4.4.4. Tördelőtáblázat-indexek hatékonysága

Ideális esetben elég sok kosarunk van ahhoz, hogy a legtöbb kosár egy blokkból álljon. Ha ez így van, akkor a tipikus keresés csak egy lemez I/O-művelettel jár, míg a beszúrás és a törlés két lemez I/O-műveletet igényel. Ez a szám jelentősen jobb, mint a hagyományos ritka vagy sűrű indexeknél, illetve a B-fa-indexeknél (habár a tördelőtáblázatok a B-fákkal ellentétben nem támogatják a 4.3.4. részben bemutatott tartományt eredményező lekérdezéseket).

Ha azonban a fájl nő, előbb-utóbb eljutunk egy olyan helyzethez, amikor egy átlagos kosárhoz tartozó lánc sok blokkot tartalmaz. Ha ez így van, akkor blokkok hosszú listáit kell végignéznünk, amely legalább egy lemez I/O-műveletet jelent blokkonként. Jó okunk van tehát rá, hogy az egy kosárra eső blokkok számát alacsonyan tartsuk.

Az eddig vizsgált tördelőtáblázatokat *statikus tördelőtáblázatoknak* nevezzük, mivel a B, a kosarak száma soha nem változik. Léteznek azonban különböző *dinamikus tördelőtáblázatok* is, ahol a B változhat, azaz a B megközelíti azt a számot, amelyet úgy kapunk, ha elosztjuk a rekordok számát azon rekordok számával, amelyek elférnek egy blokkban; ez azt jelenti, hogy körülbelül egy blokk tartozik egy kosárhoz. Két ilyen módszert fogunk bemutatni:

1. a kiterjeszthető tördelést a 4.4.5. részben és
2. a lineáris tördelést a 4.4.7. részben.

Az első úgy növeli a B értékét, hogy megduplázza azt, valahányszor túl kevésnek bizonyul, és a második mindig 1-gyel növeli a B értékét, valahányszor a fájl statisztikái alapján növelésre van szükség.

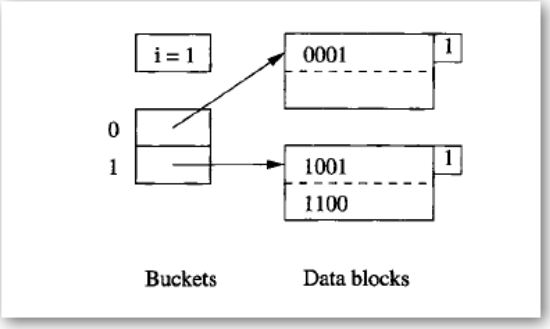
4.4.5. Kiterjeszthető tördelőtáblázatok

A dinamikus tördelés első megközelítését *kiterjeszthető tördelőtáblázatoknak* nevezzük. Az egyszerűbb statikus tördelőtáblázathoz képest a főbb kiegészítések:

1. A kosarakhoz egy közvetett szintet vezetünk be. Ez azt jelenti, hogy a kosarakat egy olyan, mutatókból álló tömb reprezentálja, ahol a mutatók blokkokra mutatnak ahelyett, hogy a tömb magukat az adatblokkokat tartalmazná.
2. A mutatókból álló tömb nőhet. Hossza mindig a 2 valamilyen hatványa, tehát egy növekedési lépésben a kosarak száma megduplázódik.
3. Mindamellett nem kell valamennyi kosárnak rendelkeznie egy adatblokkal; bizonyos kosarak osztozhatnak egy blokkon, ha a kosarakban levő rekordok elférnek ebben a blokkban.
4. A h tördelőfüggvény valamennyi kulcs esetén egy k bitből álló sorozatot számol ki, ahol a k elég nagy, mondjuk 32. A kosárjelzőszámok azonban mindenkor kevesebb számú bitet használnak, mondjuk i bitet a sorozat elejéről. Ily módon a kosártömbnek 2^i bejegyzése lesz, ahol i a felhasznált bitek száma.

4.31. példa: A 4.33. ábrán egy kisméretű, kiterjeszthető tördelőtáblázatot láthatunk. Az egyszerűség kedvéért tegyük fel, hogy $k = 4$; azaz a tördelőfüggvény egy mindössze négy bitből álló sorozatot ad vissza. Jelenleg ezen bitekből csak egy használatos, ahogyan azt fel is tüntettük a kosártömb fölötti dobozban. A kosártömbnek ily módon mindössze két bejegyzése van, egy a 0-hoz és egy az 1-hez.

A kosártömb bejegyzései két blokkra mutatnak. Az első tartalmazza az összes olyan aktuális rekordot, amelynek keresési kulcsa 0-val kezdődő bitsorozatot tördel, a második pedig azokat tartalmazza, amelyek keresési kulcsa 1-gyel kezdődő sorozatot tördel. A kényelem kedvéért a rekordok kulcsait úgy ábrázoljuk, mintha azok megegyeznének azokkal a teljes bitsorozatokkal, amelyekké a tördelőfüggvény konvertálja őket. Ily módon az első blokk egy rekordot tartalmaz, amelynek kulcsa a 0001-et tördeli, a második blokk azokat a rekordokat tartalmazza, amelyek kulcsai az 1001-et és az 1100-t tördelik. •



4.33 ábra: Kiterjeszthető tördelőtáblázat

A 4.33. ábrán megfigyelhetjük, hogy valamennyi blokk kinövésében megjelenik az 1-es szám. Ez a szám, amely tulajdonképpen a blokk fejlécében is megjelenhet, azt jelzi, hogy a tördelőfüggvény által visszaadott sorozatból hány bit használatos annak eldöntésére, hogy egy rekord az adott blokkhoz tartozik-e vagy sem. A 4.31. példában valamennyi blokk és rekord esetén egy bit használatos, de amint azt majd látni fogjuk, a különböző blokkokhoz használatos bitek száma változhat, ahogyan a tördelőtáblázat növekszik. Ily módon a kosártömb mérete az aktuálisan használt bitek száma által meghatározott, de előfordulhat, hogy bizonyos blokkok kevesebbet használnak.

4.4.6. Beszúrás kiterjeszthető tördelőtáblázatokba

Egy kiterjeszthető tördelőtáblázatba történő beszúrás ugyanúgy kezdődik, mint egy statikus tördelőtáblázatba történő beszúrás. A *K* keresésikulcs-értékű rekord beszúrásához kiszámoljuk a *h*(*K*) bitsorozatot, ennek vesszük az első *i* bitjét, és a kosártömb azon bejegyzéséhez megyünk, amelynek jelzőszáma ez az *i* bit. Megjegyzendő, hogy az *i*-t azért tudjuk meghatározni, mert a tördelő-adatszerkezetben el van tárolva.

Követjük a kosártömb ezen bejegyzésének mutatóját, és elérkezünk egy *B* blokkhoz. Ha van szabad hely a *B*-ben az új rekord elhelyezésére, akkor ezt megtesszük, és készen is vagyunk. Ha nincs szabad hely, akkor a *j* értékétől függően két lehetőségünk van; a *j* azt jelzi, hogy a tördelőfüggvény által kiszámolt érték hány bitje használatos annak eldöntésére, hogy a rekord a *B* blokkhoz tartozik vagy nem (ne feledjük, hogy a *j* érték az ábrán a blokkok kinövéseiben található).

1. Ha $j < i$, akkor a kosártömbbel semmit nem kell tennünk. Amit meg kell tennünk:

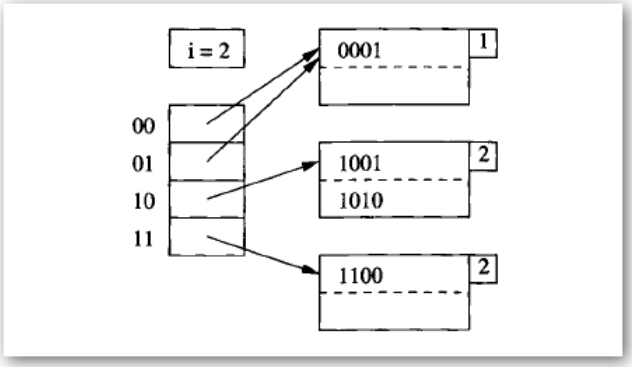
   a) A *B* blokkot kettévágjuk.
   b) A *B* rekordjait szétosztjuk a két blokk között, a (j + 1)-edik bit értéke alapján – azok a rekordok, amelyek kulcsa 0 az adott bitnél, maradnak a *B* blokkban, míg azok a rekordok, melyek ezen a helyen 1-et tartalmaznak, az új blokkba kerülnek.
   c) Mindegyik blokk „kinövésébe" $j + 1$ kerül, jelezvén, hogy ennyi bit használatos az odatartozás eldöntésére.
   d) A kosártömb mutatóit az új helyzethez igazítjuk úgy, hogy azok a bejegyzések, amelyek azelőtt a *B*-re mutattak, most vagy a *B*-re vagy az új blokkra mutassanak, a (j + 1)-edik bittől függően.

   Megjegyzendő, hogy a *B* blokk szétvágása nem biztos, hogy megoldja a problémát, hiszen a véletlen hozhatja úgy, hogy a *B* valamennyi rekordja a két blokk egyikébe kerül a szétvágás után. Ha ez így van, akkor meg kell ismételnünk az eljárást a *j* következő értékére, és arra a blokkra, amelyik még mindig tele van.
2. Ha $j = i$, akkor először meg kell növelnünk 1-gyel az *i* értékét. Megduplázzuk a kosártömb hosszát, hiszen most 2^{i+1} bemenetet tartalmaz. Tegyük fel, hogy a *w* egy *i* számú bitből álló sorozat, amely az előző kosártömb egyik bejegyzésének volt a jelzőszáma. Az új kosártömbben a *w*0 és *w*1 jelzőszámú bejegyzések (azaz a *w* 0-val és 1-gyel történő kiterjesztéséből származó két új szám) ugyanarra a blokkra mutatnak, mégpedig arra, amelyikre a *w* bejegyzés mutatott. Ez azt jelenti, hogy a két új bejegyzés megosztozik a blokkon, és a blokk maga nem változik. A blokkhoz való tartozás ugyanúgy meghatározott, mint a bitek előzőleg használt száma esetében. Végül kettévágjuk a *B* blokkot ugyanúgy, mint az 1. esetben. Mivel az *i* most nagyobb, mint *j* – alkalmazható az előző eset.

4.32. példa: Tegyük fel, hogy beszúrunk a 4.33. ábrán látható táblába egy olyan rekordot, melynek kulcsa az 1010 sorozatot tördeli. Mivel az első bit 1, a rekord a második blokkhoz tartozik. Azonban ez a blokk már tele van, tehát szét kell vágni. Mivel ebben az esetben $j = i = 1$, ezért először meg kell dupláznunk a kosártömböt a

4.34. ábrán látható módon. Az ábrán feltüntettük azt is, hogy $i = 2$.

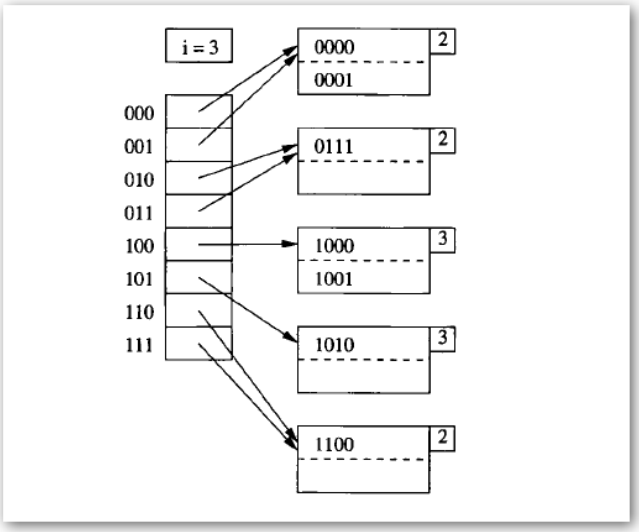
Figyeljük meg, hogy mindkét 0-val kezdődő bejegyzés arra a blokkra mutat, amelyben a tördelt kulcsok 0-val kezdődnek, és ez a blokk még mindig az 1-es egész számot tartalmazza a „kinövésben”, ami azt jelenti, hogy csak az első bit használatos a blokkhoz való tartozás eldöntésére. Azonban az 1-gyel kezdődő rekordok blokkját ketté kell vágnunk, így a hozzá tartozó rekordokat is szét kell válogatnunk az 10-val és az 11-gyel kezdődő rekordokra. Mindkét blokkban egy 2-es jelzi, hogy az odatartozást két bit határozza meg. Szerencsére a szétvágás sikeres, mivel a két új blokk mindegyike tartalmaz legalább egy rekordot, így nem kell rekurzívan tovább vágnunk.



4.34 ábra: Most a tördelőfüggvény két bitje van használatban

Most tegyük fel, hogy beszúrjuk azokat a rekordokat, amelyek kulcsai 0000-t és 0111-et tördelnek. Mindkét rekord a 4.34. ábra első blokkjába kerül, amely ezzel túlcsordul. Mivel ebben a blokkban csak egy bit használatos az odatartozás eldöntésére, és $i = 2$, ezért a kosártömbbel nem kell foglalkoznunk. Egyszerűen csak kettévágjuk a blokkot, a 0000 és 0001 a régiben maradnak, és a 0111 az új blokkba kerül. A kosártömb 01 bejegyzését beállítjuk, hogy az új blokkra mutasson. Ismét szerencsénk volt, hogy nem az összes rekord került az új blokkok valamelyikébe, így nem kellett rekurzívan tovább vágnunk.

Most tegyük fel, hogy egy olyan rekordot szúrunk be, amelynek kulcsa 1000-t tördel. Az 10-hoz tartozó blokk túlcsordul. Mivel ez már eleve 2 bitet használ az odatartozás eldöntésére, itt az ideje, hogy a kosártömböt ismét megnöveljük, és beállítsuk, hogy $i = 3$. A 4.35. ábra ezt az adatszerkezetet mutatja be. Figyeljük meg, hogy az 10-hoz tartozó blokkot kettévágtuk az 100-hoz és az 101-hez tartozó blokkokra, míg a többi blokk továbbra is két bitet használ az odatartozás eldöntésére. •



4.35 ábra: Most a tördelőfüggvény három bitje van használatban

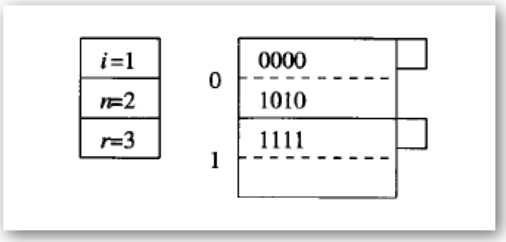
4.4.7. Lineáris tördelőtáblázatok

A kiterjeszthető tördelőtáblázatoknak van néhány fontos előnye. A legjelentősebb az, hogy egy rekord megtalálásához mindig csak egy adatblokkban kell keresnünk. A kosártömb egy bejegyzését is meg kell vizsgálnunk, ha azonban a kosártömb elég kicsi ahhoz, hogy elférjen az elsődleges memóriában, akkor nincs szükség lemez I/O-műveletre ahhoz, hogy hozzáférjünk a kosártömbhöz. Azonban a kiterjeszthető tördelőtáblázatok rendelkeznek néhány hiányossággal is:

1. Amikor meg kell dupláznunk a kosártömböt, akkor tekintélyes mennyiségű munkát kell végezni (ha az i nagy). Ez a munka megszakítja az adatfájlhoz való hozzáférést, vagy igen lassúvá tesz bizonyos beszúrásokat.
2. Ha a kosártömb méretét megduplázzuk, lehet, hogy nem fér majd el az elsődleges memóriában, vagy esetleg kiszorít olyan adatokat, amelyeket az elsődleges memóriában szeretnénk tartani. Ennek eredményeként egy eddig jól működő rendszer hirtelen elkezd sokkal több lemez I/O-műveletet használni, és elér egy észrevehető teljesítménycsökkenést.
3. Ha a blokkonkénti rekordok száma kicsi, akkor valószínű, hogy lesz egy olyan blokk, amelyet ketté kell majd vágni jóval előbb, mint ahogyan logikailag itt lenne az ideje. Ha például éppúgy, mint az eddigi példákban, két rekord van egy blokkban, akkor lehetséges, hogy három rekordnál ugyanaz a 20 bitből álló sorozat található, még akkor is, ha a rekordok összesen jóval kevesebben vannak, mint 2^{20}. Ebben az esetben $i = 20$ és egymillió bejegyzést kell használnunk a kosártömbben, még akkor is, ha a rekordokat tartalmazó blokkok száma jóval kevesebb, mint egymillió.

Egy másik stratégia, amit lineáris tördelőtáblázatoknak hívunk, jóval lassabban növeli a kosarak számát. A lineáris tördelés legfontosabb új elemei:

- n, a kosarak száma mindig úgy alakul ki, hogy a rekordok blokkonkénti átlagos száma a blokkot megtöltő rekordoknak egy állandó hányadát képezze, mondjuk 80%-át.
- Mivel a blokkokat nem lehet mindig szétvágni, ezért a túlcsordulásblokkok megengedettek, habár az egy kosárra eső túlcsordulásblokkok átlagos száma jóval kevesebb lesz, mint 1.
- A kosártömb bejegyzéseinek megszámozására használt bitek száma $\lceil \log_2 n \rceil$, ahol n a kosarak aktuális száma. Ezeket a biteket mindig a tördelőfüggvény által visszaadott bitsorozat *jobb* széléről vesszük (alacsony prioritás).
- Tegyük fel, hogy a tördelőfüggvény i bitje használatos a tömb bejegyzéseinek megszámozására, és hogy egy K kulcsú rekordot szánunk az $a_1a_2...a_i$ kosárba; ez azt jelenti, hogy a $h(K)$ utolsó i bitje $a_1a_2...a_i$. Tekintsük az $a_1a_2...a_i$-t, mint egy i bitből álló bináris egészet, és jelöljük m-mel. Ha $m < n$, akkor az m jelzőszámú kosár létezik, és a rekordot ebben a kosárban helyezzük el. Ha $n \leq m < 2^i$, akkor az m jelzőszámú kosár még nem létezik, így a rekordot az $m - 2^{i-1}$ jelzőszámú kosárban helyezzük el, amit úgy kapnánk, ha kicserélnénk az a_1-et (aminek 1-nek kell lennie) 0-ra.



4.36 ábra: Lineáris tördelőtáblázat

4.33. példa: A 4.36. ábrán egy lineáris tördelőtáblázatot láthatunk, ahol $n = 2$. Jelenleg a tördelési értéknek csak egy bitjét használjuk a rekordok kosarakhoz való tartozásának eldöntésére. A 4.31. példában bemutatott törvényszerűséget felhasználva tegyük fel, hogy a h tördelőfüggvény 4 bitet hoz létre, és a rekordokat azzal az értékkel ábrázoljuk, amit a rekord keresési kulcsára alkalmazott h függvény eredményez.

A 4.36. ábrán két kosarat láthatunk, mindkettő egy blokkból áll. A kosarak jelzőszámai 0 és 1. Minden olyan rekord, amelynek tördelési értéke 0-ra végződik, az első kosárba kerül, míg azok, amelyeknek tördelési értéke 1-re végződik, a második kosárba kerülnek.

Az adatszerkezet részét képezi még az i paraméter (a tördelőfüggvényből használatos bitek száma), az n (a kosarak aktuális száma) és az r (a tördelőtáblázat rekordjainak aktuális száma). Az r/n arány korlátozott lesz, így az átlagos kosár körülbelül egy blokkot igényel majd. Az n megválasztásakor azt az elvet követjük, mely szerint

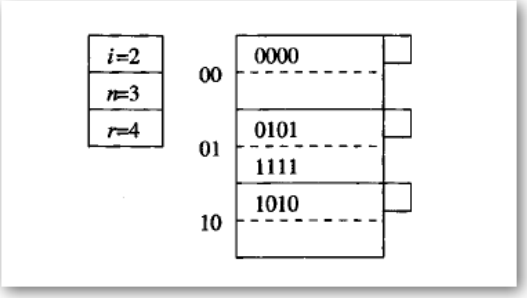
a fájlban legfeljebb 1,7n rekord van, azaz $r \leq 1{,}7n$. Ily módon, mivel a blokkok két rekordot tartalmaznak, egy kosár átlagos kihasználtsága nem haladja meg egy blokk kapacitásának a 85%-át. •

4.4.8. Beszúrás lineáris tördelőtáblázatokba

Amikor egy új rekordot szúrunk be, akkor a megfelelő kosarat a 4.4.7. részben vázolt algoritmussal határozzuk meg. Ez azt jelenti, hogy kiszámoljuk a $h(K)$-t, ahol K a rekord kulcsa, és meghatározzuk azt a számot, ahány bitet figyelembe kell vennünk a $h(K)$ bitsorozat végéről, hogy aztán a kosár jelzőszámaként használjuk. A rekordot vagy ebbe a kosárba tesszük, vagy (ha a kosár jelzőszáma nagyobb vagy egyenlő, mint n) abba a kosárba, amit úgy kapunk, hogy a vezető bitet 1-ről 0-ra cseréljük. Ha a kosárban nincs szabad hely, akkor készítünk egy túlcsordulás-blokkot, hozzáláncoljuk a kosárhoz, és ebbe tesszük a rekordot.

Minden egyes beszúrásnál összehasonlítjuk a rekordok aktuális számát, az r-et, az r/n hányados felső határával, és ha ez a hányados túl magas, akkor hozzáadjuk a táblához a következő kosarat. Megjegyzendő, hogy az általunk hozzáadott kosárnak nincs semmi köze ahhoz a kosárhoz, amelybe a beszúrás történik! Ha a hozzáadott kosár jelzőszámának bináris reprezentációja $1a_2...a_i$, akkor szétszedjük a $0a_2...a_i$ jelzőszámú kosarat, oly módon, hogy annak rekordjait egyik vagy másik kosárba tesszük, az utolsó i bitjüktől függően. Figyeljük meg, hogy valamennyi rekord tördelési értéke $a_2...a_i$-re végződik, és csupán jobbról az i-edik bit fog változni.

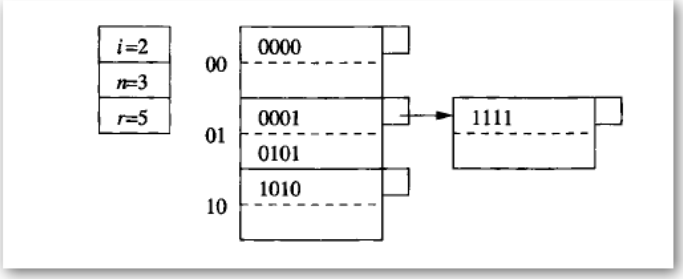
Az utolsó fontos részlet, hogy mi történik, ha az n túllépi a 2^i értéket. Ekkor az i-t megnöveljük eggyel. Technikailag valamennyi kosár jelzőszáma kap egy 0-t a saját bitsorozata elé, de semmi más fizikai változtatásra nincs szükség, hiszen ezek a bitsorozatok, egész számként értelmezve, ugyanazok maradnak.



4.37 ábra: Egy harmadik kosár hozzáadása

4.34. példa: Folytatjuk a 4.33. példát, és megnézzük, mi történik, ha egy olyan rekordot szúrunk be, amelynek kulcsa a 0101 értéket tördeli. Mivel ez a bitsorozat 1-re végződik, a rekord a 4.36. ábra második kosarába kerül. Van szabad hely a rekord számára, így nem kell túlcsordulás-blokkot készíteni.

Mivel azonban most 4 rekord van 2 kosárban, túlléptük az 1,7 hányadost, ezért fel kell emelnünk az n értékét 3-ra. Mivel $\lceil \log_2 3 \rceil = 2$, kezdhetünk úgy gondolni a 0 és 1 kosarakra, mint 00 és 01 kosarakra, de nem szükséges módosítani az adatszerkezetet. Hozzáadjuk a táblához a következő kosarat, amelynek a jelzőszáma 10 lesz. Ezután szétszedjük a 00 kosarat, azt a kosarat, amelynek jelzőszáma csak az első bitben különbözik a hozzáadott kosártól. Amikor elvégezzük a szétszedést, akkor az a rekord, amelynek kulcsa 0000-t tördel, marad a 00-s kosárban, míg az a rekord, amelynek kulcsa 1010-t tördel, átmegy az 10-s kosárba, mivel a végződések így kívánják. Az eredményül kapott tördelőtáblázatot a 4.37. ábrán láthatjuk.



4.38 ábra: Szükség esetén túlcsordulás-blokkokat használunk

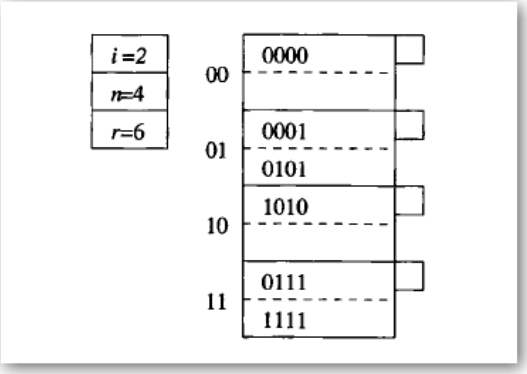
Most tegyük fel, hogy egy olyan rekordot akarunk beszúrni, amelynek keresési kulcsa 0001-et tördel. A két utolsó bit 01, így ebbe a kosárba tesszük, amely jelenleg létezik. Sajnos a kosár blokkja tele van, így hozzáadunk

egy túlcsordulás-blokkot. A három rekordot elosztjuk a kosár két blokkja között; a tördelt kulcsok numerikus sorrendjében helyezzük el őket, de a sorrend nem igazán fontos. Mivel a táblában a rekordok és a blokkok aránya 5/3, és ez kevesebb, mint 1,7, ezért nem készítünk új kosarat. Az eredményt a 4.38. ábrán láthatjuk.

Végül, nézzük meg egy olyan rekord beszúrását, amelynek keresési kulcsa 0111-et tördel. Az utolsó két bit 11, de az 11-es kosár nem létezik. Ily módon átirányítjuk ezt a rekordot a 01-es kosárba, amely szám csak a 0-s első bitben különbözik a keresett értéktől. Az új rekord befér a kosár túlcsordulásblokkjába.

Azonban a rekordok kosarakhoz viszonyított aránya meghaladja az 1,7-et, ezért létre kell hoznunk egy új kosarat, 11 jelzőszámmal. Ez a kosár történetesen az, amit az új rekord számára kerestünk. Szétosztjuk a 01-es kosár négy rekordját, a 0001-es és 0101-es marad, a 0111-es és az 1111-es átmegy az új kosárba. Mivel a 01-es kosár jelenleg két rekordot tartalmaz, ezért eltörölhetjük a túlcsordulásblokkot. A tördelőtáblázat ezen állapotát a 4.39. ábrán láthatjuk.

Figyeljük meg, hogy a 4.39. ábrába történő következő beszúrásnál a rekordok és a kosarak aránya meg fogja haladni az 1,7-et. Ekkor meg fogjuk növelni az *n* értékét 5-re, és az *i* értéke 3 lesz. •



4.39 ábra: Egy negyedik kosár hozzáadása

4.35. példa: Egy lineáris tördelőtáblázatban történő keresés megegyezik azzal az eljárással, amellyel kiválasztjuk azt a kosarat, amelybe a beszúrni kívánt rekord kerülni fog. Ha a keresett rekord nincs ebben a kosárban, akkor sehol máshol sem lehet. Szemléltetésképpen nézzük meg a 4.37. ábra helyzetét, ahol $i = 2$ és $n = 3$.

Először tegyük fel, hogy egy olyan rekordot akarunk megtalálni, amelynek kulcsa az 1010-t tördeli. Mivel $i = 2$, ezért a két utolsó bitet nézzük, ami 10, és ezt bináris egészként értelmezünk, nevezetesen $m = 2$. Mivel $m < n$, ezért az 10-s kosár létezik, itt fogjuk keresni. Ne feledjük, hogy csupán az a tény, hogy találunk egy olyan rekordot, amelynek tördelési értéke 1010, még nem jelenti azt, hogy ez az a rekord, amit keresünk; ahhoz, hogy ebben biztosak legyünk, ellenőriznünk kell a rekord teljes kulcsát.

Másodszor nézzük meg egy olyan rekordnak a keresését, amelynek kulcsa az 1011-et tördeli. Most olyan kosárban kell keresnünk, amelynek jelzőszáma 11. Mivel ez a szám bináris egészként $m = 3$, és $m \geq n$, az 11-es kosár nem létezik. Átmegyünk a 01-es kosárba, kicserélvén a vezető bitet 1-ről 0-ra. A 01-es kosárnak azonban nincs olyan rekordja, amelynek a kulcsa az 1011-et tördelné, így a keresett rekord biztosan nincs a tördelőtáblázatban. •