# Algoritmusokkal kapcsolatos alapfogalmak

## Pszeudokód

A képen diagram, kör, vázlat, sor látható

Automatikusan generált leírásA pszeudokódban eltolással jelöljük a blokkokat. 3 db használható ciklus szerepel benne, a for, a while és a repeat-until. Feltételes utasítás esetén az if-else-t használjuk. Megjegyzéseket pedig a // segítségével tehetünk a programban.

## Bemenet mérete és futási idő

A **bemenet méretének** legjobb mértéke a vizsgált feladattól függ. Például rendezésnél a rendező tömb n elemszáma lehet a mérték. Néha megfelelőbb a bemenet méretét egy szám helyett inkább kettővel leírni. Pl. ha az algoritmus bemenete egy gráf, a bemenet méretét leírhatjuk a gráf éleinek és csúcsainak számával.  
Az **algoritmusok futási ideje** egy bizonyos bemenetre a végrehajtott alapműveletek vagy „lépések” száma. A pszeudokód mindegyik sorának végrehajtásához állandó mennyiségű idő szükséges.

### Vizsgáljuk meg a 1+2+…+n függvényt

**Osszead**()

x=0;

**for** i=0 to n

{

x=x+1

}

**return** x

Elindul az összeadás metódus. Létrehozunk egy x változót, aminek a kezdőértéke 0.

Ezután írunk egy ciklust, ami 100-ig megy, ezen belül megadjuk, hogy x legyen egyenlő x+1, ezzel hozzáadjuk, mindig a kövi számot. Végül a függvény visszatér x-el.

#### Gauss megoldása

Ahhoz, hogy összeadjuk 1-től n-ig a számokat Gauss kitalált egy képletet, aminek a segítségével sokkal gyorsabban össze lehet adni a számokat.

#### A képen szöveg, sor, diagram, Diagram látható Automatikusan generált leírásHasonlítsuk össze a kettőt

Az első képletnek lineáris futási ideje volt, mivel a bemenet mérete egyenesen arányos a futási idővel.

Míg a Gauss képlete segítségével egy olyan pszeudokódot tudunk írni, aminek a lépésszáma 1, tehát akármennyi a bemenet konstans időben fog lefutni.

### Keresés

Bemenetként megadunk egy tömböt (A[n]), és egy x számot. Kimenetként pedig szeretnénk megkapni az x szám A tömbbeli pozícióját.

#### Lineáris keresés

A képen szöveg, sor, diagram, Diagram látható

Automatikusan generált leírásLineáris kereséskor 2 féle lépésszámról kell beszélni az átlagos ((n+1)/2) és a legrosszabb esetről (n).

#### Bináris keresés

Ebben az esetben a legrosszabb eset a log2(n)+1. -> A függvény egyre kevésbé növekszik a bemenethez képest.

# Rendezés

**Bemenet**: Számok véges (n>=2 elemű sorozata) <a1, a2, …, an>  
**Kimenet**: A bemenet azon <a’1, a’2, …, a’n> permutációja, melyre a’1 <= a’2 <= a’n.  
Szeretnénk, hogy ez alapján egy rendezett tömböt kapjunk, erre használhatjuk a beszúró rendezést.

## A képen szöveg, Betűtípus, kézírás, fehér látható Automatikusan generált leírásBeszúró rendezés

**A beszúró rendezés alapgondolata, hogy végig megyünk az elemeken és úgy rendezzük sorba őket, hogy megvizsgáljuk, hogy az előtte lévő elemek kisebbek-e, és ha igen akkor megcseréljük őket**.

Beszúró rendezés esetén **minden sor végrehajtása egy konstans sor**. Amikor a futási idejét vizsgáljuk egy pszeudokódnak akkor, 2 dolgot kell figyelembe venni: a konstansok számát, ami jelen esetben a sorok száma, és az időt, ami azt mutatja meg, hogy egyes sorokat hányszor kell végrehajtani. **Fontos megjegyezni, hogy a megjegyzéskor a konstans értéke 0,** mivel amikor elindul a fordítás akkor törlődnek a sorok.

### Legjobb eset

Akkor beszélünk a legjobb esetről, ha a bemeneti tömb már eleve rendezve van. Ebben az esetben a futási ideje **T(n) = a\*n+b.**

### Legrosszabb eset

Ebben az esetben a tömb elemei fordítottan vannak rendezve. **T(n)=a\*n2+b\*n+c**

# Halmaz, multihalmaz, tömb

Az adatszerkezet adatokból, a köztük lévő kapcsolatokból, a rajtuk végrehajtható műveletekből és a memóriában illetve a háttértáron történő fizikai tárolás megvalósításából áll.   
Az absztrakt adattípus az adatelemek és a köztük lévő kapcsolatok logikai modellje.

**Műveletek**

Létrehozás, módosítás (elem hozzáadás, csere, törlés), és az elem elérése. Néhány adattípust lehet rendezni, lehet bennük keresni, be lehet őket járni és fel lehet őket dolgozni.

**Reprezentáció**

Adatszerkezeteknél kétféle reprezentációról beszélhetünk folytonos és láncolt.

**Osztályozás**

Típus alapján osztályozhatjuk őket homogén (ebben az esetben azonos típusú elemekből áll) és heterogén (különböző típusú elemek). Ha az elemek számát nézzük, akkor lehet dinamikus(változhat az elemek száma) és statikus (ebben az esetben az elemek száma fix). Ha kapcsolat alapján szeretnénk osztályozni, akkor lehet struktúra nélküli, asszociatív, szekvenciális (mindig van következő elem), hierarchikus vagy hálós.

## Halmaz

*A halmaz egymástól megkülönböztethető elemekből áll. Ha x az S halmaz eleme, akkor a következő jelölést használjuk. x∈ S. A halmaz elemeit felsorolással is megadhatjuk S = {1,2,3}.*

A halmaz homogén, dinamikus, struktúra nélküli folytonos reprezentációval ábrázolt adatszerkezet.

***Műveletek***

Adatszerkezetet folytonos reprezentációval hozzuk létre. Módosítani tudjuk elem hozzáadással (unió -> egyelemű halmaz hozzáadása), elem törléssel (különbség -> egynemű halmaz kivonása), elem cserével (különbség, majd unió). Az elemet el is tudjuk érni ∈.

## Tömb

A tömb olyan adattípus, mely előre meghatározott, konstans számú elemből áll, és minden elemet egyértelműen azonosít egy egész számokból álló véges sorozat. Ezen sorozat elemeinek száma a tömb dimenziója. Az egydimenzós tömböt vektornak hívjuk, és a számot, amely az adott elem tömbön belüli helyzetét mutatja, az adott elem indexének nevezzük.

A tömb homogén, statikus, asszociatív, folytonos reprezentációval ábrázolt adatszerkezet.

## Mátrix

A kétdimenziós tömböt mátrixnak hívjuk, és minden elemet 2 számjegyből álló sorozattal azonosítunk.

## Vektor

A vektor ábrázolását tekintjük a legtermészetesebb folytonos reprezentációnak