

2. gyakorlat

Keresési, rendezési algoritmusok

A keresés célja: lokalizálni egy elemet egy N elemből álló halmazban a kulcsa alapján.

Szekvenciális keresések: adott K_1, \dots, K_N sorozat, és a k keresett érték.

1. Szekvenciális keresés rendezetlen tömbben

```
BEGIN
  INPUT: K[N], k
  i := 1
  WHILE i <= N
    IF K[i] = k
      OUTPUT: „sikerés keresés”
    ELSE
      i := i+1
    END
  END
  OUTPUT: „sikertelen keresés”
END
```

2. Szekvenciális keresés rendezett tömbben

(tfh. a tömb elemei növekvő sorrendben rendezettek)

```
BEGIN
  INPUT: K[N], k
  i := 1
  WHILE i <= N
    IF K[i] = k
      OUTPUT: „sikerés keresés”
    ELSE
      IF K[i] > k
        OUTPUT: „sikertelen keresés”
      ELSE
        i := i+1
      END
    END
  END
  OUTPUT: „sikertelen keresés”
END
```

Sikertelen keresés esetén átlagosan kétszer gyorsabb, mint a rendezetlen tömbben való keresés.

3. Felezéses (bináris/logaritmusos) keresés rendezett sorozatban

A sorozat közepén kezdjük a keresést. Ha nem ez a keresett elem, akkor megvizsgáljuk, a tömb alsó vagy felső felébe esik-e, majd megfelezzük azt is, így folytatva a keresést egészen addig, míg meg nem találtuk a keresett elemet, vagy a vizsgált tömb rész már csak egyetlen elemet tartalmaz.

```

BEGIN
  INPUT: K[N], k
  e := 1, u := N
  i := N
  WHILE i >= 1
    i := (e+u)/2           //egész osztás, a középső elem indexe
    IF K[i] = k
      OUTPUT: „sikeres keresés”
    ELSE
      IF K[i] > k
        u := i-1
      ELSE
        e := i+1
      END
    END
  END
  OUTPUT: „sikertelen keresés”
END

```

Rendezés: az a folyamat, amikor egy halmaz elemeit valamilyen szabály szerint (növekvő vagy csökkenő) sorba rendezzük. A rendezés meggyorsítja az elemek későbbi keresését, mivel egy rendezett halmazon való keresés jóval gyorsabban megy végbe.

A rendezési algoritmusok csoportosítása:

1. Helyben rendező
 - Buborék (cserélő) rendezés (*Bubble sort*)
 - Shell rendezés (*Shell sort*)
 - Minimum-/maximumkiválasztásos rendezés (*Selection sort*)
 - Beszúró rendezés (*Insertion sort*)
 - Kupacrendezés (*Heap sort*)
 - Gyorsrendezés (*Quick sort*)
2. Összehasonlító rendezést végző
 - A felsorolt helyben rendező algoritmusok
 - Összefésülő rendezés (*Merge sort*)
3. Lineáris időben rendező
 - Leszámláló rendezés (*Counting sort*)
 - Számjegyes rendezés (*Radix sort*)
 - Edényrendezés (*Bucket sort*)

Ajánlott irodalom:

T. H. Cormen, C. E. Leiserson, R. L. Rivest: *Algoritmusok*, Műszaki Könyvkiadó, Bp. 1999.

Buborék rendezés: párosával összehasonlítja az elemeket, s ha nem a megfelelő sorrendben követik egymást, akkor felcserélődnek. Addig ismétlődik az eljárás, amíg további cserére nincs szükség.

```
BEGIN
  INPUT: a[N]
  ind := 0
  WHILE ind=0
    i := 1
    WHILE i <=N-1
      IF a[i] > a[i+1]
        s := a[i]
        a[i] := a[i+1]
        a[i+1] := s
        ind := 1
      END
      i := i+1
    END
  END
  OUTPUT: a[N]
END
```

```
BEGIN
  INPUT: a[N]
  i := 1
  WHILE i <= N-1
    j := i+1
    WHILE j <= N
      IF a[j] < a[i]           //növekvő sorrend esetén
        s := a[i]
        a[i] := a[j]
        a[j] := s
      END
    END
    i := i+1
  END
  OUTPUT: a[N]
END
```

Beszűrő rendezés: (rendezés közvetlen beszúrással) a kártyalapok rendezéséhez hasonló az algoritmus. Üres bal kézzel kezdünk, a lapok színükkel lefelé az asztalon vannak. Egy lapot felvesszünk az asztalról és elhelyezzük a bal kezünkben a megfelelő helyen. A hely megtalálásához összehasonlítjuk a felvett lapot a kezünkben lévőkkel egyesével sorban jobbról balra. Hatékony kisszámú elem rendezésére.

```
BEGIN
  INPUT: a[N]
  i := 2
  WHILE i <= N
    s := a[i]
    j := i-1
    WHILE (j>0) AND (s<a[j])
      a[j+1] := a[j]
      j := j-1
    END
    a[j+1] := s
    i := i+1
  END
  OUTPUT: a[N]
END
```

Shell rendezés (rendezés fogyó növekménnyel):

A buborékrendezés cserélési fázisát gyorsítja meg azzal, hogy egymástól távol lévő elemeket cserél fel, s a cserélendő elemek közti távolságot fokozatosan csökkenti, egészen 1-ig. Mire a növekmény 1 lesz, az elemek a helyükre kerültek a beszűrős rendezés algoritmusá alapján.

```
BEGIN
  INPUT: a[n], h[t]
  s := t
  WHILE s >= 1
    h := h[s]
    i := h+1
    WHILE i <= n
      k := a[i]
      j := i-h
      WHILE (j>0) AND (a[j]>k)
        a[j+h] := a[j]
        i := i-h
      END
      a[j+h] := k
      i := i+1
    END
    s := s-1
  END
  OUTPUT: a[N]
END
```

Minimum-/maximumkiválasztásos rendezés:

először a legkisebb/legnagyobb elemet határozzuk meg, majd a következő legkisebbet/legnagyobbat s így tovább. Tehát az aktuális első elemet a mögötte lévők közül csak a legkisebbel/legnagyobbval cseréljük ki.

```
BEGIN
  INPUT: a[N]
  i := 1
  WHILE i <= N-1
    ind := i
    j := i+1
    WHILE j <= N
      IF a[ind] > a[j]                //min. kiválasztás
        ind := j
      END
      j := j+1
    END
    s := a[i]
    a[i] := a[ind]
    a[ind] := s
    i := i+1
  END
  OUTPUT: a[N]
END
```

Számláló rendezés:

Különböző elemek esetében minden elemre megszámoljuk a nála kisebb elemeket. Ha $k-1$ db kisebb elem van egy adott elemnél, akkor az adott elem a k -adik helyre kerül.

```
BEGIN
  INPUT: a[N]
  i := 1
  WHILE i <= N
    k := 1
    j := 1
    WHILE j <= N
      IF a[i] > a[j]
        k := k+1
      END
      j := j+1
    END
    b[k] = a[i]
    i := i+1
  END
  OUTPUT: b[N]
END
```

Házi feladat: Elolvasni, önállóan feldolgozni a két „Rendezési algoritmusok” című segédletet.