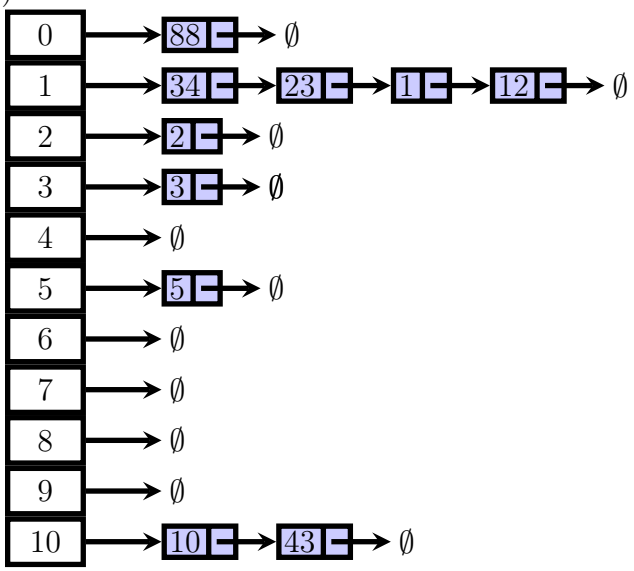
1. Vödrös hash-t használva akarunk számokat tárolni, a $h(x) = x \pmod{11}$ hash függvényt használva.
(a) Szúrjuk be a kezdetben üres, 11 méretű hash táblába a 2, 5, 12, 1, 3, 88, 23, 43, 10, 34 elemeket!
(b) Hogyan zajlanak a feltöltött táblában a következő keresések: keres(1), keres(20), keres(45)?
(c) Hogyan zajlik az a) pont táblájában az 23 törlése?

Megoldás:
(a)



(b) keres(1): Az 1. listán haladva a 3. cellában megtalálja az 1 elemet.
keres(20): Az 9. listán indulva rögtön kiderül, hogy nincs a táblában a 20.
keres(45): Az 1. listán haladva a végére érve kiderül, hogy nincs a táblában a 45.

(c) Először keres(23), ez az 1. listán haladva a 2. cellában megtalálja az 23 elemet. Ezután a 34-ből a 23-ra mutatót átállítjuk, hogy oda mutasson, ahova a 23 mutatója mutatott (az 1-re).

2. Nyitott címzéssel hash-elünk egy kezdetben üres $M = 11$ méretű táblába a $h(x) = x \pmod{11}$ hash-függvénnyel. Mi lesz a tábla állapota, ha a 4, 5, 14, 15, 16, 26, 3 kulcsokat ebben a sorrendben beszúrjuk és az ütközések feloldására (a) lineáris próbát használunk? (b) kvadratikus maradék próbát használunk? (c) kettős hash-elést használunk, amikor $h'(x) = (7x \pmod{(10)}) + 1$ a második hash-függvény? Hány ütközés történt az egyes esetekben?

Megoldás: (a) Az ütközések száma: $0 + 0 + 0 + 2 + 4 + 4 + 4 = 14$, a végén a tábla elemei sorban 26, 16, 15, 14, 4, 5, –, –, –, –, 3

(b) Az ütközések száma (felfelé indulok): $0+0+0+3+1+4+2 = 10$, a végén a tábla elemei sorban 26, –, 3, 14, 4, 5, 16, –, 15, –, –

(c) Az ütközések száma (most is lefelé indulok, mint a lineáris próbánál): $0+0+0+1+1+1+2 = 5$, a végén a tábla elemei sorban –, 26, 16, 14, 4, 5, –, –, –, 15, 3

3. Egy B_{20}-fának (huszadrendű B-fának) 10^9 levele van. Mekkora a fa szintjeinek minimális, illetve maximális száma?

Megoldás: Ha m szint van, akkor $2 \cdot 10^{m-2} \leq 10^9 \leq 20^{m-1}$. Tehát $m \leq 9 - \log_{10}(2) + 2 \approx 10.69897$ és $m > \log_{20}(10^9) + 1 = \log_{10}(10^9)/\log_{10}(20) + 1 \approx 9/(1.30103) + 1 \approx 7.91759608$

---

4. Előfordulhat-e nyitott címzéses hash-elés esetén, hogy az $n > 3$ méretű táblában csak 3 elem van, de a keresés lépésszáma n?

Megoldás: Igen, ha korábban törlések is történtek, pontosabban, ha minden helyről, ahol most nincs elem volt törlés. Ekkor a keresés ezeken továbbmegy, és ha a 3 bent levő egyike sem a keresett elem, akkor mindenképpen bejárjuk az egész táblát.

5. Jó választás-e $M = 7$ méretű táblánál a $h(x) = x^2 \pmod{7}$ hash-függvény?

Megoldás: Nem, mert $f(1) = f(6)$, $f(2) = f(5)$ és $f(3) = f(4)$, tehát minden a 0, 1, 4, 2 helyek egyikénél kezd ahelyett, hogy a 7 lehetséges helyre szétoszlanának (több ütközés várható).

6. Az alábbi 11 méretű hash táblába nyílt címzéssel, lineáris próbával szúrtunk be néhány egész számot majd egyet közülük kitöröltünk, így az alábbi állapotot kaptuk ($*$ jelöli a törölt cellát, a kitöltetlen cellák mindvégig üresek voltak). A használt hash függvény a $h(x) = x \pmod{11}$ függvény volt.

0	1	2	3	4	5	6	7	8	9	10
11	4	*	3	15		6	18			22

Ebben a táblázatban egyetlen olyan szám van, amire igaz, hogy ha ezt a számot kitöröljük, majd újra beszúrjuk, akkor a szám egy másik helyre kerül vissza. Melyik ez a szám? Válaszának indoklására mutassa be, hogy hogyan zajlik ennek a számnak a törlése és beszúrása.

Megoldás: Ez a szám az 1. cellában levő 4 lesz. Egyfelől könnyen látható, hogy a táblában levő többi szám mind a mostani helyére kerül vissza törlés és beszúrás után. A 4 törlése $h(4) = 4$ alapján a 4. cellánál kezdődik, innen lépkedünk visszafelé a 3., 2., majd 1. cellához, innen töröljük a négyest, majd *-ot írunk a cellába. A 4-es beszúrása az így kapott táblába ismét kereséssel kezdődik a 4. cellánál. Itt ütközést kapunk, tovább kell lépni a 3. cellához, ahol szintén ütközést tapasztalunk. A keresőúton következő cella a 2. szabad, a * helyére beírható a 4-es.

7. Egy B_{11} fába az x elem beszúrásakor 2 csúcsvágást kellett végezni. Legalább hány elemet tárolt a beszúrás előtt az adatstruktúra?

Megoldás: Mivel 2 csúcs vágás volt, ezért a leveleken kívül van legalább 2 szint. A legkevesebb elem nyilván akkor lesz, ha a lehető legkisebb a magasság és minden csúcsnak a lehető legkevesebb gyereke van. Ebben az esetben a két csúcsvágás a gyökérben és annak egyik gyerekében történt. Csúcsvágás csak akkor történik, ha már 11 gyerek van. Ezért a gyökérnek van 11 gyereke, mindegyik gyerekének van legalább 6 gyereke, de az egyiknek 11. Ez összesen $10 \cdot 6 + 11 = 71$ levelet eredményez. Tehát legalább 71 elemet tárolt már a fa.

8. Tervezzünk olyan adatszerkezetet, ami egy rendezett halmaz elemeinek tárolására szolgál. A megvalósítandó műveletek: *Felépít*(n): n elemből felépíti a struktúrát; *Mintör*, *Maxtör*: a min. illetve max. elem törlése; *Beszúr*(x): az x elemet a struktúrába illeszti. Az egyes műveletek uniform költsége ne legyen több, mint *Felépít*: $O(n)$; *Mintör, Maxtör, Beszúr*: $O(\log n)$, ahol n a tárolt elemek száma.

Megoldás: **Adatszerkezet**: Az elemeket *kétszer tároljuk*: egyszer egy max kupacban, egyszer pedig egy min kupacban, és az *ugyanahhoz a kulcshoz tartozó két példányt mutatókkal összekötjük.*

- **Felépít**: Mindkét kupac felépíthető $O(n)$ idő alatt mutatókkal együtt is.
- **Mintör**: A minimum elem eltávolítható a min kupacból $O(\log n)$ idő alatt, és a mutató segítségével megtaláljuk ugyanennek a kulcsnak a helyét a max kupacban is. Mivel ismert a pozíció, a pozíció szerinti törlés elvégezhető $O(\log n)$ idő alatt.
- **Maxtör**: A maximum elem eltávolítható a max kupacból $O(\log n)$ idő alatt, és a mutató segítségével megtaláljuk ugyanennek a kulcsnak a helyét a min kupacban. Ismét, mivel ismert a pozíció, a törlés elvégezhető $O(\log n)$ idő alatt.
- **Beszúrás**: Be kell szúrni az elemet mindkét kupacba, mindkettő $O(\log n)$ időben elvégezhető (a mutatókat az elemek a beszúrás folyamán "húzzák magukkal").

---

9. A $T[0 : M]$ táblában $2n$ elemet ($n < M/3$) helyeztünk el valamilyen hash-függvény segítségével, amikor azt tapasztaltuk, hogy az elemek mindegyike az első $3n$ hely egyikére került. Ha nem volt közben törlés és a végén a táblában minden $3i$ indexű hely üres maradt ($0 \leq i < n$), akkor legfeljebb hány ütközés lehetett, ha (a) lineáris próbát használtunk? (b) kvadratikus maradék próbát használtunk?

Megoldás: (a) Ezek szerint a táblázat elején mindig két foglalt helyet követ egy üres. Lineáris próbánál csak a két szomszédos ütközhetett a másodiknak érkező beszúrásakor, tehát legfeljebb n ütközés volt. Ennyi lehet is, pl. ha az elemek sorban 2, $M + 2$, 5, $M + 5$, stb.

(b) Mivel mindenkinek legalább az egyik szomszédja üres, elemenként megint legfeljebb egy ütközés lehetett. Ezért az ütközések csak egy-egy szomszédos páron belül történhettek. Azaz most is legfeljebb n ütközés lehetett. Az előző példa mutatja, hogy ez lehetséges is.

10. A 0 és 70 közötti páratlan számokat szúrjuk be valamilyen sorrendben, nyílt címzéssel, lineáris próbával egy kezdetben üres 47 méretű hash táblába, a $h(x) = x \pmod{47}$ hash függvényt használva. Mutassa meg, hogy mindegy, hogy a számok milyen sorrendben érkeznek, az ütközések száma minden sorrend esetén ugyanannyi.

    Megoldás: Nem lesz egyetlen ütközés sem, bármelyik sorrendben is szúrjuk be őket. Egy x elem beszúrásakor csak akkor lehetne ütközés, ha már korábban beszúrtunk volna egy olyan y elemet, amire $x \equiv y \pmod{47}$. Ilyen azonban nem lesz, hiszen a beszúrandó elemek maradéka 47-tel osztva rendre $1, 3, 5, \ldots, 45, 0, 2, 4, \ldots, 22$.

11. Tervezzen adatstruktúrát a következő feltételekkel. Természetes számokat kell tárolni, egy szám többször is szerepelhet. A szükséges műveletek:
BESZÚR(i): i egy újabb példányát tároljuk
TÖRÖL(i): i egy példányát töröljük
MINDTÖRÖL(i): i összes példányát töröljük
DARAB(i): visszaadja, hogy hány példány van i-ből
ELEM(K): megmondja, a nagyság szerinti rendezésben a K-adik elem értékét.
Az adatstruktúra legyen olyan, hogy ha m-*féle* elemet tárolunk, akkor mindegyik művelet lépésigénye $O(\log m)$.
(Például ha a tárolt elemek 1,1,3,3,3,8, akkor DARAB(1)=2, ELEM(4)=3 és m =3.)

    Megoldás: A feladat megoldásához minden csúcsban egy hármast tárolunk:

    $$(i,\ \#i,\ \texttt{méret})$$

    ahol:

    - i – a csúcsban tárolt szám,
    - $\#i$ – az i-ből tárolt példányok száma,
    - `méret` – a csúcs **bal oldali részfájában** tárolt elemek (multiplicitással vett) száma.

    Adatszerkezet: egy *piros-fekete fa*, melyben a csúcsok az i értékek szerint rendezettek (kulcs: a hármashoz tartozó i).

    A műveletek a következők:

    - **BESZÚR(i)**:
        - Keressük meg i-t a fában ($O(\log m)$).
        - Ha nem találjuk, beszúrjuk az $(i, 1, 0)$ hármast, mivel az új csúcsnak nincs bal oldali részfája (tehát `size` = 0).
        - Ha a csúcs már létezik, növeljük a $\#i$ értékét eggyel.
        - Minden olyan csúcs esetében, ahol a beszúrás *balra* történik (vagyis a bal részfába kerül az új elem), frissítjük az adott csúcs `size` értékét, eggyel növelve azt.
    - **TÖRÖL(i)**:
        - Keressük meg i-t a fában ($O(\log m)$).
        - Amennyiben megtaláljuk, csökkentsük a $\#i$ értékét eggyel, és frissítsük az érintett csúcsok `size` értékét, ha a törlés a bal oldali részfából történt.
        - Ha $\#i$ 0-ra csökken, töröljük a csúcsot a fából.
    - **MINDTÖRÖL(i)**:
        - Keressük meg i-t, majd töröljük az $(i, \#i, \texttt{méret})$ csúcsot a fából, frissítve az érintett csúcsok `size` értékeit.

- **DARAB**(i):
  - Keressük meg i-t, és adjuk vissza a csúcsban lévő $\#i$ értékét.
- **ELEM**(K):
  - Induljunk a gyökértől, és minden csúcsnál olvassuk le a bal oldali részfa elemeinek számát, legyen ez L.
  - Ha $K \leq L$, akkor a keresést a bal gyermek felé folytatjuk.
  - Ha $K > L + \#i$, akkor állítsuk be $K := K - (L + \#i)$, és lépjünk a jobb gyermekre.
  - Ha $L < K \leq L + \#i$, akkor a keresett K-adik elem az aktuális csúcsban tárolt i.

Mivel a fa magassága $O(\log m)$, minden művelet (BESZÚR, TÖRÖL, MINDTÖRÖL, DARAB, valamint ELEM) $O(\log m)$ időben végrehajtható.