4 Halmaz

A halmaz ábrázolása karakterisztikus függvénnyel történik, amelyet általában folytonos módon ábrázolunk a következő módon:

- Adunk egy felső korlátot a lehetséges elemek számára. (Megadjuk az „alaphalmaz” számosságát.) (Itt használjuk ki, hogy nem engedjük meg a végtelen számosságú halmazok létét.)
- Lefoglalunk annyi bitet, ahány eleme lehet a halmaznak.
- A lehetséges elemeket „sorba állítjuk”.
- Az elemekhez a sorrend alapján kölcsönösen egyértelműen hozzárendelünk egy-egy bitet a lefoglaltak közül.
- A továbbiakban ezen bit 1-es értéke jelzi ha az adott elem eleme a halmaznak, és a bit 0-s értéke jelzi, ha nem eleme a halmaznak.

4.1 Halmazok ábrázolása

A lehetséges elemek halmaza:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Az A = { 0, 2, 5, 8, 9, 13, 15 } halmaz reprezentációja:

1	0	1	0	0	1	0	0	1	1	0	0	0	1	0	1

C-beli megvalósítás:

```
int a[] = { 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1 };
```

4.2 A halmazműveletek megvalósítása

4.2.1 Az „eleme” művelet

```
int a[N] = { 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1 }, i = 10;

if( i >= 0 && i < N && a[i] != 0 )
        printf( "%d benne van a halmazban.\n", i );
else
        printf( "%d nincs benne a halmazban.\n", i );
```

4.2.2 Az unió képzés művelete

- Visszavezethető a logikai VAGY műveletre.
- Tekintsük a két halmazt leíró karakterisztikus függvények bitvektorát.
- Hajtsuk végre a két bitvektor közötti (bitenkénti) logikai VAGY műveletet.
- Eredményként a két kiinduló halmaz uniójának karakterisztikus függvényéhez tartozó bitvektort kapjuk.

```
int a[N] = { 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1 }, b[M] = { 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1 };
int i;

for( i = 0; i < M; ++i )        /* Tfh. N >= M */
    if( b[i] == 1 )
        a[i] = 1;
```

4.2.3 A metszet képzés művelete

- Visszavezethető a logikai ÉS műveletre.
- Tekintsük a két halmazt leíró karakterisztikus függvények bitvektorát.
- Hajtsuk végre a két bitvektor közötti (bitenkénti) logikai ÉS műveletet.
- Eredményként a két kiinduló halmaz metszetének karakterisztikus függvényéhez tartozó bitvektort kapjuk.

```
int a[N] = { 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1 }, b[M] = { 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1 };
int i;

for( i = 0; i < N; ++i )                /* Tfh. N >= M */
    if( i < M && b[i] == 1 && a[i] == 1 )
        a[i] = 1;
    else
        a[i] = 0;
```

4.2.4 A komplementer képzés művelete:

Visszavezethető a logikai NEGÁLÁS műveletére.

```
int a[N] = { 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1 };
int i;

for( i = 0; i < N; ++i )
    if( a[i] == 1 )
        a[i] = 0;
    else
        a[i] = 1;
```

4.2.5 A különbség képzés művelete:

Visszavezethető logikai műveletekre.

```
int a[N] = { 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1 }, b[M] = { 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1 };
int i;

for( i = 0; i < M; ++i )            /* Tfh. N >= M */
    if( b[i] == 1 )
        a[i] = 0;
```

5 Multihalmaz

A multihalmazok és a halmazok között annyi a különbség, hogy a multihalmaz egy elemet többször is tartalmazhat. Egy elem multiplicitása azt adja meg, hogy az elem hányszor van benne a halmazban. Az ábrázolás szintén karakterisztikus függvénnyel történik, azonban ebben az esetben a tömb elemekben nem logikai értéket tárolunk, hanem egész számokat (melyek a multiplicitást jelentik).

- Adunk egy felső korlátot a lehetséges elemek számára.
- Lefoglalunk annyi egész számnak megfelelő helyet, ahány eleme lehet a multihalmaznak.
- A lehetséges elemeket „sorba állítjuk”.
- Az elemekhez a sorrend alapján kölcsönösen egyértelműen hozzárendelünk egy-egy egész számot a lefoglaltak közül.
- A továbbiakban ezen egész szám értéke jelzi, hogy az adott elem (hányszoros) eleme a halmaznak (a 0-s érték jelzi, ha nem eleme a halmaznak).

5.1 Multihalmaz ábrázolása

A lehetséges elemek halmaza:

0	1	2	3	4	5

Az A = { 0, 0, 1, 1, 1, 3 } halmaz reprezentációja:

2	3	0	1	0	0

C-beli megvalósítás:

```
int a[] = { 2, 3, 0, 1, 0, 0 };
```

5.2 A multihalmaz-műveletek megvalósítása

5.2.1 Az „eleme” művelet

```
int a[N] = { 2, 3, 0, 1, 0, 0 }, i = 3;

if( i >= 0 && i < N && a[i] > 0 )
        printf( "%d benne van a halmazban.\n", i );
else
        printf( "%d nincs benne a halmazban.\n", i );
```

5.2.2 Az unió képzés művelete

Egy adott értékű elem annyiszor fog szerepelni az unióban, ahányszor a kiinduló multihalmazokban (összeg):

```
int a[N] = { 2, 3, 0, 1, 0, 0 }, b[M] = { 100, 0, 0, 1, 10 };
int i;

for( i = 0; i < M; ++i )          /* Tfh. N >= M */
    if( b[i] > 0 )
        a[i] += b[i];
```

5.2.3 A metszet képzés művelete

Egy adott értékű elem annyiszor fog szerepelni a metszetben, ahányszor a kiinduló multihalmazokban mindegyikében szerepel. (minimum)

```
int a[N] = { 2, 3, 0, 1, 0, 0 }, b[M] = { 100, 0, 0, 1, 10 };

int i;

for( i = 0; i < N; ++i )                /* Tfh. N >= M */
        if( i < M && b[i] < a[i] )
                a[i] = b[i];
        else if( i > M )
                a[i] = 0;
```

5.2.4 A különbség képzés művelete:

Egy adott elem a kiinduló multihalmazokban való előfordulásainak különbségeszer fog előfordulni, ha a baloldali multihalmazban fordul elő többször. Egyébként 0-szor. (kivonás)

```
int a[N] = { 2, 3, 0, 1, 0, 0 }, b[M] = { 100, 0, 0, 1, 10 };

int i;

for( i = 0; i < M; ++i )            /* Tfh. N >= M */
        if( b[i] < a[i] )
            a[i] −= b[i];
        else
            a[i] = 0;
```