**Matematikai és valószínűségszámítási ismeretek**

**2. tétel**

**1. rész**

Valószínűség fogalma és kiszámításának kombinatorikus módszerei (permutációk, variációk, kombinációk). Feltételes valószínűség, függetlenség, Bayes-formula.

**Valószínűség**

Elemi események: egy kísérlet lehetséges kimenetelei

Esemény: elemi eseményekből álló halmazok, jele: A, B, C …

Eseménytér: egy kísérlethez tartozó összes elemi esemény, jele: Ω

Valószínűség: tekintsünk egy kísérletet, és ehhez kapcsolódva egy **A** eseményt. Hajtsuk végre a kísérletet *n*-szer egymástól függetlenül, azonos körülmények között. Jelölje  az **A** bekövetkezései számát. Ha a  relatív gyakoriság nagy *n* esetén egy fix szám körül ingadozik, akkor ezt az **A** -ra jellemző számot **P(A)**-val jelöljük és **A** valószínűségének nevezzük.

Axiómák:

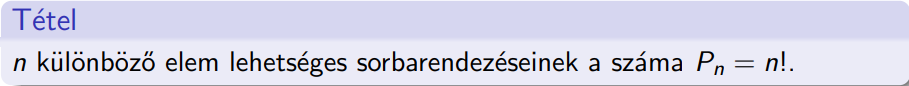
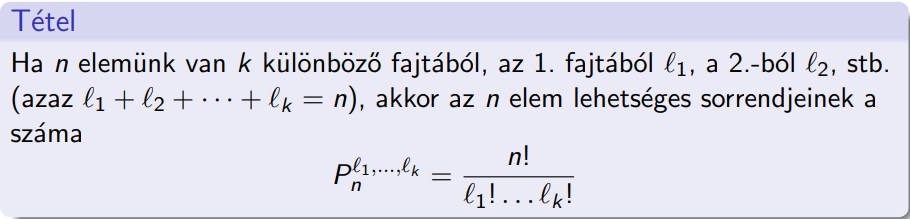
1. 
2.  (A biztos esemény mindig bekövetkezik)
3.  (Ha A és B egymást kizáró események)

Klasszikus kiszámítási módja:

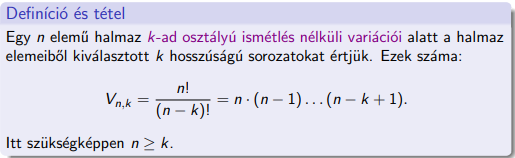
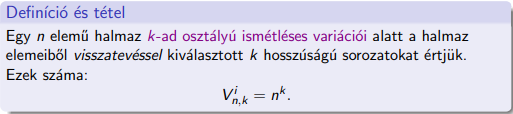


**Kombinatorikus kiszámítási módszerek**

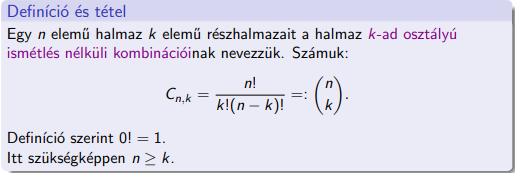
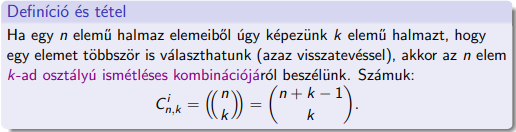
Permutáció: egy A halmaz önmagára vett bijektív leképezése, vagy A elemeinek valamilyen sorrendben való felsorolása (= sorba rendezés)

* ismétlés nélküli 
* ismétléses 

Variáció: *n* elemű halmazból kiválasztott *k* hosszúságú sorozatok (= kiválasztás és sorba rendezés)

* ismétlés nélküli 
* ismétléses 

Kombináció: *n* elemű halmaz *k* elemű részhalmazai (= kiválasztás)

* ismétlés nélküli 
* ismétléses 

**Feltételes valószínűség**

Legyen A és B esemény, P(B) > 0. Ekkor az A esemény B-re vonatkozó feltételes valószínűségén a   
mennyiséget értjük.

**Függetlenség**

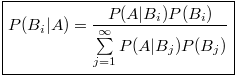
Azt mondjuk, hogy A és B független események, ha

Jelentése: Egyik esemény bekövetkezése sem befolyásolja a másik bekövetkezési esélyét.

**Bayes-formula**

A Bayes-tétel a valószínűségszámításban egy feltételes valószínűség és a fordítottja között állít fel kapcsolatot. A valamiféle hipotézis, B egy megfigyelhető esemény és a tétel azt adja meg, hogyan erősíti vagy gyengíti az esemény megfigyelése a hipotézis helyességébe vetett hitünket.

A formula: 

Legyen A egy esemény, B1, B2, … teljes eseményrendszer, P(A) > 0, P(Bi) > 0, i = 1,2…. Ekkor minden j-re.

**2. rész**

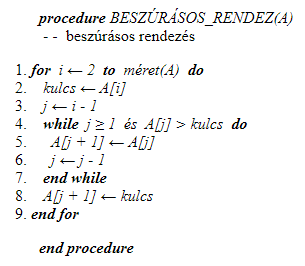
Algoritmusok lépésszáma: beszúrásos rendezés, összefésüléses rendezés, keresések lineáris és logaritmikus lépésszámmal. Gyorsrendezés, az összehasonlítások minimális száma. Rendezés lineáris lépésszámmal: radix rendezés, vödör rendezés.

**Beszúrásos rendezés**

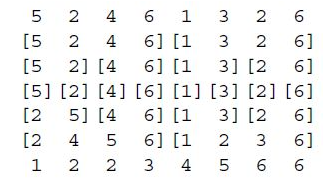
A rendezés lényege: a tömb második elemétől indulva lépked végig az elemeken, ellenőrizzük, hogy az adott elem kisebb-e az előtte lévő elemnél. Ha kisebb, akkor egyesével addig léptetjük a tömbben az elemet, amíg előtte kisebb, utána nagyobb szám lesz. Ha nagyobb, akkor nem történik helycsere. Nagy tömbök esetén nem hatékony, viszont kis tömböknél a leghatékonyabb

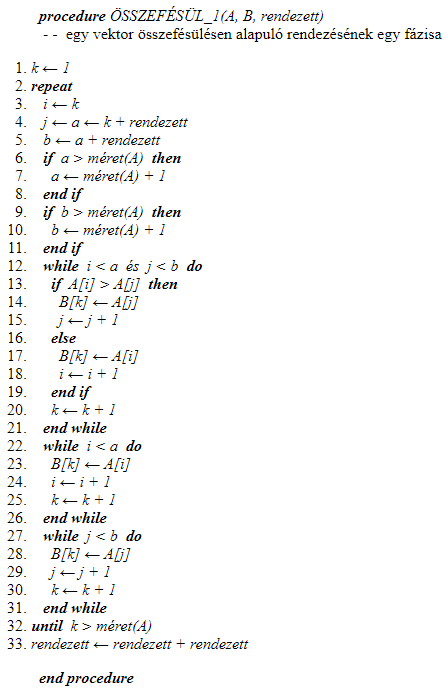
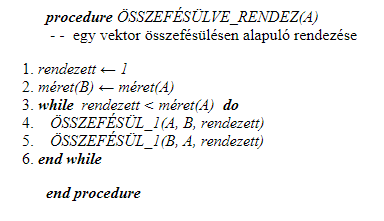
Lépésszám, időbonyolultság:

* Legrosszabb eset: O(n2) 🡪 ha pont fordítva van rendezve a kiindulási tömb
* Legjobb eset: O(n) 🡪 ha a kiindulási tömb eleve rendezve van



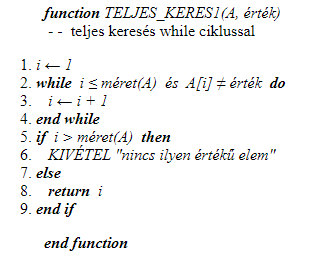
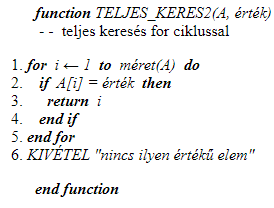
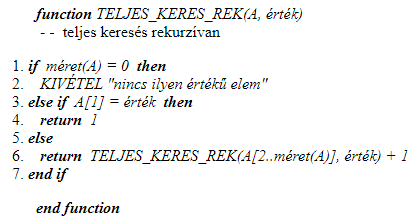
**Összefésüléses rendezés**

A rendezés lényege: a tömböt felosztjuk két részre, a részeket külön rendezzük, majd összefésüljük. Ez rekurzívan történik, tehát egészen addig osztjuk 2 részre a résztömböket, amíg egy elemű tömbök maradnak. Ezeket kell párosával összefésülni. Ennek lényege, hogy a két résztömb soron következő elemeit hasonlítja össze, így készítve egy új összefésült tömböt. Ezt egészen addig ismételve, míg az eredeti tömbünk rendezett változatát kapjuk vissza. Példa: 

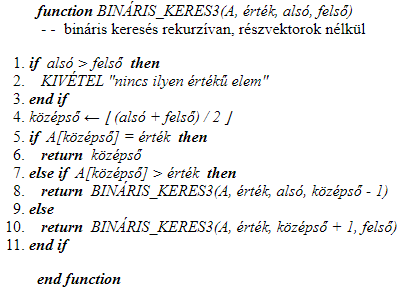
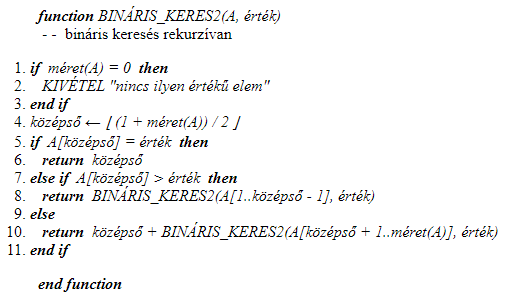
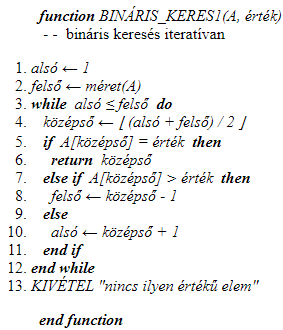
Lépésszám, időbonyolultság: O(n \* log(n)) 🡪 ahol log(n) a felosztások/szintek száma; bizonyos helyzetekben gyorsabb is lehet, mint a gyorsrendezés, viszont hátránya a magas tárterület igénye a felosztások miatt (nem helyben rendez) 

**Keresések**

Lineáris keresés: a tömb elemeinek iterálása az elejétől egészen a keresett elem megtalálásáig

Rendezetlen tömbön is működik.

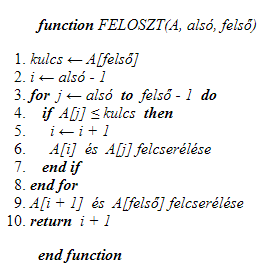
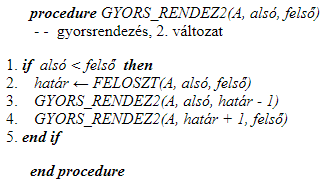
Lépésszám, időbonyolultság: O(n) 🡪 ezért is hívják lineáris keresésnek, mert a lépészám lineárisan függ a tömb elemszámától

Bináris keresés: csak rendezett tömbön! Megvizsgálja a középső elemet, ha nem az a keresett, akkor, ha annál nagyobb, akkor a középső elem utáni résztömbben keres, ha kisebb, akkor a középső elem előtti résztömbben, ugyanilyen elven. 

Lépésszám, időbonyolultság: O(log(n)) 🡪 nagy elemszámú tömbök esetén lényegesen gyorsabb lehet, mint a lineáris

**Gyorsrendezés**

A rendezés lényege: Kiválasztunk egy kitüntetett elemet (pivot), majd a tőle kisebb vagy egyenlő elemeket tőle jobbra, a nagyobb elemeket tőle balra helyezzük el. Ekkor a pivot elem a végleges sorrendet tekintve a helyén van. Ezt követően a pivot előtti és utána résztömbön is elvégezzük ezt az eljárást (rekurzív).

Itt a kitüntetett elem a tömb utolsó eleme. Az ’alsó’ és a ’felső’ változók a paraméterként adott tömb első és utolsó indexei. Ezen eljárás során a tömb elejére kerülnek a pivotnál kisebb vagy egyenlő elemek. a ’FELOSZT’ eljárás 9. sorában az utolsó helyen álló pivotot cseréljük fel a sorrendben első pivotnál nagyobb elemmel. Ennek a helyére került pivotnak az indexét kapja vissza a ’GYORS\_RENDEZ’ eljárás, majd az rekurzívan hívja magát a pivot előtt és utáni résztömbökre.

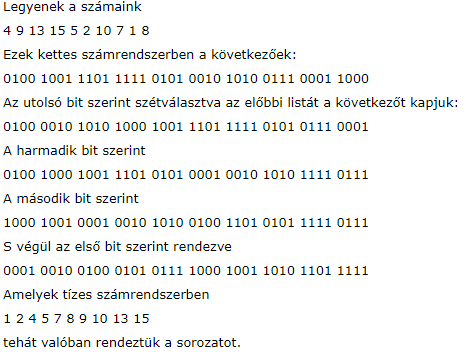
(Kicsit másik módszer, de részletes, vizuális magyarázattal: <https://www.youtube.com/watch?v=Hoixgm4-P4M>)

Bonyolultság:

* legrosszabb eset: O(n2) 🡪 például akkor, ha a pivot mindig a legnagyobb eleme a tömbnek
* legjobb eset: O(n \* log(n)) 🡪 egyenlő, vagy közel egyenlő felosztás esetén

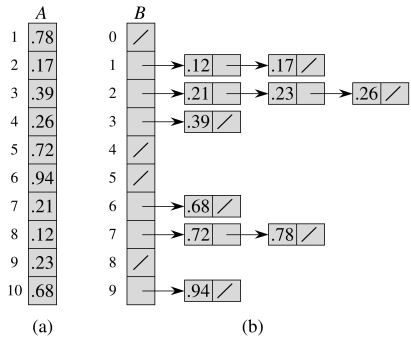
**Radix(számjegyes) rendezés**

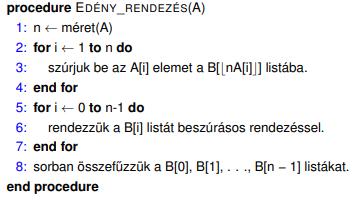
Feltételezzük, hogy a rendezni kívánt tömbünk minden eleme ugyanannyi számjegyből áll, majd a legkisebb helyiértéktől haladva a legnagyobb felé, helyiértékenként rendezzük a tömböt egy választott stabil algoritmussal (pl.: leszámláló rendezés).



Bonyolultság: O(*d* \* (*n + k)*), ahol *d* a számjegyek száma, *n* a rendezni kívánt elemek száma, *k* pedig a lehetséges számjegyek száma (lineáris idejű)

**Vödör(edény) rendezés**

Feltételezzük, hogy a rendezni kívánt *n* értékekre igaz, hogy: 0 ≤ *n* < 1 és az értékek egyenletes eloszlásból származnak. A vödrök láncolt listák lesznek. Ezekben helyezzük el az elemeket az első tizedes jegy alapján, majd az egyes vödrökben beszúrásos rendezéssel rendezzük az elemeket. Az eljárás végén pedig összefűzzük a rendezett vödrök tartalmát. 

Bonyolultság: a bonyolultság a vödrök rendezési algoritmusától függ

* Legrosszabb eset: O(n2) 🡪 ha vödrök elemszáma nagyban eltér, és a szétválogatást követően a vödrökben lévő elemek fordított sorrendben vannak
* Legjobb eset: O(n) 🡪 egyenletes eloszlású számokkal, és ha a szétválogatást követően már eleve rendezett vödröket kapunk

Összességében a vödör rendezés lineáris, egészen addig, amíg az edényméretek négyzeteinek összege lineáris a teljes elemszámban.