Eötvös Loránd Tudományegyetem

Informatikai Kar

B+ fa műveleteinek grafikus szemléltetése

Témavezető:

dr. Ásványi Tibor

egyetemi docens

Készítette:

Varga Balázs

Programtervező Informatikus

BSc. szakos hallgató

Budapest, 2016

Tartalomjegyzék

1. Bevezetés

Az adatbázisokban, fájlrendszerekben kulcsfontosságú, hogy az adatokat strukturáltan, rendezve tároljuk. Így az alapvető műveletek; keresés, beszúrás, törlés hatékonyan hajthatóak végre. A B+ fa tökéletesen kiegyensúlyozott keresőfa, ezért a fa magassága O(lg n), ahol n a fában tárolt kulcsok száma. Ez lehetővé teszi, hogy akár milliárd nagyságrendű adathalmazon is gyorsan végezhessünk műveleteket. A gyakorlatban számos ismert fájlrendszer: NTFS, BFS használja például a könyvtárak indexeinek vagy adatainak tárolására [1]. Relációs adatbázis rendszerekben is segítséget nyújt a B+ fa, tábla indexelések megvalósítására. Ilyen adatbázisok például a Microsoft SQL Server, Oracle 8, SQLite [1]. A program célja, hogy megismertesse a felhasználót a B+ fa alapvető műveleteivel grafikusan ábrázolva, szöveges leírással segítve a megértésüket.

2. Elméleti háttér

2.1. Matematikai alapfogalmak

Alapvető fogalmak, amelyek elengedhetetlenek, ha fákról, gráfokról van szó.

Irányított gráf definíciója

Egy irányított gráf alatt egy G = (V, E) párost értünk, ahol V a csúcsok véges halmaza, $E \subseteq V \times V$ pedig az élek halmaza. $(u, v) \in E$ esetén az (u, v) az u csúcsból, a v csúcsba vezető él [2].

Kifok definíciója

Adott csúcsból kivezető élek száma a csúcs kifoka [2].

Befok definíciója

Adott csúcsba vezető élek száma a csúcs befoka [2].

Irányított séta definíciója

Legyen G = (V, E) irányított gráf. A $v_0, e_1, v_1, e_2, \ldots, v_{n-1}, e_n, v_n$ n hosszú sorozatot irányított sétának nevezzük. Ekkor $v_{i-1}, v_i \in V$ $(0 \leq i \leq n)$, $e_j \in E$ $(1 \leq j \leq n)$, ahol e_j a v_{i-1}-ből v_i-be vezető él [2].

Kör definíciója

Ha egy irányított sétában a kezdő és végpont ismétlődésén kívül nincs más csúcsismétlődés, akkor körnek nevezzük [2].

Összefüggőség definíciója

Egy irányított gráfot összefüggőnek nevezünk, ha az irányítás elhagyásával bármely két csúcsa összeköthető sétával [2].

Irányított fa definíciója

Az irányított fa olyan irányított gráf, amely összefüggő, körmentes és van egy olyan csúcsa, amelynek a befoka 0, az összes többi csúcs befoka 1 [2]. Ha a gráf csúcshalmaza és élhalmaza üres, akkor a fa üres.

2.2. B+ fa definíciók

Keresőfa jellemzése

A keresőfák közös jellemzője, hogy a csúcsokban lévő rendezett kulcsok alapján hatékonyan meghatározhatjuk, hogy melyik részfában kell keresni egy adott kulcsot.

Szülő definíciója

A fa azon csúcsa, amelyből vezet ki él, tehát a kifoka ≥ 1. A B+ fában ezek lesznek a **belső csúcsok**, ezekben tároljuk a hasító kulcsokat.

Gyerek definíciója

A fa olyan csúcsa, amelynek van szülője, a befoka 1 és vezethet ki belőle él, kifoka ≥ 0.

Nagyszülő definíciója

A fa olyan speciális szülő csúcsa, aminek legalább két szinttel lejjebb található a gyereke, azaz a nagyszülő egy adott csúcs szülőjének szülője.

Gyökér definíciója

Az irányított fában az a csúcs, amelynek a befoka 0 [2]. Vagyis ez lesz a fa azon kitüntetett csúcsa, aminek nem lesz szülője.

Levél definíciója

Olyan speciális csúcs, melynek a kifoka 0, nem vezet már ki belőle él, a B+ fában ezekben a csúcsokba fogjuk tárolni a **kulcsokat**.

Részfa definíciója

Legyen F fa, ekkor F' nemüres részfája F-nek, ha F tetszőleges belső csúcsa gyökere F'-nek.

B+ fa definíciója

A B+ fa olyan keresőfa, amiben minden csúcs legfeljebb d mutatót (4 ≤ d) és legfeljebb d - 1 kulcsot tartalmaz. A fában tárolt adatok (és pointere) a hozzájuk tartozó kulcsokkal együtt a levelekben vannak. A belsőkulcsok csak hasító kulcsok. Egy adott kulcsú adat keresése során ezek alapján tudhatjuk, melyik ágon keressünk tovább. A belső csúcsokban mindegyik részfa két hasító kulcs között van kivéve az első, ami az első hasító kulcstól balra és az utolsó, ami az utolsó hasító kulcsot jobbra található. A gyökértől

mindegyik levél ugyanolyan távolságra helyezkedik el, vagyis minden levél azonos mélységben, a legalsó szinten van. A B+ fára a következő invariánsok teljesülnek:

- Minden csúcsban eggyel több mutató van, mint kulcs, ahol d ($4 \leq d$) a felső határ a mutatók számára.
- A gyökértől mindegyik levél ugyanolyan távol található.
- Minden *Cs* belső csúcsra, ahol *k* a *Cs* csúcsban a kulcsok száma: az első gyerekhez tartozó részfában minden kulcs kisebb, mint a *Cs* első kulcsa; az utolsó gyerekhez tartozó részfában minden kulcs nagyobb-egyenlő, mint a *Cs* utolsó kulcsa; és az *i*-edik gyerekhez tartozó részfában ($2 \leq i \leq k$) lévő tetszőleges *r* kulcsra *Cs*.kulcs[i-1] $\leq r <$ *Cs*.kulcs[i].
- A gyökér csúcsnak legalább 2 gyereke van, kivéve ha ez a gyökér a fa egyetlen levele.
- Minden gyökértől különböző belső csúcsnak legalább floor(d / 2) gyereke van (d = 4-re legalább 2 gyereke és minimum 1 hasító kulcsa van).
- Minden levél legalább floor(d / 2) kulcsot tartalmaz (d = 4-re ez minimum 2 kulcs)
- A B+ fában tárolt kulcsok megjelennek valamely levélben, balról jobbra szigorúan monoton növekvő sorrendben [3].

Fa magasság definíciója:

A gyökértől egy levélig vezető séta hossza. A B+ fa esetében ez tetszőleges levél. A fa magassága: O(lg n), ahol n a B+ fával ábrázolt adathalmaz mérete.

Keresés a fában

A gyökérből kiindulva a hasító kulcsok és a keresendő kulcs összehasonlításával, a levélig eljutva, megállapítható, hogy egy adott érték benne van a fában vagy nincs.

Kulcsok száma

A programban d = 4 mutató számmal dolgozok, azaz a levelek maximum 4 - 1 = 3 és minimum *floor*(4 / 2) = 2 kulcsot tartalmaznak.

Hasító kulcsok száma

A programban d = 4 mutató számmal dolgozok, tehát minden belső csúcs maximum 3 és minimum 1 hasító kulcsot tartalmaznak.

Mutatók, részfák, gyerekek száma

A programban d = 4 mutatószámmal dolgozok, ezért a gyökérnek és a belső csúcsoknak maximum 4 minimum 2 részfájuk, gyerekük lehet.

Baloldali kulcs

Az előző definícióban leírt módon, a baloldali kulcs, mindig az adott csúcs 1. kulcsa lesz.

Középső kulcs

Ha a csúcs tele van, vagyis a kulcsok száma 3, akkor a középső kulcs a 2. kulcs lesz.

Jobboldali kulcs

Mivel minden levél $2 \leq k \leq 3$, kulcsot tartalmazhat, ezért a jobboldali mindig a csúcs utolsó kulcsa lesz.

Zárójelezett (szöveges) alak

Egy negyedfokú (d = 4) B+ fa zárójelezett alakja a következőképpen épülhet fel, ha a fa nem egyetlen levélből áll:

- $(T_1, k_1, T_2, k_2, T_3, k_3, T_4)$
- $(T_1, k_1, T_2, k_2, T_3)$
- (T_1, k_1, T_2).

Ahol T*i* az *i*-edik részfa, k_i pedig a hasító kulcs. A leveleket pedig kétféleképpen írhatjuk le: (k_1, k_2) vagy (k_1, k_2, k_3), ahol k_i a levélben található kulcsok. Ha a fában csak egy kulcs található, akkor (k_1) alakban irható fel.

3. Műveletek

3.1. Keresés

A gyökérből indulva, összehasonlítjuk a keresett kulcsot, a hasító kulcsokkal és a két kulcs közötti reláció alapján eldöntjük, hogy melyik részfában folytassuk a keresést, ezt addig ismételjük, míg levél csúcsot nem kapunk. Ha találunk egyezést a levélben, akkor a fa tartalmazza a keresett kulcsot. A keresés algoritmusa igazzal tér vissza, ha megtalálható a keresett érték a fában és hamissal, ha nem. Első lépésben megadjuk a fa gyökerét kezdőcsúcsnak, és a keresett kulcsot. Ha egyszintű a fa és találunk egyezést a gyökér (levél) kulcsai és a keresett kulcs között akkor leáll a keresés, megtaláltuk a kulcsot. Ha több szintes a fa, akkor először azt ellenőrizzük, hogy a keresett kulcs kisebb-e az első hasító kulcsnál, ha igen akkor az első hasító kulcs baloldali részfájában keresünk tovább. Ha nagyobb, akkor megkeressük az utolsó olyan hasító kulcsot, ami még kisebb-egyenlő a keresett kulcsnál és folytatjuk ennek a hasító kulcsnak a jobboldali részfájában a keresést. Mindkét esetben előbb utóbb levél csúcshoz jutunk, ahol már tudjuk ellenőrizni, hogy a keresett kulcs benne van-e a fában.

3.2. Beszúrás

A művelet segítségével új kulcsot rögzítünk a fában, úgy, hogy a fa továbbra is megtartsa az invariánsokat. Mivel a B+ fában csak a levelekben tároljuk a kulcsokat, ezért legelső feladat a beszúrásnál az, hogy megkeressük azt a levelet, ami a beszúrandó kulcsnak megfelel. Először tehát egy speciális keresést fogunk végrehajtani, ami nem igazzal vagy hamissal tér vissza, hanem azzal a csúccsal (levéllel), amibe be tudjuk szúrni a kulcsot. A beszúrás algoritmus működése nagyon hasonló a kereséshez, de ebben az esetben a megtalált csúccsal tér vissza. Most már tudjuk hova kell beszúrni a kulcsot. Ha van üres hely a kulcsnak szúrjuk be a megfelelő helyre. Ha nincs, vágjuk szét két csúccsá és osszuk el a 4 kulcsot a két csúcs között. Vegyük a második csúcs legkisebb értékét, és szúrjuk be a szülő csúcsba. Ha a szülő csúcs is túllépte a maximális hasító kulcsszámot, akkor ismételjük meg a vágást, mindaddig, amíg a fa nem teljesíti az invariánsokat. A folyamat egészen a gyökér csúcsig is eljuthat. Ilyenkor új gyökeret kell létrehozni és ekkor a fa magassága nő.

3.2.1. Üres fába történő kulcs beszúrás

Ha a fa üres, hozzunk létre egy csúcsot a megadott kulccsal, így létrehozva az első csúcsot, a gyökeret.

3.3. Törlés

A törlés művelettel a fából kulcsokat fogunk eltávolítani, úgy, hogy ne sértsük meg az invariánsokat. Hasonlóan a beszúráshoz, ebben az esetben is egy kereséssel indul az algoritmus, meg kell találnunk azt a csúcsot, amiben a törlendő kulcs található. Ha megtaláltuk a levelet, töröljük a kulcsot a levélből. Természetesen csak olyan kulcsot tudunk eltávolítani a fából, amit levélben tárolunk, direkt *hasító kulcs* törlésére nincs lehetőségünk.

- Ha a levél még tartalmaz elég kulcsot (minimum 2), hogy teljesítse az invariánsokat, akkor kész vagyunk.
- Ha a csúcsban már túl kevés kulcs van, de a baloldali vagy jobboldali testvérben több kulcs van, mint amennyi szükséges, akkor lehetőségünk van kulcsot kérni. Az elkért kulcsot szúrjuk be a csúcsba, majd javítsuk ki a testvérek szülőjében a hasító kulcsokat, hogy megfelelően reprezentálják az új vágási pontokat. Ennek során csak a közös szülőjükben lévő, a testvérekhez tartozó hasító kulcsot kell átírni.
- Ha a csúcsban már túl kevés kulcs van, hogy teljesítse az invariánsokat és a baloldali és jobboldali testvére is minimumon van, akkor egyesítsük egy vele szomszédos testvérével. Ha a csúcs nem levél, akkor be kell vonnunk a szülőből a hasító kulcsot a testvérek egyesítésébe. Levél vagy nem, mindkét esetben meg kell ismételni a törlő algoritmust a szülőre, hogy eltávolítsuk a szülőből a hasító kulcsot, ami eddig elválasztotta, most egyesíti a kulcsokat. Ha a szülő a gyökércsúcs, és az utolsó kulcsa az eltávolítandó hasító kulcs, akkor a most egyesített csúcs lesz az új gyökér, ekkor a fa magassága csökken [3].

3.3.1. Törlés a gyökérből

A gyökérből csak abban az esetben törölhetünk kulcsot, hogyha ez a fa egyetlen levele. Ebben az esetben a minimális kulcs száma 1, csak akkor kell véglegesen törölni a gyökeret, ha az utolsó kulcsát is kitöröltük.

4. Felhasználói dokumentáció

A felhasználói dokumentációban részletezni fogom a program futtatásához szükséges rendszerkövetelményeket, a felhasználó lehetőségeit az alkalmazás használatára, a menük, gombok elhelyezését és jelentését.

4.1. Bevezetés

A program célja megismertetni a felhasználót a B+ fa alapvető műveleteivel, úgy hogy közben folyamatosan nyomon követhetjük a változásokat. A grafikus megjelenés segít megérteni, hogyan változik a fa felépítése az egyes műveletek végrehajtása után.

4.1.1. Rendszerkövetelmény

- Windows 7 vagy újabb operációs rendszer
- Java Runtime Environment 1.8 vagy újabb verzió
- 1024*768 felbontású kijelző vagy nagyobb
- Minimum 1 GB RAM

4.1.2. Program futtatása

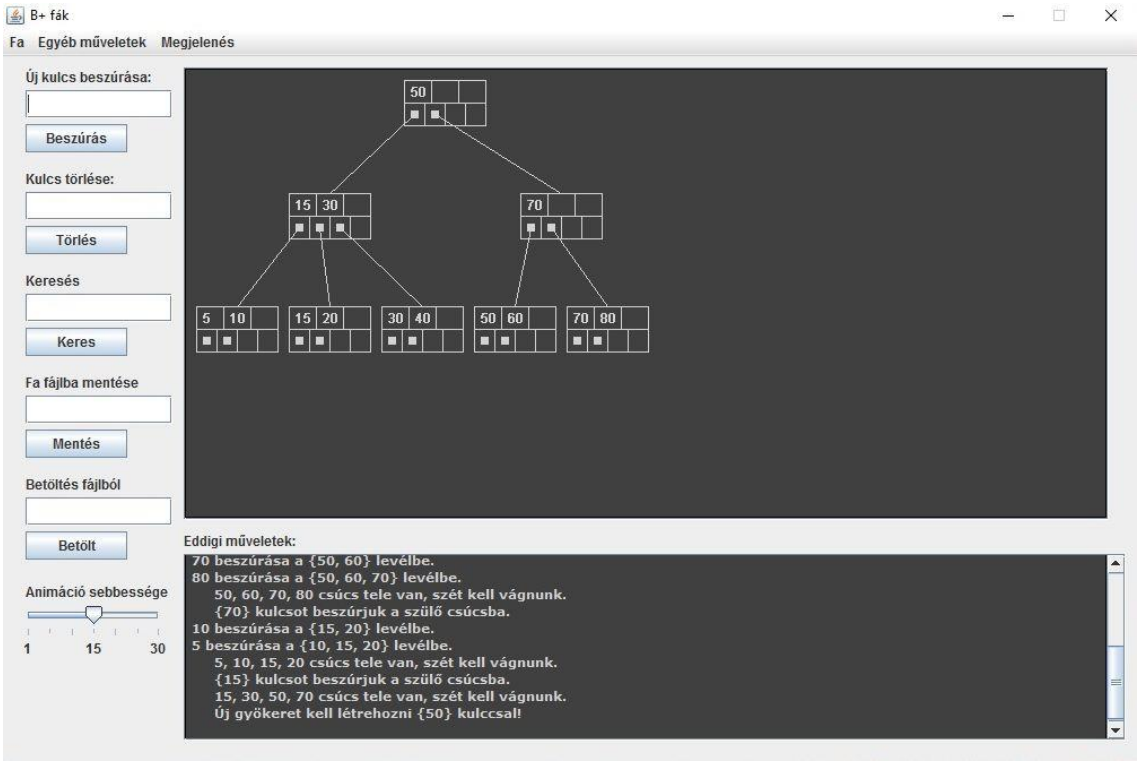
A mellékelt CD-n megtalálható .jar kiterjesztésű fájl futtatásával indítható a program.

4.2. Felhasználási útmutató

A felhasználási útmutatóban bemutatom a programablak felépítését. A különböző menük, gombok, szöveges beviteli felületek elhelyezését és használatát.

4.2.1. Program felülete

A program elindításakor egyből a fő ablak jelenik meg (1. ábra). Három fő részre oszthatjuk a felületet. Bal oldalt a vezérlő elemek találhatok meg, tőle jobbra egy kirajzoló panel, amin a fa fog megjelenni, és a panel alatti üzenetdoboz, amiben a felhasználónak szánt üzenetek, értesítések lesznek olvashatók. Legfelül a menü van, amiben további funkciókat érhetünk el. Az ablak átméretezésére nincs lehetőség.



1. ábra

Programablak

4.2.2. Vezérlőelemek használata

A vezérlőelemek a program bal részén helyezkednek el (2. ábra). A program használata során ezek az egységek letiltásra kerülnek a fa animációja során (3. ábra).

2. ábra

Aktív vezérlőelemek

3. ábra
Letiltott vezérlőelemek

Kulcs beszúrása

Kulcs beszúrására a vezérlő elemek közül a szöveges beviteli mező és a beszúrás gomb használatával van lehetőség. Kétféle bemenetet fogad el a program.

- Egy darab pozitív egész szám megadása, az elfogadható kulcsok halmaza a következőképpen adható meg: $1 \leq k \leq 9999$, ahol k a megadott kulcs. A maximális érték amit a program kezel 9999.
- Beszúrandó kulcsok felsorolása vesszővel és szóközzel elválasztva. Ebben az esetben maximum 20 kulcs adható meg egyszerre, például 10, 105, 130.
- Ha megfelelően adtuk meg a bemenetet, akkor a beszúrás gomb megnyomásával vagy az ENTER billentyű leütésével, a program elhelyezi a megadott értéket, értékeket a fában. Minden vizuális változtatás a kirajzoló panelen fog megjelenni. Ha szükség van animációra, akkor a vezérlő elemek használatát ilyenkor letiltja a program, amíg be nem fejeződik az aktuális folyamat. Az üzenetdobozban megjelennek a beszúrt értékek, és ha módosítás (vágás) történik a fában, akkor további üzentetekkel tájékoztatja a felhasználót. Például: „82 beszúrása a {62, 71} levélbe.”, „10, 20, 30, 40 csúcs tele van, szét kell vágnunk.”, „Új gyökeret kell létrehozni {30} kulccsal!”. Ha az animáció véget ért, akkor a vezérlőelemek újra elérhetők.
- Ha hibás az input vagy olyan kulcsot adtunk meg, ami már megtalálható a fa levelében akkor a program nem módosítja a fát. Kivéve, ha több kulcsot adtunk meg és köztük szerepel olyan, ami már megtalálható a fában. Ekkor a program

beszúrja azokat a kulcsokat, amikre van lehetősége, a többit kihagyja. A különböző hibaüzenetekről a program értesíti a felhasználót. Például: „15 beszúrása sikertelen, mert már szerepel a fa levelében.”, „A kulcs csak 1 és 9999 közötti szám lehet!”.

Kulcs törlése

Kulcs vagy kulcsok törlésénél is hasonlóan kell megadnunk az értéket vagy értékeket, mint a beszúrásnál. A törlés folyamatát a Törlés gomb megnyomásával, vagy az ENTER billentyű leütésével indíthatjuk el.

- Ha megfelelően adtuk meg a bemenetet, akkor a törölni kívánt kulcsot vagy kulcsokat a program megpróbálja eltávolítani a fából. Ha a törlés olyan módosításokat végez a fán, ami változtat a felépítésén, akkor elkezdődik az animáció. Ilyen műveletek lehetnek például a csúcsok összevonása, részfák elkérése. Ekkor a vezérlő elemek használatára szintén nincs lehetőség, mindaddig, amíg a program be nem fejezi az adott feladatot. A művelet vagy műveletek végrehajtása közben, a program folyamatosan tájékoztatja a felhasználót, az üzenetdobozban megjelenő üzenetekkel. Például: „15 törlése a {15, 20} levélből.”, „Jobb testvértől elkérjük a legkisebb kulcsot {44}.”. Ha az animáció befejeződött, akkor a vezérlőelemek újra használhatók.
- Hasonlóan a beszúrásnál hibás kulcs, kulcsok megadása esetén nem történik semmi, a felhasználót értesíti a program erről. Kivéve, ha több kulcsot adtunk meg és köztük szerepel olyan, ami nem található a fában. Ekkor a program törli azokat a kulcsokat, amikre van lehetősége, a többit kihagyja. A program értesíti a felhasználót, ha hibás kulcsot adott meg. Például: „A kulcs csak 1 és 9999 közötti szám lehet!”, „1 törlése sikertelen, mert nem található a levelekben.”.

Keresés

Kulcs keresésénél egyszerre egy elemre tudunk rákeresni. A megadott elemnek a beszúrásban definiált formátumot kell követnie. A folyamatot a Keresés gomb megnyomásával vagy az ENTER billentyűzet leütésével indíthatjuk el. Ilyenkor animáció nem történik, ha a keresett kulcs megtalálható a fában, akkor a felhasználót a program az üzenetdobozban megjelenő szöveggel értesíti, hogy a keresés sikeres: „1 megtalálható a {1, 54, 66} levélben.”. Hibás bemenet esetén, vagy ha a fában nem található meg a kulcs, ezt is tudatja a felhasználóval: „1 nem található meg a fa leveleiben.”.

Fa mentése fájlba

A program lehetőséget ad a felhasználónak a fa adott állapotának mentésére. A beviteli mező egy fájlnevet vár. A helyes fájlnévre a következő szabályok érvényesek:

- csak ékezet nélküli kisbetűkből és számokból állhat
- kisbetűvel kell kezdődnie
- minimum 1 karakter hosszú

A Mentés gombra kattintva, az alkalmazás gyökér mappájába két fájl fog megjelenni. A .txt kiterjesztésű fájl a fa zárójelezett alakját tartalmazza. A .dat típusú fájlban, pedig az eddigi műveletek listája található meg, ez a program részére lesz hasznos, hogy a fa betöltése során vissza tudja építeni a fát (lásd lentebb).

Fa betöltése fájlból

Fa betöltésekor a beviteli mező szintén egy fájlnevet vár. Ebben az esetben kicsit módosulnak a helyes bemenet szabályai:

- csak ékezet nélküli kisbetűkből és számokból állhat
- kisbetűvel kell kezdődnie
- „.dat”-ra kell végződnie

Ha helyesen adtuk meg a fájlnevet és létezik, akkor a program felépíti a fát, a fájlból beolvasott utasítássorozat alapján. Ilyenkor az animáció sebessége 30 lesz és a vezérlő elemek használatára nem lesz lehetőség, amíg a folyamat be nem fejeződik. Ha helytelenül adtuk meg a fájlnevet vagy nem létezik, akkor a program tájékoztatja a felhasználót,

Animáció sebessége

A program fő feladata, hogy kirajzolja a fát, a felhasználó által végzett műveletek alapján. Ezek a folyamatok változtathatják a fa felépítését, ezért fontos, hogy a felhasználó nyomon tudja követni a műveletek kimenetelét. Az animáció sebesség változtatására a vezérlő elemek utolsó, legalsó részén van lehetőség. Egy csúszka segítségével 1-30-as skálán lehet változatni a kirajzolás sebességét. A program egyes műveletek folyamán, például *feltöltés véletlen számokkal* (lásd lentebb), vagy a *fa beolvasása fájlból* folyamat során változtathatja az animáció gyorsaságát. Mindkét esetben a felhasználót a program tájékoztatja, az éppen aktuális sebességről az üzenetdobozban: „Animáció sebessége: 20”.

4.2.3. Menü használata

A program menüje az ablak felső részében található. Itt olyan funkciók érhetők el, amik nem tartoznak a vezérlő elemekhez, de hasznosak lehetnek. A menüsáv három menüt tartalmaz, amikben további almenük találhatók.

Fa menü

A fa menü tartalmazza azokat a lehetőseket, amikkel gyors műveleteket végezhetünk a fán, három ilyen művelet érhető el: *új fa létrehozása, feltöltés véletlenszerű számokkal* és a fa *zárójelezett alakja*.

- **Új fa létrehozása**

Ebben az almenüben a felhasználónak lehetősége van új fát létrehozni. Ilyenkor a kirajzoló panelen eltűnik az eddig megjelenített fa, és minden eddigi művelet törlésre kerül visszavonhatatlanul.

- **Feltöltés véletlenszerű számokkal**

A fához tartozó második almenüre kattintva véletlenszerű számokkal tudjuk feltölteni a fát. Ekkor 20 darab véletlenszerű szám beszúrásra kerül. A folyamat előtt az animáció sebessége 30 (maximális) lesz, hogy a művelet minél hamarabb befejeződjön. A 20 szám egymástól különböző, de nincs garantálva, hogy mindegyik érték, olyan, ami még nem szerepel a fában. Animáció közben, nincs lehetőség az almenüre kattintani, mindaddig, amíg az adott folyamat be nem fejeződik.

- **Zárójelezett alak**

Az utolsó almenü pedig a zárójelezett alak kiírására szolgál. Rákattintva az üzenetdobozban megjelenik a fa zárójelezett alakja.

Egyéb műveletek menü

- **Random számok (beszúráshoz)**

Ez a menü arra szolgál, hogy ha gyakorolni szeretnénk a fa beszúrás műveletét, akkor generálhatunk véletlenszerű (random) számokat. Az almenüre kattintva 20 darab különböző szám fog megjelenni az üzenetdobozban. Ebben az esetben sincs garantálva, hogy a fa nem tartalmazza a felsorolt számokat.

- **Random számok (törléshez)**

Ebben az esetben 5 darab véletlenszerű szám fog megjelenni az üzenetdobozban. Ha a fa nem tartalmaz 5 kulcsot, akkor az összes kulcs felsorolásra kerül. Mindegyik szám különböző és szerepel a fában, így lehetőséget adva a felhasználónak a törlés művelet gyakorlására.

- **Üzenetdoboz törlése**

Az utolsó almenüre kattintva az üzenetdobozt tudjuk kitörölni.

Megjelenés menü

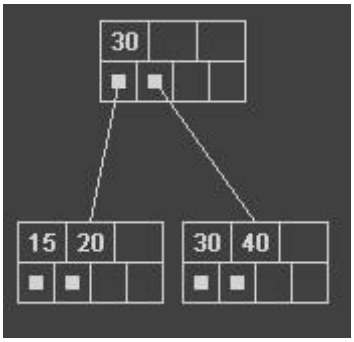
Az utolsó menüben két almenü található, amivel a kirajzoló panel és az üzenetdoboz megjelenését változtathatjuk meg. A *világosra* kattintva a panel és az üzenetdoboz háttérszíne fehér lesz. A fa megjelenése és az értesítések színe feketére változik. A *sötét* almenüre kattintva, pedig a panel és az üzenetdoboz háttere sötétszürke, a fa megjelenése és az üzenetdoboz betűszíne pedig világosszürke lesz.

4.2.4. Kirajzoló panel

A kirajzoló panel foglalja el az programablak jelentős részét, erre rajzolja ki a program a fát. Ezen a felületen fog megjelenni a fa aktuális állapota.

Csúcsok megjelenítése

A csúcs egy téglalapként jelenik meg, amely a felénél ketté van osztva. A felső részben három egyenlő méretű téglalapban a kulcsok találhatók. Az alsó részben, pedig négy darab, szintén egyenlő méretű téglalap foglal, helyet, amikben a pointermezők vannak. A pointermezőket négy kisebb, kitöltött négyzetként ábrázoljuk, ezek lesznek a pointerek kiindulási pontjai. Ha a csúcs levél, vagy olyan belső csúcs aminek van szülője, akkor egy vonallal összekötjük őket. A vonal a szülő adott pointermezőjét és a megfelelő gyerek csúcsot köti össze (4. ábra). Mivel a fa elég hamar szélesedik, ezért ha elérte a megjelenített grafika a panel szélét, akkor egy vízszintes görgetősáv segítségével lehet tovább görgetni a képernyőn. Ha a fa mérete meghaladja a panel alsó részét, akkor egy függőleges görgetősáv jelenik meg.



5. ábra

Csúcsok felépítése

Animáció

Műveletektől függően a program animálva jeleníti meg a fa egyes állapotátmeneteit. A vezérlőelemek legutolsó komponensével állíthatunk a kívánt sebességen, ekkor az animálás lassabban vagy gyorsabban fog megjelenni. Bármennyire is törekedtem, nem minden átmenet látható pontosan, ilyen műveletek például törlésnél a csúcsok összevonása. Viszont a beszúrásnál a csúcsok vágása jól látható és könnyen nyomon követhetők a változások, főleg ha lassúra állítjuk a kirajzolás sebességét. A vonalak kirajzolása folyamatos, a fa növekedése és csökkenésénél, valamint a vágások, összevonások alkalmával is. A szülő csúcs mindig a legszélső és a legutolsó gyereke között helyezkedik el. A levelek egymástól egyenlő távolságra vannak, hogy minél kevesebb helyet foglaljanak el. A csúcsok szélessége folyamatosan változik, mindig az adott csúcs legnagyobb kulcsától függően.

Üzenetdoboz

Az üzenetdoboz a programablak legalsó részén a kirajzoló panel alatt található. Ebben jelennek meg a felhasználó számára legfontosabb értesítések, ami a fa állapotára vagy egyéb módosításokra vonatkozik (lásd fentebb).

5. Fejlesztői dokumentáció

Ebben a fejezetben fogom részletezni a program szerkezetét, felépítését a választott technológiákat. A programban egy olyan adatszerkezetet kellett létrehoznom, amivel sikeresen tudom reprezentálni a B+ fa tulajdonságait, műveleteit. Kirajzolásához pedig olyan grafikai megvalósítást kellett alkalmaznom, hogy nyomon követhetőek legyenek a felhasználó számára a műveletek.

5.1. Választott technológia

A választott programozási nyelv a Java, a fejlesztői környezet pedig NetBeans IDE 8.0.2 . A döntés mellett több érv is szól. Ezt a nyelvet sikerült egyetemi tanulmányaim alatt a legjobban megismerni, elsajátítani. A Java egy magas szintű objektumorientált nyelv, amiben lehetőség van jól strukturált programot írni, elkülönítve a modell és a nézet réteget, felhasználva az objektumorientáltságból származó előnyöket és a grafikai lehetőségeit a nyelvnek.

5.2. Tervezés

A program tervezésekor elsősorban az adatszerkezettel kezdtem. Ehhez olyan osztályokra van szükség, amiben tárolni lehet a fa adatait, csúcsait, végre lehet rajta hajtani műveleteket. A modell csomag tartalmazza ezeket a fájlokat. A program másik fő része a nézet csomag, ami a fa kirajzolásáért fog felelni. Benne található egy csúcs felépítése és a megjelenítéséhez tartozó legfontosabb adatok, műveletek. A programban tehát két fő rész kommunikál egymás között, a modell változásait a nézet segítségével jelenítem meg.

5.2.1. Csúcsok adatainak, műveleteinek reprezentálása

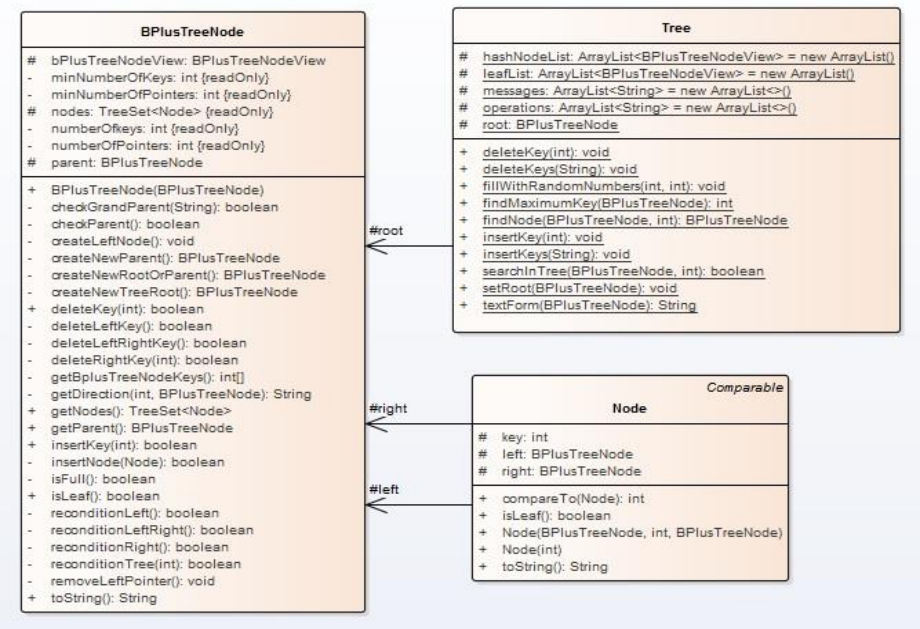
A B+ fa csúcsinak meg kell, hogy feleljenek a **2.2.** pontban definiált invariánsoknak. Olyan adatszerkezet kell, ami rendezhető és könnyen lehet műveleteket végezni rajta. Minden belső csúcsnak ismernie kell a gyerekeit és a szülőjét. A levelekben minden beszúrt értéknek meg kell jelennie, rendezetten, úgy hogy az invariánsokat teljesítse és ismernie kell a szülőjét. Mivel a program csak a B+ fa szemléltetésére szolgál, ezért a levelekben további adatokat nem tárol. Figyelembe kell venni, hogy melyik csúcsnak lehet bal és jobb pointere. Minden belső csúcs első hasító kulcsának van baloldali gyereke, amiben tőle szigorúan kisebb értékek találhatók meg. Jobboldali gyereke mindegyik hasító kulcsnak van, melyekben a tőle nagyobb-egyenlő értékek vannak. Ezek az invariánsok kirajzolásnál is fontos szerepet kapnak. A csúcsokat úgy kell megjeleníteni

egy kirajzoló panelen, hogy a kulcsok olvashatóan jelenjenek meg. Ehhez úgy kell igazítani a csúcs méretét, hogy a legnagyobb benne található kulcs se „lógjon” ki. Minden szülő pointermezőjét a megfelelő gyerekkel kell összekötni egy vonallal, hogy látható legyen a köztük levő kapcsolat. A műveletek végrehajtásakor szemléletesen kell animálni a fa átmeneteit. Egy rövid szöveges leírás is társul az animáció mellé, hogy még érhetőbb és nyomon követhetőbb legyen az adott művelet. Ehhez egy szöveges felületet kell biztosítani. Továbbá lehetőséget kell adni a felhasználónak újrakezdésre, a fa tetszőleges állapotának kimentésére és betöltésére.

5.3. A program felépítése

5.3.1. A model csomag osztályai

A model csomagban találhatók azok az osztályok, amelyek a fa adatait és megjelenítési paramétereit tartalmazzák. A fa főbb metódusait és gyökér csúcsát a `Tree` osztály tartalmazza. Egy B+ fa csúcsa (`BPlusTreeNode`) a benne található kulcsok és pointerekből épül fel. A programban egy külön osztályt (`Node`) hoztam létre a kulcsok és a két pointer tárolására. A programban úgy definiáltam a hasító kulcsok mutatóját, hogy csak az első hasító kulcs ismeri a bal és jobb oldali gyerekét, a további hasító kulcsok csak a jobb oldali gyereküket ismerik. A továbbiakban csúcs alatt egy `BPlusTreeNode` objektumot értek. Az alábbi képen az osztálydiagram található, amin az osztályok közötti kapcsolatokat tüntetem fel (6. ábra).



6. ábra

Osztálydiagram

Node osztály

A `Node` osztály 3 adattagot tartalmaz:

- `key` – az osztályban tárolt kulcs (`int`)
- `left` – baloldali gyerek (`BPlusTreeNode`)
- `right` – jobboldali gyerek (`BPlusTreeNode`)

A kulcsok `int` vagyis egész szám, a gyerekek pedig `BPlusTreeNode` típusúak. Mivel a program csak 1 és 9999 közötti értékeket enged meg, ezért a kulcs egy pozitív egész szám. Ahhoz, hogy a későbbiekben rendezetten tudjuk tárolni a kulcsokat, az osztálynak meg kell valósítania a `Comparable<Node>` interfészt, `Node` típusparaméterrel. Ehhez felül kell definiálnunk a `int compareTo(Node node)` metódust és magunknak kell megadni, hogy milyen feltételek szerint szeretnénk rendezni a `Node` típusú elemeket. Mivel a B+ fa definíciójában, már lefektettük, hogy szigorúan monoton növekvően kell tárolni az értékeket, ezért úgy módosítottam a függvényt, hogy megfeleljen a kritériumnak. Az objektum példányosítására két konstruktort készítettem, egyikben csak a kulcsot adjuk meg, és a gyerekeket nem, ekkor a `left` és a `right` értéke `null` lesz. A másik konstruktorban meg kell adni a két gyereket és a kulcsot is. Getter és Setter metódusokra nincs szükség, minden adattagot `protected` kulcsszóval láttam el, így a csomag további osztályai hozzáférnek az adattagokhoz. A rendezhetőség megvalósításán kívül, egy fontosabb metódusa van az osztálynak. Az `boolean isLeaf()` logikai visszatérésű függvény, amely eldönti egy adott csúcsról, hogy levél-e. Ez azt jelenti, hogy a `left` és a `right` adattagoknak `null` az értéke.

BPlusTreeNode osztály

A `BPlusTreeNode` osztály tartalmazza a B+ fa csúcsainak felépítését és műveleteit. Egy objektum fontosabb adattagjai:

- `nodes` – a fában tárolt `Node`-ok listája `(TreeSet<Node>)`
- `parent` – a csúcs szülője `(BPlusTreeNode)`
- `bPlusTreeNodeView` – a csúcs megjelenítéséhez tartozó objektum, ami a későbbiekben lesz fontos `(BPlusTreeNodeView)`

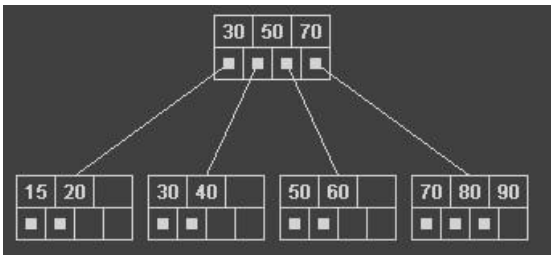
Az osztályban találhatók a beszúrás és a törlés műveletekhez tartozó metódusok. A következőkben ezeket fejtem ki.

Beszúrás megvalósítása

A beszúrás a beszúrandó kulcs helyének megkeresésével kezdődik. A beszúrásra csak akkor kerül sor, ha a fa még nem tartalmazza a kulcsot. A `Tree` osztályban definiált `static BPlusTreeNode findNode(int key)` osztályszintű metódus megkeresi a megadott kulcsnak megfelelő levelet. Az keresés algoritmusát a későbbiekben ismertetetem (lásd `Tree` osztály). Ha megvan a levél, akkor beszúrjuk a kulcsot, a megfelelő helyre, úgy, hogy létrehozunk egy `Node` objektumot a megadott kulccsal és hozzáadjuk a levél `nodes` listájához. Ezt az `boolean insertKey(int key)` metódus levélre való meghívásával történik. A `TreeSet`-ben rendezetten tároljuk a `Node` objektumokat, ezért a beszúrandó kulcs a megfelelő helyre fog kerülni. Ha a beszúrás után a levélben tárolt kulcsok száma (`nodes` mérete) kisebb-egyenlő, mint három, akkor készen vagyunk.

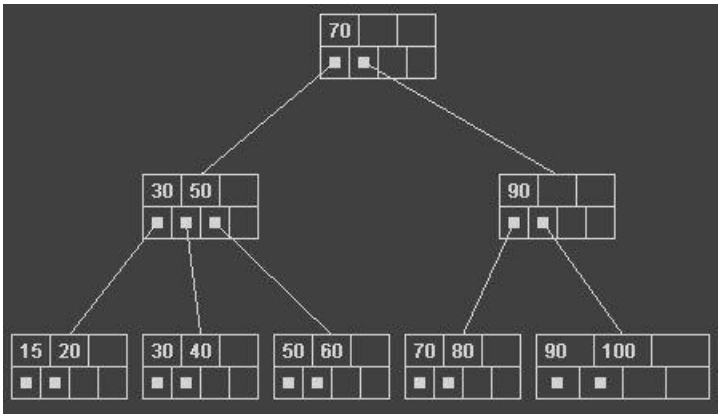
Csúcs vágása

Ha a csúcsban tárolt kulcsok vagy hasító kulcsok száma túllépte a maximális kulcsszámot (`nodes` mérete nagyobb, mint 3), akkor vágásra van szükség. Minden vágás levélszinten kezdődik, mert először a levelekben alakulhat ki túlcsordulás. A vágás során bontsuk szét a levelet két csúccsá és osszuk el egyenlően a két csúcs között a 4 kulcsot. Az első két kulcs a baloldali csúcsba kerül, az utolsó kettő a jobboldaliba. Majd vegyük a jobboldali csúcs legkisebb érték másolatát és szúrjuk be a szülőbe. Ha nincs még szülő csúcs, hozzuk létre. Ha szülő csúcs lépte túl a maximális kulcsszámot, vegyük ki a középső kulcsot a kulcsok elosztása során és szúrjuk be a szülőbe. Mivel a programban a vágások során nem kulcsokat, hanem `Node`-okat helyezünk át a különböző csúcsokba, minden művelet folyamán figyelni kell arra, hogy az egyes `Node` objektumok `left` és `right` adattagja a megfelelő gyereket tartalmazza. A szülő csúcsoknál is hasonlóan kell eljárni, ha a szülő csúcsba új `Node`-ot szúrtunk, akkor be kell állítani a pointereit, hogy a megfelelő gyerekekre mutassanak. Az alábbi két ábrán (7-8. ábra) vágásokra láthatunk példát. Először a 70, 80, 90 kulcsokat tartalmazó levelet kell szétvágni. Ennek során a 90-es kulcs bekerül a szülő csúcsba. Mivel a szülő is túllépte a megengedett hasító kulcsszámot, ezért azt is szét kell vágni, ezzel új gyökeret létrehozva.



7. ábra

100 beszúrása előtt a fa állapota



8. ábra

100 beszúrása után

Beszúrásnál alkalmazott további metódusok és rövid leírásuk

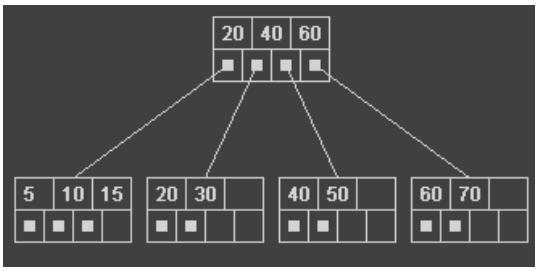
- `isFull()` – eldönti, hogy az adott csúcs túllépte-e a megengedett kulcsok számát
- `isleaf()` – eldönti, hogy az adott csúcs levél-e
- `checkParent()` – ellenőrzi a szülő csúcsot, és ha tele van, az alábbi két művelet közül a megfelelőt választja:
    - `createNewTreeRoot()` – a vágás folyamán új gyökeret hoz létre és beállítja a pointereket
    - `createNewParent()` – a vágás után beszúrja az új szülő hasító kulcsot a szülő csúcsba (`BPlusTreeNode`) és beállítja a pointereket

Törlés megvalósítása

A törlés algoritmusa a megadott kulcs levelének megkeresésével kezdődik. Törlésre csak akkor kerül sor, ha a megtalált levél tartalmazza a kulcsot. A megkeresett levélre meghívjuk a `boolean deleteKey(int key)` metódust, ami eltávolítja a csúcs `nodes` listájából a megadott kulccsal szereplő `Node` objektumot. Ha a levél még tartalmaz elég kulcsot, azaz legalább kettőt, akkor készen vagyunk.

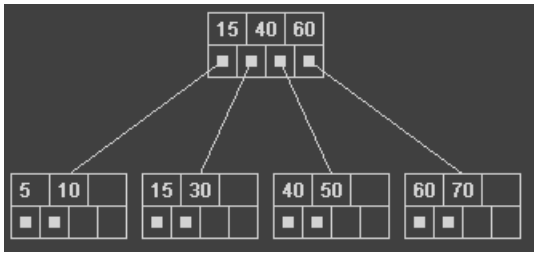
Testvértől kulcs vagy hasító kulcs elkérése

Ha a törlés után túl kevés kulcs maradt a levélben, de a megelőző vagy következő testvérének több van, mint amennyi szükséges, akkor a törlés algoritmusa mindig a baloldal legnagyobb (utolsó) kulcsát kéri el. Ha a baloldal minimumon kulcsszámmal rendelkezik, akkor a jobboldaltól kéri el a legkisebb (első) kulcsot. Mindkét esetben javítani kell a szülőben a hasító kulcsot. A 9-10. ábrán egy olyan törlés látható, aminek során a baloldali testvértől kérünk kulcsot. A 20-as kulcsot kitörüljük a levélből majd a megmarad 30-as kulcs mellé elkérjük a 15-ös értéket. Végül a szülőben átírjuk a hasító kulcsot 15-re.



9. ábra

20 kulcs törlése előtti állapot

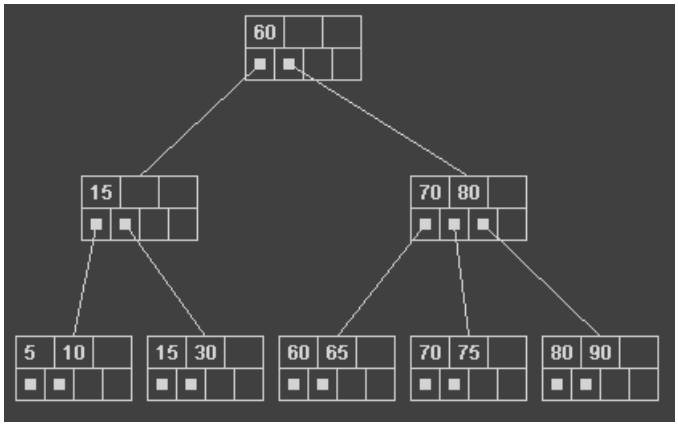


10. ábra

20 kulcs törlése utáni állapot

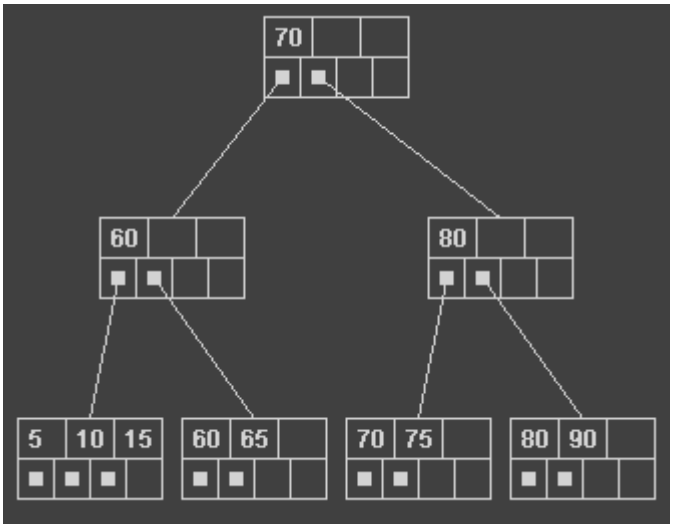
Ha nem tudunk kulcsot kérni, akkor összevonjuk a szülő maradék gyerekeit egy közös csúcsba. Majd a szülő bal és jobb testvéreitől próbálunk hasító kulcsot és a hozzá tartozó megfelelő részfát elkérni. Ezt csak akkor tehetjük meg, ha az említett két testvér közül legalább az egyik a minimálisnál több hasító kulccsal rendelkezik. Ebben az esetben is először a szülő baloldali testvérétől kérjük el a legnagyobb hasító kulcsot és a hozzá tartozó jobb részfát. Ha erre nincs lehetőség, mert minimális kulcsszámmal rendelkezik, akkor a jobboldali testvér legkisebb hasító kulcsát kérjük el és a hozzá tartozó bal részfát. Mindkét esetben javítani kell az érintett szülőkben és a nagyszülőben a hasító kulcsokat és a hozzájuk tartozó pointereket, hogy a művelet után a fa teljesítse az invariánsokat.

A következő két képen (11-12. ábra) egy levél elkérése látható. A 30-as kulcs törlésével a baloldali részfából, már nem marad a szülőnek jobboldali gyereke. Ekkor a maradék kulcsokat 5, 10, 15 egy levélbe kerülnek, majd az algoritmus elkéri a szülő jobb testvérének baloldali gyerekét. A nagyszülőből lekerül a hasító kulcs a szülőbe. A szülő jobboldali testvéréből pedig felkerül a nagyszülőbe a hasító kulcs.



11. ábra

30 törlése előtti állapot



12. ábra

30 törlése utáni állapot

Levelek, szülők összevonása

Abban az esetben, amikor nem tudunk kulcsot kérni a testvér levelektől, akkor a megmaradt kulcsot az algoritmus beszúrja a bal vagy jobb testvér levelébe. Erre csak akkor van lehetőség, ha a szülő a minimumnál több hasító kulccsal rendelkezik. Az algoritmus mindig a jobboldali testvérbe szúrja be a megmaradt kulcsot. Ha nincs

jobboldali testvér, akkor a baloldali testvérbe helyezzük át a megmaradt kulcsot. Mindkét esetben törölni kell az üres levelet és a hozzá tartozó hasító kulcsot a szülőből. Ezt a műveletet levelek összevonásának nevezzük.

Szülők összevonására akkor van szükség, ha nem tudunk a testvérektől hasító kulcsot kérni. Két lehetséges eset van.
Az első esetben, a nagyszülőnek a minimumnál több hasító kulcsa van. Ekkor az algoritmus először mindig a jobboldali testvérrel vonja össze a szülőt. Ha nincs jobboldali testvér, akkor a baloldalival. Mindkét esetben be kell vonni a nagyszülő hasító kulcsát a szülők egyesítésébe. Ekkor a nagyszülő hasító kulcsa lekerül az egyesített szülőbe.
A másik esetben a nagyszülő minimum kulcsszámmal rendelkezik. Ekkor a nagyszülőbe vonjuk össze a gyerekeit. Ha az így keletkezett összevont nagyszülő a fa gyökere, akkor a fa magassága csökken, különben folytatódik művelet további összevonásokkal vagy hasító kulcsok elkérésével, mindaddig amíg a fa nem teljesíti az invariánsokat.

Törlésnél alkalmazott további metódusok és rövid leírásuk

- `deleteLeftKey(),deleteLeftRightKey(),deleteRightKey(int key)` – a levélszinten való kulcsok elkéréséért, összevonásáért felelnek
- `reconditionLeft(),reconditionLeftRight(),reconditionRight()` – levél csúcsok elkéréséért, összevonásáért felelnek
- `reconditionTree()` – részfák elkéréséért, összevonásáért felel a metódus

Tree osztály

Ez az osztály felel a fára vonatkozó egyéb műveletekért. Példányosítani nem kell, statikus adattagokkal és metódusokkal dolgozok. Az osztály a következő osztályszintű adattagokat tartalmazza:

- `root` – a fa gyökere (`BplusTreeNode`)
- `leafList` – leveleket tartalmazó lista, a kirajzoláshoz szükséges (`ArrayList<BPlusTreeNodeView>`)
- `hashNodeList` – a belső csúcsokat tartalmazó lista, ez is a kirajzoláshoz szükséges (`ArrayList<BPlusTreeNodeView>`)
- `messages` – üzenetek tartalmazza a felhasználó részére (`ArrayList<String>`)
- `operations` – a fa eddigi műveleteit tartalmazza, fájlba íráshoz szükséges (`ArrayList<String>`)

A fa gyökerét nem lenne fontos tárolni, hiszen a csúcsok bejárásával egyértelműen meghatározható melyik a gyökér, de az egyszerűség kedvéért jobbnak találtam mégis egy kiemelt változóban eltárolni. Az `operations` és a `messages` listák szöveget tartalmaznak, amik a fájlba íráshoz és a felhasználó számára fontosak. Ahogy említettem már, a beszúrás és a törlés is egy kereséssel kezdődik, ami a fa gyökeréből indul ki és egy levelet fog visszaadni. A keresés algoritmusa az alábbi:

```
static BPlusTreeNode findNode(BPlusTreeNode node, int key) {
      if (node.isLeaf()) {
            return node;
      }
      if (node.nodes.first().key > key) {
            return findNode(node.nodes.first().left, key);
      }
      return findNode(node.nodes.floor(new Node(key)).right, key);
 }
```

Az algoritmus egy csúcsot és egy kulcsot vár paraméterül. Ameddig nem talál levél csúcsot, addig a megfelelő részfákon folytatja a keresést a gyökérből kiindulva. Ha megtalálta a levelet, akkor visszatérési értéke a csúcs lesz.

Az osztályban megvalósított további metódusok leírása

- `void insertKey(int key)`

A metódus megkeresi a megfelelő levelet, és beszúrja a csúcsba a kulcsot

- `void deleteKey(int key)`

A metódus szintén megkeresi a megfelelő levelet és törli a kulcsot belőle

- `void insertKeys(String numbers)`

A paraméterben kapott szöveget (számsorozat vesszőkkel elválasztva) a metódus feldolgozza, ennek során létrehoz egy számokat tartalmazó tömböt. Ezen a tömbön végigiterálva minden kulcsra meghívja a beszúrás metódusát.

- `void deleteKeys(String numbers)`

Hasonlóan az előző metódushoz, ez is egy szöveget kap paraméterül, ugyanúgy létrehozza a tömböt. Majd végigiterálva a tömbön minden kulcsra meghívja a törlés metódusát.

5.3.2. A view csomag osztályai

A view csomagban található a programablak felépítése, a kirajzoló panel és a csúcsok rajzolásához tartozó információk.

MainFrame osztály

Ebben az osztályban hozom létre a fő programablakot. Az egyes komponenseket példányosítom, és elhelyezem a felületen. Minden, a panelen található objektumot a `javax.swing` csomagból importálom és használom fel. A felhasználó tevékenységét `ActionListener`-ek segítségével követem. Ilyen tevékenységek lehetnek, gombok megnyomása, az ENTER billentyű leütése, vagy menükre kattintás. A főbb komponenseket már leírtam a felhasználói dokumentációban, (**4.1.** fejezet) ezek találhatók a panelen.

Point osztály

A `Point` osztályban definiálom a csúcsok elhelyezkedését meghatározó x, y koordinátákat. Ezzel adom meg a csúcsok pozícióját a panelen. Egy pont példányosításához egy konstruktort hoztam létre, ami x, y szám (`int`) típusú értéket vár. Az `Object` ősosztályból származó `equals` függvényt felüldefiniáltam, hogy megadhassam, mikor egyenlő két `Point` típusú objektum. Ez akkor lesz igaz, ha két objektum x és y adattagja megegyezik.

BplusTreeNodeLine osztály

A csúcsok összekötéséhez meg kell adni egy kiinduló pontot és egy végpontot. Az objektumban tárolt két adattag – `from` és `to` – `Point` típusúak. A konstruktor is két `Point` objektumot vár és beállítja a két objektumszintű adattagot. A vonalak elhelyezkedését két csúcs között már kifejtettem (**5.2.1.** fejezet).

BplusTreeNodeView osztály

A csúcsok fő megjelenítési adatait ez az osztály tartalmazza. Fontosabb adattagjai:

- `keyRectangles` – kulcsok téglalapjainak koordinátái `(ArrayList<Point>)`
- `pointerRectangles` – pointerek téglalapjainak koordinátái `(ArrayList<Point>)`
- `pointerInnerRectangles` – pointermezők belső téglalapjainak koordinátái `(ArrayList<Point>)`
- `keyCoordinates` – kulcsok elhelyezkedésének adatai `(HashMap<Point, String>)`
- `origin` – a csúcs kezdőpontja, ami csúcs a jobb felső sarja lesz `(Point)`
- `keys` – a csúcsban tárolt kulcsok tömbje `(int[])`
- `line` – a csúcsból húzott vonal koordinátái `(BPlusTreeNodeLine)`

- `parent` – a csúcs szülőjének a nézet objektuma `(BplusTreeNodeView)`
- `fullWidth` a csúcs teljes szélessége `(int)`
- `destination` – a csúcs érkezési helye `(Point)`

Legfontosabb feladata az osztály egy objektumának az, hogy tárolja a csúcsban a kulcsok listáját és a csúcs felépítéséhez szükséges minden pont elhelyezkedését. Ez a kirajzoló panel számára lesz fontos, hiszen minden csúcs nézet objektuma a `Tree` osztály `leafList` vagy `hashNodeList` adattagjába van, a listák segítségével rajzolom ki a fát a kirajzoló panelen. Az említett két listában, ahhoz, hogy rendezve tudjam tárolni az elemeket, az osztály megvalósítja a `Comparable<BPlusTreeNodeView>` interfészt, ebben az esetben is felüldefiniáltam az `int compareTo(…)` metódust. A konstruktor vár egy rendezett szám tömböt, ami tartalmazza a csúcsban tárolt kulcsokat, és két koordinátát x, y (`int`) ezek alkotják a csúcs kezdőpontját. A konstruktorral létrehozzuk az üres listákat és feltöltjük pontokkal, hogy a kirajzoló panel meg tudja jeleníteni. Említettem, hogy a csúcs szélessége az adott csúcs legnagyobb kulcsától függ.
Ez mindig a paraméterül kapott tömb utolsó eleme lesz. Ezt az elemet szöveggé alakítva lekérdezzük a hosszát és ezt az értéket megszorozzuk 12-vel. Ha a kulcs 1 számjegyű, akkor ez a szám 20 lesz. Fontos, hogy a csúcs teljes hossza (`fullWidth`) osztató legyen hárommal, mert a csúcsban a kulcsok elhelyezkedését szolgáló felső részt 3 egyenlő téglalapra kell osztani. Az is fontos, hogy a teljes hossz osztható legyen néggyel, mert a pointereknek fenntartott alsó téglalapot is 4 egyenlő részre kell osztani. A kulcsok, pointerek, téglalapjait listába tároljuk, viszont a karakterek elhelyezkedését `HashMap` adatszerkezetben kell tárolni. Erre azért van szükség, mert minden különböző ponthoz, különböző szám tartozik. Ha a `destination` változó nem `null`, akkor az azt jelenti, hogy a csúcsot mozgatni kell, egészen addig, amíg az kezdőpontja meg nem egyezik az új érkezési helyével.
Az osztályhoz tartozik egy `setView(int[] keys, int x, int y)` metódus is, ami hasonlóan működik a konstruktorhoz, beállítja a már létező objektum adattagjait a paraméterben kapott értékek szerint. Erre akkor lehet szükség, ha mozgatni kell a csúcsokat a `moveNode()` metódussal.

TreePanel osztály

A `TreePanel` osztály kiterjeszti (`extends`) a `JPanel` osztályt. Ez által hozzáférünk a `JPanel` nyújtotta lehetőségekhez például metódusok felüldefiniálása, adattagok elérésére. Az osztály fő feladata kirajzolni a fa aktuális állapotát a

`BPlusTreeNodeView`-ban tárolt adatok alapján. Ehhez be kell állítani egy `Timer` típusú objektumot, aminek segítségével ellenőrizni tudom, hogy történt-e változás a fában. Ehhez implementálni kell az `ActionListener` interfészt és felül kell definiálni a `void actionPerformed(ActionEvent ae)` metódust. Ebben a metódusban történik a fa ellenőrzése megadott időközönként. Ha történt változás, akkor végig iterálunk a következő két listán: `leafList` és `hashNodeList` és a bennük tárolt `BplusTreeNodeView` objektumok alapján újra kirajzoljuk a panelre az objektumokat, frissítve a fa állapotát.

5.3.3. A main csomag osztálya

A main csomagban csak egy fájl található a `Main.java`. Ez a futtatható osztálya az egész projektnek. Elindítja a programot a `MainFrame` osztály példányosításával.

5.3.4. Tesztelés

A kód szintaktikai helyességét a fejlesztői környezet ellenőrizte. A tesztelést két fő csoportra lehet bontani. Egyik a fa logika felépítésének tesztelése, a másik a grafikai kirajzolása. A program fejlesztését a model csomagban található osztályokkal kezdtem és folyamatosan ellenőriztem, hogy az adatokat megfelelően tárolják az objektumok. Ahogy a fát minimálisan ki tudtam rajzolni leegyszerűsödött ez a feladat. Végül a különböző animációkat, a fa grafikai megjelenését is tesztelni kellett. A következőkben a fekete-doboz tesztelést használom, amivel nem vagyok „tekintettel” a program felépítésére, csak azt ellenőrzöm, hogy a megadott bemenetre a program milyen kimenetet eredményez. Ehhez egy táblázatot fogok használni, ami a következőket tartalmazza. A bemenetet és a művelet végrehajtásának elvárt eredményét egy rövid szöveges leírással. Mellette pedig a program kimenetét, vagyis a teszt eredményét.

Beszúrás tesztelése

Először a beszúrásra vonatkozó teszteseteket fogom leírni. Ebben az esetben fontos a bemenet helyessége és a műveletek végrehajtása (vágás).

Teszteset rövid leírása	**Művelet eredménye**
Üres fába történő 1 darab kulcs beszúrása. Új gyökér létrehozása.	Sikeres
0 vagy negatív szám beszúrása. Nem történik semmi a fában.	Sikeres

Kulcs beszúrása minimális elemszámú levélbe. Kulcs megjelenik a megfelelő helyen, rendezetten.	Sikeres
Kulcs beszúrása egyszintű fába, ami vágást eredményez. Új gyökér létrehozása, kulcsok megfelelő elhelyezése a létrehozott levelekben.	Sikeres
Maximális kulcsszámú levélbe történő beszúrás. Csúcs szétvágása, szülő létrehozása.	Sikeres
Több kulcs beszúrása, megfelelő formátumban. Megfelelő műveletek elvégzése.	Sikeres
Több kulcs megadása rossz formátumban. Nem történik semmi.	Sikeres

Ezeket a főbb teszteseteket hajtottam végre a beszúrás teszteléséhez.

Törlés tesztelése

A törlésnél sokkal több teszteset van, hiszen az algoritmus is sokkal bonyolultabb. A tesztelésnél kiemelt fontosságú volt, hogy a művelet befejezése után a fa továbbra is teljesítse az invariánsokat. Mivel a kulcsok helyességét ugyanazzal az eljárással ellenőrzöm, mint a beszúrásnál, ezért a következőkben feltesszük, hogy a bemenet megfelelő.

Teszteset rövid leírása	**Művelet eredménye**
Kulcs törlése üres fából. Nem történik semmi.	Sikeres
Olyan kulcs törlése a fából, ami nem található benne. Nem történik semmi.	Sikeres
Kulcs törlése egy elemszámú gyökérből. Kulcs és csúcs eltávolítása, a fa üres.	Sikeres
Kulcs törlése maximum elemszámú levélből. Kulcs eltávolítása.	Sikeres

Kulcs törlése minimum elemszámú levélből, kulcs elkérése balról.	Sikeres
Kulcs törlése minimum elemszámú levélből, kulcs elkérése jobbról.	Sikeres
Kulcs törlése minimum elemszámú levélből, kulcs jobboldali levélbe beszúrása.	Sikeres
Kulcs törlése minimum elemszámú levélből, kulcs baloldali levélbe beszúrása.	Sikeres
Kulcs törlése minimum elemszámú levélből, levél csúcs elkérése balról.	Sikeres
Kulcs törlése minimum elemszámú levélből, levél csúcs elkérése jobbról.	Sikeres
Kulcs törlése minimum elemszámú levélből, szülő jobboldali testvérébe összevonás.	Sikeres
Kulcs törlése minimum elemszámú levélből, szülő baloldali testvérébe összevonás.	Sikeres
Kulcs törlése minimum elemszámú levélből, maradék részfák összevonása a nagyszülőbe.	Sikeres
Kulcs törlése minimum elemszámú levélből, nagyszülőbe való összevonás után részfa elkérése balról.	Sikeres
Kulcs törlése minimum elemszámú levélből, nagyszülőbe való összevonás után részfa elkérése jobbról.	Sikeres
Kulcs törlése minimum elemszámú levélből, nagyszülőbe való összevonás után szülő jobb testvérével összevonás	Sikeres

Kulcs törlése minimum elemszámú levélből, nagyszülőbe való összevonás után szülő bal testvérével összevonás	Sikeres
Kulcs törlése minimum elemszámú levélből, nagyszülőbe való összevonás után megmaradt részfák újból összevonása a szülőbe.	Sikeres
Több kulcs törlése	Sikeres

Vezérlőelemek tesztelése

Teszteset rövid leírása	**Művelet eredménye**
Beszúrás elindítása gomb megnyomásával.	Sikeres
Beszúrás elindítása Enter leütésével.	Sikeres
Törlés elindítása gomb megnyomásával.	Sikeres
Törlés elindítása Enter leütésével.	Sikeres
Keresés elindítása gomb megnyomásával.	Sikeres
Keresés elindítása Enter leütésével.	Sikeres
Szöveges beviteli mezőkbe helytelen input megadása. Nem történi semmi, felhasználó értesítése.	Sikeres
Fájlba írás megfelelő fájlnév megadásával. Létrejön a két fájl.	Sikeres
Fájlból betöltés megfelelő fájlnév megadásával. Fa kimentett állapota megjelenik.	Sikeres
Animáció sebesség változtatása a csúszka segítségével. Animáció sebessége nő vagy csökken.	Sikeres

Menü tesztelése

Teszteset rövid leírása	**Művelet eredménye**
Üres fa létrehozása. Az fa eddigi állapota törlődik.	Sikeres
Feltöltés véletlen számokkal. 20 különböző elem beszúrásra kerül.	Sikeres
Zárójelezett alak létrehozása nemüres fa esetén. A szöveges alakja megjelenik az üzenetdobozban.	Sikeres
Zárójelezett alak létrehozása üres fa esetén. Az üzenetdobozban „Zárójelezett alak: üres” szöveg megjelenik.	Sikeres
Random számok beszúráshoz. 20 darab véletlenszerű különböző szám megjelenik az üzenetdobozban.	Sikeres
Random számok törléshez. 5 darab véletlenszerű különböző szám megjelenik az üzenetdobozban. Ezeket a számokat a fa tartalmazza.	Sikeres
Üzenetdoboz törlése. Az üzenetdoboz üres lesz.	Sikeres
Sötét téma választása. Az üzenetdoboz és a kirajzoló panel színe megváltozik.	Sikeres
Világos téma választása. Az üzenetdoboz és a kirajzoló panel színe megváltozik.	Sikeres

Továbbfejlesztési lehetőségek

A programot főleg a grafikai megjelenítésnél lehetne továbbfejleszteni. A `TreePanel` osztály átalakításával és egy teljes kontroller csomag hozzáadásával hatékonyabban lehetne kezelni a modellben történő változásokat és a megjelenítés fázisait. A modellben a törlés és beszúrás műveleteit továbbfejlesztve a program képes lehetne akár n-edfokú B+ fák ábrázolására is.

6. Összefoglalás

Kész programról

A program célja a B+ fák műveletinek ábrázolása, a felhasználó számára követhetően megjelenítve. Ezt sikeresen kiviteleztem, de a program használata nem garantálja a B+ fák teljes körű megértését. A használatához elengedhetetlen előismeretek szükségesek. Tanulásra, gyakorlásra használható, hiszen az animáció és leírás segítségével könnyebben átlátható az adatszerkezet működése. A felhasználónak lehetősége van bármikor menteni és betölteni a fát. Kényelmi funkció még a véletlenszerű számok kiírása törléshez és beszúráshoz. A program ugyanakkor csak a B+ fa szemléltetésére alkalmas, a levelekben nem tárol további adatokat, csak kulcsokat.

Hivatkozások

[1] Wikipedia: B+ tree
https://en.wikipedia.org/wiki/B%2B_tree 2016-05-06

[2] Járai Antal: Bevezetés a matematikába, Eötvös Kiadó, 2012, [444]
ISBN-987-963-284-077-2

[3] Dr. Ásványi Tibor: B+ fák
http://aszt.inf.elte.hu/~asvanyi/ad/B+ fa.pdf 2016-01-24