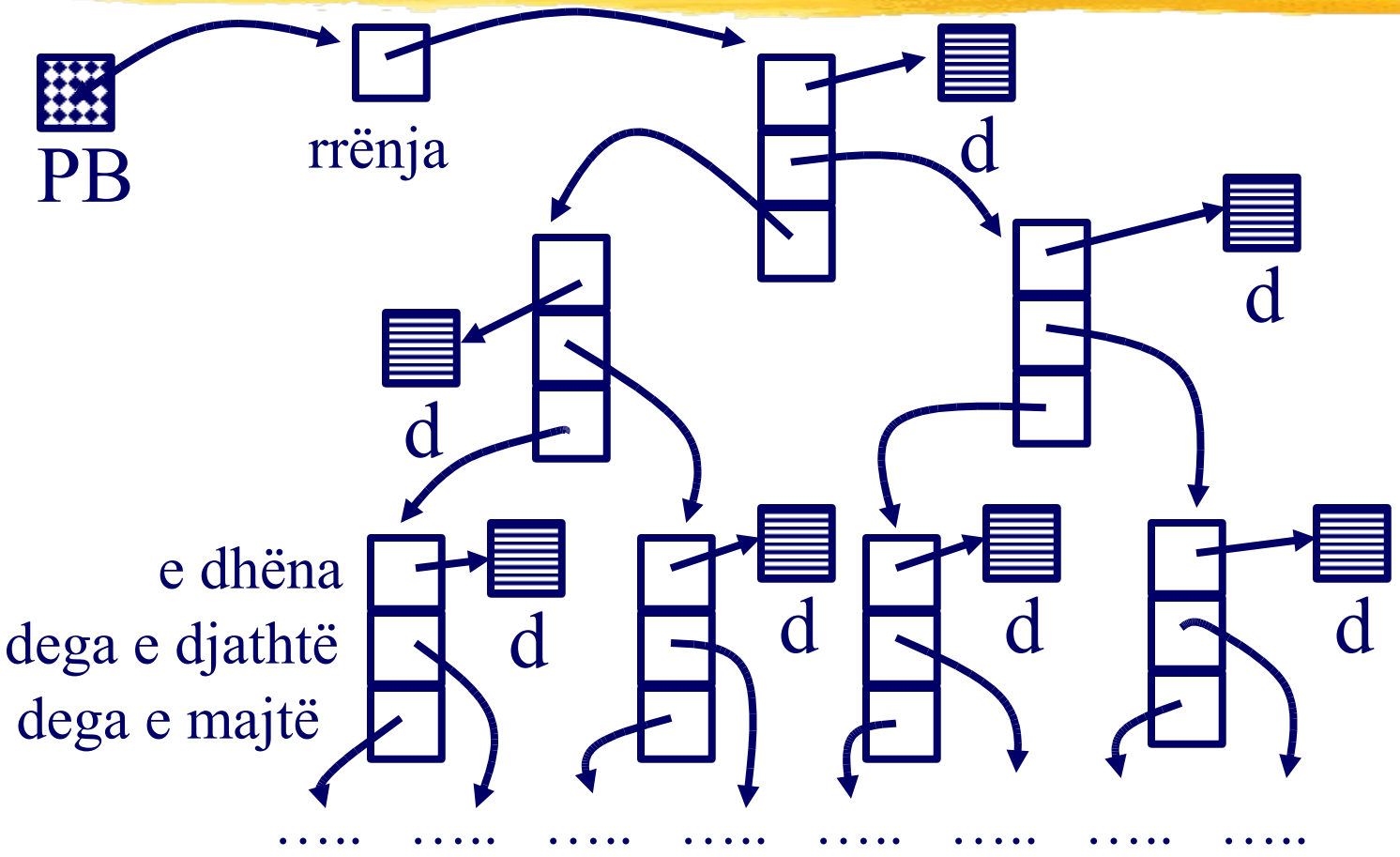
• Atëherë, mund të fitohet vlerësimi vijues:

$$O(L) = O(n/m) = O(max) = O(1)$$

Hash tabelë: metodat

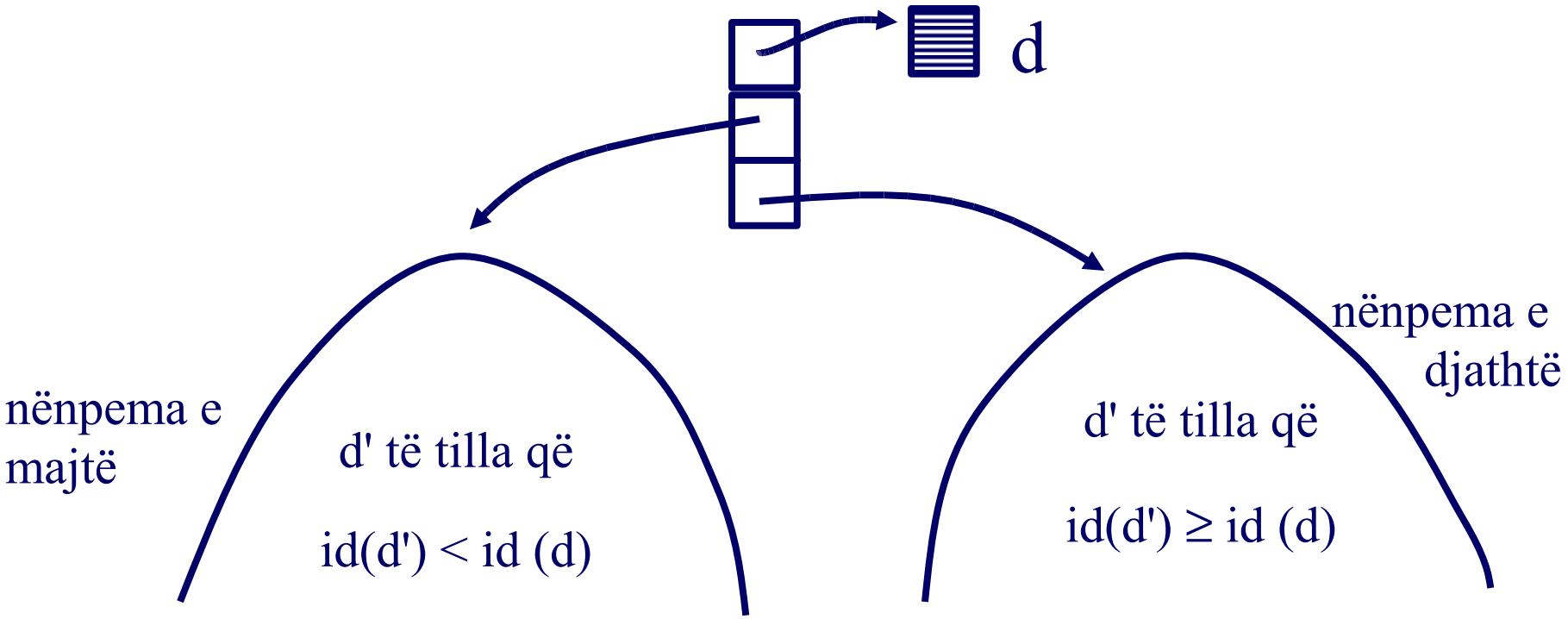
`public void add(Data d):`	$O(1)$
`public Data get(index):`	$O(L)=O(1)$
`public Data remove(index):`	$O(L)=O(1)$
`public int size():`	$O(1)$
`public String toString ():`	$O(n)$

Pemë binare: paraqitja



Pemë binare: përshkrim

- Shtimi duhet ta plotësojë vetinë vijuese:



Pemë binare: përshkrim

• Shtimi, largimi dhe kërkimi kërkojnë thjesht një kalim të plotë të tërë gjatësisë së njërës degë të pemës:

$$gjatësia\ e\ degës \leq lartësia\ e\ pemës = h$$

• Kur një pemë është e ekuioibruar, kemi:

$$h \approx log\ n$$

• Ekzisotjnë algoritma të sofistikuar për riekuilibrimin e një pemë të çekuilibruar.

Pemë binare: metodat

`public void add(Data d):`	$O(h)=O(\log n)$
`public Data get(id):`	$O(h)=O(\log n)$
`public Data remove(id):`	$O(h)=O(\log n)$
`public int size():`	$O(1)$
`public String toString():`	$O(n)$