# Keresőfák (Bináris keresőfa)

Cél: Kulcsos adatrekordok hatékony tárolása.

Adatfeldolgozás feltétele: Minden kulcs egyedi. A duplikált kulcs hibának számít (Rendezésnél ez megengedett.)

Motiváció: Lineáris tárolás és elérés nem hatékony. vagyis

## Bináris keresőfa

Ez a legrégebbi.

"Dinamikus" (hogyan kell felépíteni) definíció: Legelsőből elkezdem a fát. Utána a bejövő elemeket a "kisebb balra megy, nagyobb jobbra" szlogen alapján rakom be.

## Példa

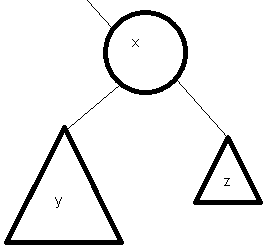
Bemenet:

ADS szintű rajz, de az ábrázolás szintjén gondolkodunk:

|  |  |  |
| --- | --- | --- |
| p1 (→100) | | |
| p3 (→10) | 30 | p4 (→50) |

Minden kapcsolat oda-vissza pointer.

"Statikus" definíció: belső pontra esetén: .



Fontos tulajdonság: inorder bejárás esetén rendezett kulcssorozatot kapunk.

## Műveletek

Minden adatszerkezetben van: Keresés, beszúrás, törlés

Kell még: Minimális kulcs elérése (és max), Következő kulcsra lépés (és előző)

Azok az adatszerkezetek, amelyekre ez az 5 művelet definiálva van, a "Szótár".

Műveletek: (Javasolt hívási forma bekeretezve.)

### Keresés

(: fa, : kulcsérték)

3 példa:

Struktogramja:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | | |
|  | | |  | |
|  | | | | |
|  |  | | | |
|  |  | | |
|  | |  |

Ez egy bejárás.

Iteratívan:

|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
|  | |  | |
|  | | | |
|  | | | |
|  |  | | |
|  | |  |
|  | | | |

### Beszúrás

Paraméterlista: fa, mit ( pointer egy már elkészített, komplett rekordra)

-t pedig hibajelzésre használjuk. (Ha sikerült beszúrni pointer lesz, ha nem, akkor NULL.)

Ez a 110-től balra lesz felvéve.

|  |  |  |
| --- | --- | --- |
| (→100) | | |
| (→115) | 110 | (→120) |

Ez a 70-től jobbra lesz felvéve.

|  |  |  |
| --- | --- | --- |
| (→80) | | |
| (→60) | 70 | (→75) |

Ez nem lesz felvéve, dupla kulcs lenne.

### Törlés

A törlés nem egyszerű. Nem fizikailag, hanem logikailag kell kitörölni.

A által mutatott elemet, pontosabban annak tartalmát kell törölni. A módosítható, ha kell. (Szükség esetén átrendezhető.)

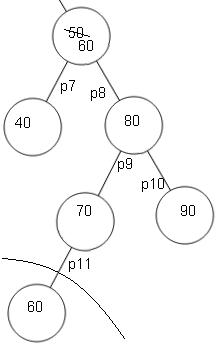
3 típusú törlést fogunk megismerni.

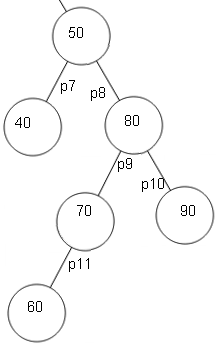
Levél:

(Ez a -asra mutat, ami egy levél.) Ekkor annyit kell tenni, hogy jobb pointerét lenullázom.

Nem levél, de csak egy gyereke van:

(Ez a -esre mutat, ami egy egygyerekes kulcs.) Ekkor annyit kell tenni, hogy kiláncolom.

Két gyereke van:

 (Ez az 50-re mutat, aminek két gyereke is van. Sőt, jobb oldali gyereke komplett fa.)

Rákövetkezőjét (60) az 50 helyére rakjuk, majd kitöröljük.

Általánosan: Jobb részfája minimális elemét töröljük, előbb azonban adataival, kulcsával felülírjuk a törlendő elemet.

### Minimum kulcs

Mindig balra megyünk.

Üres fára NULL-t ad.

### Következő kulcs

Feltétel: helyes pointer.

Az ez utáni kulcsot tartalmazó rekordot keressük.

( az 50-re mutat, következő lesz a 60, ennek pointere a )

Jobb oldali részfa minimuma (Megnézi, van-e jobb ága, ha van, azon a részfán belül keres minimumot.)

( a 90-re mutat, következő lesz a 100, ennek pointere a )

( a 20-ra mutat, következő lesz a 30, ennek pointere a )

Ha nincs jobb ága, akkor "Balra föl ameddig lehet és egyet jobbra föl." Ez hogy lehet formálisan elmondani? Nézzük meg fordítva: Az elemet úgy látjuk, hogy elmegyünk egyet balra és ott keresünk maximumot.

( az a , nincs következő)

Nincs jobb oldali leágazása, az első módszer nem működik. A második módszer sem.

### Vizsgára

Fenti műveleteket struktogrammal együtt tudni kell, kivéve a törlést, amit elég rajzban végrehajtani tudni.

## A bináris keresőfa műveleteinek hatékonysága

Keresés, beszúrás, minimum: Elindulok a gyökérből és cikk-cakkban megyek valameddig.

Következő, törlés: Útvonal a fában, gyökér-levél, ami a fa teljes magasságával egyenlő, vagy rövidebb.

Tehát minden művelet egy útvonalat jár be a fában. Ezért (Bármely művelet futásideje egy fára = Ordó a fa magassága)

Fontos tehát a fa magassága. Legrosszabb esetben , amikor lineáris a sorozat (1, 2, 3, 4, 5).

## Egy bemenetre ugyan annyi féle fa van, mint permutáció? Nem.

Az kulcsok sorrendben következhetnek, a keresőfák száma ennél kevesebb.

Ha minden bemenet permutáció egyformán valószínű lenne az életben, és nagyon sokszor felvennénk a fát, akkor átlagosan milyen magas lenne?

, ahol és között van. ()

De a gyakorlatban ez nem így van. A feltevés nem teljesül.

* Cégnél adatok adminisztratív összerakása egy meghatározott módon történik. Nem random, van benne valamilyen tendencia.
* Ha törlés is van, nem csak beszúrás, akkor nem is értelmező mi az, hogy kulcssorrend valószínűsége.

E miatt fel kell készülnünk arra, hogy a fánk () mégis csak megnyúlhat. -t és így -t karban kell tartani.

Minden művelet után olcsó eljárással lekérdezhetjük, szét van-e csúszva a fa. És egy másik olcsó művelettel összeránthatjuk. Tehát a karbantartás sem viszi ki a faműveleteket a tartományból.

Mi az AVL-fát fogjuk erre tanulni. Hasonló a piros-fekete fa (red-black tree)

Előadás vége.