

# Záróvizsga jegyzet - PTI BSc

2017 után felvett hallgatóknak

2023

**Fontos:** ez csupán egy jegyzet, amiben szerepelnek általam megfogalmazott részek is, ezért előfordulhatnak pontatlanságok is. Tehát a jegyzet ily módon kezelendő.

**UI:** Sok sikert a záróvizsgán!

# Tartalomjegyzék

<b>1. Tétel</b>	<b>4</b>
1.1. Diszkrét és folytonos valószínűségi eloszlás fogalma. Nevezetes eloszlások: binomiális, Poisson, egyenletes, exponenciális, normális.	4
1.2. Adatszerkezetekkel kapcsolatos alapfogalmak: absztrakció (logikai és fizikai szint), absztrakt adatszerkezetek (homogén-heterogén, statikus-dinamikus, struktúra, műveletek). Elemi adatszerkezetek: lista, verem, sor. Halmaz, multihalmaz, mátrix. Fák ábrázolása, keresések, bejárások, törlés, beszúrás.	8
<b>2. Tétel</b>	<b>16</b>
2.1. Valószínűség fogalma és kiszámításának kombinatorikus módszerei (permutációk, variációk, kombinációk). Feltételes valószínűség, függetlenség, Bayes-formula.	16
2.2. Algoritmusok lépésszáma: beszúrásos rendezés, összefésüléssel rendezés, keresések lineáris és logaritmikus lépésszámmal. Gyorsrendezés, az összehasonlítások minimális száma. Rendezés lineáris lépésszámmal: radix rendezés, vödör rendezés.	19
<b>3. Téma</b>	<b>23</b>
3.1. Függvények szélsőértéke, függvényvizsgálat. A legkisebb négyzetek módszere	23
3.2. Az elsőrendű logika nyelvének szintaxisa. Változók kötött és szabad előfordulása. A nyelv interpretációja, változókiértékelés. Termek és formulák értéke interpretációban, változókiértékelés mellett. Törvény, ellentmondás, ekvivalencia, következmény. Normálformák, prenex formulák. Logikai kalkulusok.	27
<b>4. Tétel</b>	<b>33</b>
4.1. Függvények, görbék, felületek leírása és számítógépes ábrázolása.	33
4.2. Problémák reprezentálása állapottéren. A megoldás keresése visszalépéssel. Szisztematikus és heurisztikus gráfkereső eljárások: a szélességi, a mélységi és az A algoritmusok. Kétszemélyes játékok és reprezentálásuk. A nyerő stratégia. Lépésajánló algoritmusok.	37
<b>5. Tétel</b>	<b>42</b>

5.1.	Mátrix fogalma, műveletek, determináns, rang. Speciális mátrixok, inverz. Mátrix, mint lineáris transzformáció. Sajátérték, sajátvektor. . . . .	42
5.2.	A problémaredukciós reprezentáció és az ÉS/VAGY gráfok. Ismeretreprezentációs technikák, bizonytalanság-kezelés (fuzzy logika). A rezolúciós kalkulus. A logikai program és az SLD rezolúció. A logikai programozás alapvető módszerei.	45
<b>6.</b>	<b>Tétel . . . . .</b>	<b>49</b>
6.1.	Gráf fogalma és megadásának módjai. Egyszerű, irányított és irányítatlan gráfok. Séta, út, összefüggőség. Nevezetes gráfok: páros gráf, teljes gráf, fa, kör, súlyozott gráf. . . . .	49
6.2.	Generatív nyelvtanok, nyelvosztályok, a Chomsky-hierarchia. Véges automaták, lineáris idejű felismerés, veremautomaták. . . . .	52
<b>7.</b>	<b>Tétel . . . . .</b>	<b>58</b>
7.1.	Lineáris egyenletrendszer fogalma és megoldása Gauss eliminációval. . . . .	58
7.2.	Determinisztikus Turing-gépek, lineárisan korlátozott automaták, eldönthetetlen problémák, tár és idő korlátok. Nemdeterminisztikus Turing-gépek, nevezetes nyelvosztályok, P, NP. . . . .	60
<b>8.</b>	<b>Tétel . . . . .</b>	<b>64</b>
8.1.	Statisztikai minta és becslések, átlag és szórás. Konfidenciaintervallumok. Az $u$ -próba. . . . .	64
8.2.	Az informatikai biztonság fogalma, legfontosabb biztonsági célok. Fizikai, emberi, technikai fenyegetések és ellenük való védekezés. Algoritmikus védelem eszközei: titkosítás, digitális aláírás, hash függvények. Az AES és RSA algoritmusok. .	66
<b>9.</b>	<b>Tétel: Adatbázisrendszerek. Adatbázis, adatbázisrendszer, adatbáziskezelő rendszer (DBMS) fogalma és jellemzői. Egyed, tulajdonság és kapcsolat fogalma és tulajdonságai. Relációs, objektum-relációs és NoSQL adatbázisok jellemzése. A funkcionális függés fogalma. Koncepcionális adatbázis-tervezés, az ER modell és leképezése relációs modellre. Az SQL elemei: DDL, DML, DCL, egyszerű lekérdezések és táblák összekapcsolása.</b>	<b>72</b>

10.Tétel: Lexikális egységek. Adattípusok. Nevesített konstans. Változó. Kifejezések. Utasítások. Programegységek. Paraméterkiértékelés, paraméterátadás. Blokk. Hatáskörkezelés, láthatóság. Absztrakt adattípus. Kivételkezelés. . . . .	78
11.Tétel: Az objektumorientált paradigma alapfogalmai. Osztály, objektum, példányosítás. Öröklődés, osztályhierarchia. Polimorfizmus, metódustúlterhelés. A bezárási eszközrendszer. Absztrakt osztályok és interfészek. Típus-tagok. . . . .	82
12.Tétel: Operációs rendszerek fogalma, felépítése, osztályozásuk. Fájlok és fájlrendszerek. Speciális fájlok Unix alatt. Átirányítás, csővezetékek. Folyamatkezelés. Jelzések, szignálok. Ütemezett végrehajtás . . . . .	86
13.Tétel: Verziókezelés, verziókezelő rendszerek. Szoftvertesztelési alapfogalmak (tesztszintek, tesztípusok, teszttervezési módszerek). Objektumorientált tervezési alapelvek (GoF, SOLID). Függőségbefecskendezés. Architektúrális minták (MVC). Tervezési minták. Szabad és nem szabad szoftverek. Szoftverlicencek, szabad és nyílt forrású licencek fajtái . . . . .	92
14.Tétel: Hagyományos szoftverfejlesztési módszertanok: vízesés modell, V-modell, spirális fejlesztési modell, prototípus alapú fejlesztés, iteratív és inkrementális módszertanok, gyors alkalmazásfejlesztés. Agilis szoftverfejlesztési módszertanok: az agilis szoftverfejlesztés alapjai, az agilis kiáltvány, valamint egy szabadon választott agilis módszertan részletes bemutatása . .	99
15.Tétel: A web működésének alapjai. Web szabványok és szabványügyi szervezetek. URI-k és felépítésük. HTTP: kérések és válaszok felépítése, metódusok, állapotkódok, tartalomgyezytetés, sütik. A web jelölőnyelvei: XML és HTML dokumentumok felépítése. Stíluslap nyelvek. JSON . . . . .	106
16.Számítógép-hálózatok osztályozási szempontjai. Hálózati rétegmodellek. IP technológia címzési rendszere, és vezérlése. Forgalomirányítás elve és az útválasztási kategóriák jellemzése. TCP és UDP mechanizmusok. . . . .	114

# Matematikai és számítástudományi ismeretek

## 1. Tétel

**1.1. Diszkrét és folytonos valószínűségi eloszlás fogalma. Nevezetes eloszlások:**  
binomiális, Poisson, egyenletes, exponenciális, normális.

**Valószínűségi eloszlás:** **(definíció)** Egy  $p_1, p_2, \dots$  sorozatot valószínűségi eloszlásnak nevezünk, ha  $p_i \geq 0, \sum_{i=1}^{\infty} p_i = 1$ .

**Eloszlásfüggvény:** **(definíció)** Egy  $\xi : \Omega \rightarrow \mathbb{R}$  valószínűségi változó **eloszlásfüggvényén** azt az  $F_\xi(x)$  függvényt értjük, melyre teljesül:

$$F_\xi(x) = P(\omega \in \Omega : \xi(\omega) < x)$$

**Eloszlásfüggvény jellemzői:** monoton nemcsökkenő, balról folytonos és  $\lim_{x \rightarrow -\infty} F_\xi(x) = 0$  és  $\lim_{x \rightarrow \infty} F_\xi(x) = 1$

Továbbá, ez a függvény azt adja meg, hogy mennyi a valószínűsége annak, hogy  $\xi < x$ .

**Diszkrét valószínűségi eloszlás:** **(definíció)** Legyen  $\xi$  olyan diszkrét valószínűségi változó, amelynek értékkészlete  $x_1, x_2, \dots$ . Ekkor az  $A_i, i = 1, 2, \dots$ , halmazok teljes eseményrendszert alkotnak. Ebből következik, hogy a

$$p_i = P(A) = P(\xi = x_i), i = 1, 2, \dots$$

számok **diszkrét eloszlást alkotnak**. (azaz  $p_i \geq 0$  és  $\sum_{i=1}^{\infty} p_i = 1$ )

**Diszkrét eloszlások:**

- **Binomiális:** Ha egy kísérletet  $n$ -szer függetlenül megismételünk, és  $\xi$  jelenti a  $p$  valószínűségű  $A$  esemény bekövetkezéseinek számát, akkor

$$P(\xi = k) = \binom{n}{k} p^k (1-p)^{n-k}, k = 0, 1, \dots, n.$$

**Például:** visszatevéses húzás.

Akkor használatos, ha  $\xi$  **korlátos**.

- **Poisson:** Ha  $\xi$  valószínűségi változó,  $\lambda$  paraméterrel *Poisson-eloszlású*, akkor

$$P(\xi = k) = \frac{\lambda^k}{k!} e^{-\lambda}, k = 0, 1, 2, \dots, \text{ ahol } \lambda > 0 \text{ konstans.}$$

Akkor használatos, ha átlagról beszélünk és  $\xi$  **korlátlan**.

- **Hipergeometrikus:**

$$P(\xi = k) = \frac{\binom{K}{k} \binom{N-K}{n-k}}{\binom{N}{n}}$$

Akkor használatos, ha az **összes elem** és **összes selejt** ismert (vagyis  $N, K$ ).

**Folytonos valószínűségi eloszlás:** **(definíció)** A  $\xi$  valószínűségi változó eloszlása folytonos pontosan akkor, ha az eloszlásfüggvénye folytonos függvény.

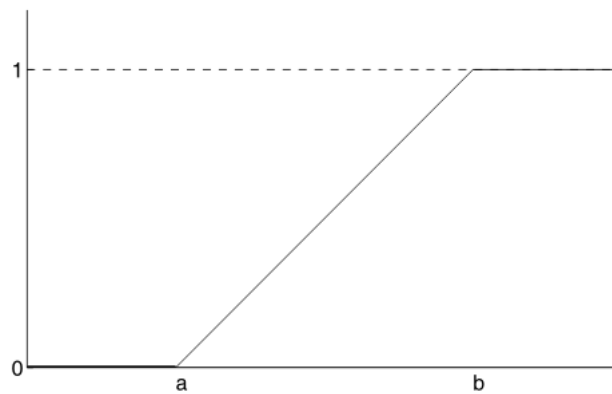
**Folytonos valószínűségi változó:** **(definíció)** Legyen  $(\Omega, F, P)$  egy valószínűségi mező. A  $\xi : \Omega \rightarrow \mathbb{R}$  leképezést **folytonos valószínűségi változónak** nevezzük, ha bármely rögzített  $x \in \mathbb{R}$  esetén

$$\omega \in \Omega : \xi(\omega) < x \in F$$

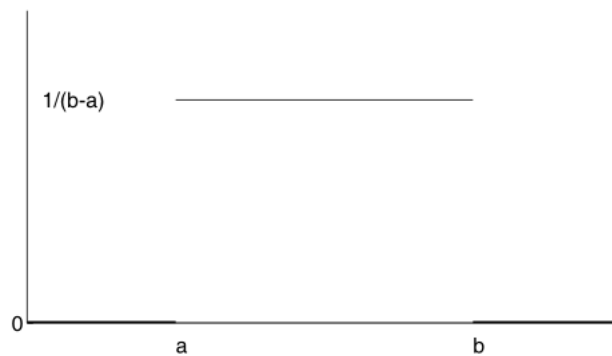
**Folytonos eloszlások:** intervallumok, távolságok és hasonló értékek kiszámolására használatosak.

- **Egyenletes eloszlás:** **(definíció)** Ha egy véges intervallumra úgy dobunk egy pontot, hogy az intervallum bármely részintervallumára annak hosszával arányos valószínűséggel essen, akkor a pont  $x$ -koordinátája egyenletes eloszlású. Nevezetes képletei:

$$\begin{aligned} \text{– Eloszlásfüggvénye: } F(x) &= \begin{cases} 0, & \text{ha } x \leq a \\ \frac{x-a}{b-a} & \text{ha } a < x \leq b \\ 1, & \text{ha } x > b \end{cases} \\ \text{– Sűrűségfüggvénye: } f(x) &= \begin{cases} \frac{1}{b-a}, & \text{ha } a \leq x \leq b \\ 0, & \text{egyébként} \end{cases} \end{aligned}$$



1. ábra. Egyenletes eloszlás eloszlásfüggvénye



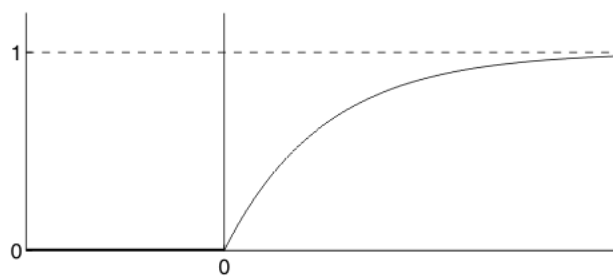
2. ábra. Egyenletes eloszlás sűrűségfüggvénye

- **Exponenciális eloszlás:** **(definíció)** Az exponenciális eloszlás **élettartamok** és **várakozási idők** eloszlásaként lép fel. Az exponenciális eloszlás és a vele kapcsolatos más eloszlások a **sorbanállás-elméletben** és a **megbízhatóság-elméletben** használatosak. Nevezetes képletei:

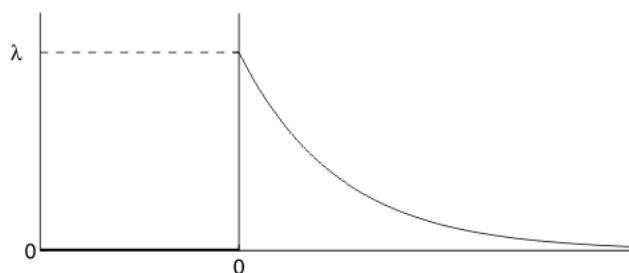
- **Eloszlásfüggvénye:** 
$$F(x) = \begin{cases} 0, & \text{ha } x \leq 0 \\ 1 - e^{-\lambda x}, & \text{ha } x > 0 \end{cases}$$
- **Sűrűségfüggvénye:** 
$$f(x) = \begin{cases} 0, & \text{ha } x \leq 0 \\ \lambda e^{-\lambda x}, & \text{ha } x > 0 \end{cases}$$

Exponenciális eloszlás jellemzői:

- **Örökifjúság:** ha a  $t$  életkort elérte, akkor ugyanolyan eséllyel él még  $s$  ideig, mintha éppen akkor született volna.
- **Normális eloszlás:** **(definíció)** A normális eloszláson alapul a **statisztika klasszikus elméletének** túlnyomó része. A valószínűségi változót normális eloszlásúnak



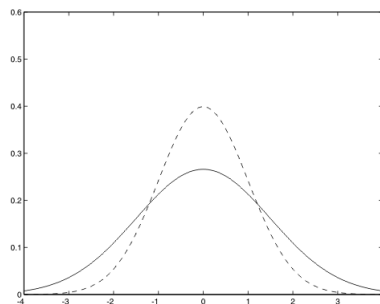
**3. ábra.** Exponenciális eloszlás eloszlásfüggvénye



**4. ábra.** Exponenciális eloszlás sűrűségfüggvénye

nevezzük, ha sűrűségfüggvénye:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-m)^2}{2\sigma^2}}, \text{ ahol } m \in \mathbb{R}, \sigma > 0.$$



**5. ábra.** Normális sűrűségfüggvények különböző szórásokra

Mivel a normális eloszlás sűrűségfüggvénye nem integrálható, ezért bevezetünk egy *speciális normális eloszlást*, aminek várható értéke 0, a szórása pedig 1. Ezt **standard normális eloszlásnak** nevezzük.

A standard normális eloszlás sűrűségfüggvénye így a következő:  $f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ .



**1.2. Adatszerkezetekkel kapcsolatos alapfogalmak:** absztrakció (logikai és fizikai szint), absztrakt adatszerkezetek (homogén-heterogén, statikus-dinamikus, struktúra, műveletek). Elemi adatszerkezetek: lista, verem, sor. Halmaz, multihalmaz, mátrix. Fák ábrázolása, keresések, bejárások, törlés, beszúrás.

Az adatelemek lehetnek **egyszerűek** (atomiak) és **összetettek**. Minden adatelem rendelkezik valamilyen **értékkel**.

**Logikai absztrakció:** **(definíció)** Az adatelemek között jól meghatározott kapcsolatrendszer van. Az adatelemek és a közöttük lévő kapcsolatok definiálják a **logikai** (*absztrakt*) adatszerkezetet. Független hardvertől, szoftvertől.

**Fizikai absztrakció** (*társzerkezet*): **(definíció)** Adatszerkezet az **operatív tárban** vagy **periférián** (*háttértáron*).

---

#### **Absztrakt adatszerkezetek osztályozása:**

1. Változhat-e az elemek száma?

- statikus
- dinamikus

2. Változhat-e az elemek típusa?

- homogén
- heterogén

3. Milyen kapcsolatban állnak egymással az adatelemek?

Egy **homogén adatszerkezet** lehet:

- struktúra nélküli
- asszociatív
- szekvenciális
- hierarchikus
- hálós

A heterogén adatszerkezeteket **nem csoportosítjuk** ilyen szempont alapján.

**Műveletek:** létrehozás, módosítás (bővítés, törlés – fizikai vagy logikai –, csere), rendezés, keresés, elérés, bejárás, feldolgozás.

**Ábrázolási (tárolási) módok** (*memóriabeli*):

1. **Folytonos** (*vektorszerű*): összefüggő folytonos tárterület, azonos méretekkel.
  2. **Szétszórt** (*láncolt*): minden elem esetében az **értéket** és egy, a **következő elemre mutató memóriacímet** is tároljuk. A tárhelyek mérete nem feltétlen azonos.
- 

**Elemi adatszerkezetek:**

- **Lista:**

Szétszórt reprezentáció fajtái:

- **Egyirányba láncolt lista:** a listaelem az *értéket* és a *következő* elemre mutató *memóriacímet* tartalmazza.
- **Kétirányba láncolt lista:** hasonló az egyirányúhoz, azzal a különbséggel, hogy a mutatórész két részből áll: az *előző* és a *következő* elemre mutató *memóriacímekből*.
- **Cirkuláris lista:** hasonló az egyirányúhoz, azzal a különbséggel, hogy az *utolsó elem az elsőhöz van láncolva*.
- **Multilista:** a listaelemek adatrésze **összetett**.

Pl. az adatrész minden komponensére felépíthető egy egyirányba láncolt lista.

- **Verem:**

- speciális lista adatszerkezet
- **LIFO** adatszerkezet (*Last In First Out*) – vagyis csak a tetejére lehet berakni, és csak onnan lehet kivenni is.

- **Sor:**

- speciális lista adatszerkezet
  - **műveletei:** első elemhez hozzáférése (*ACCESS HEAD*) vagy törlése (*GET/-POP*) és utolsó elem mögé szúrás (*PUT/INJECT*)
  - **FIFO** adatszerkezet (*First In First Out*)
  - **további speciális sorok:** kétvégű sorok (aljuknál összeragasztott vermeknek tekinthetőek), prioritásos sorok
- 

### Halmaz és multihalmaz:

- struktúra nélküli (nincs az elemek között kapcsolat), homogén és dinamikus
- a halmaz minden eleme különböző, a multihalmazban előfordulhatnak azonos elemek
- **műveletei:** eleme, unió, metszet, különbség
- **ábrázolásuk:** folytonosan, *karakterisztikus függvénnyel*
  - A **halmaz** lehetséges elemeit sorbarendezzük, és egy **bitsorban** jelöljük az adott elemnél hogy benne van-e (*1*) vagy nincs (*0*).
  - A **multihalmaz** lehetséges elemeit sorbarendezzük, és egy **bájsorban** tároljuk az adott elem előfordulásának számát ( $0, 1, \dots, n$ ).

### Mátrix:

- két- vagy több dimenziós tömb
- statikus, homogén és asszociatív
- az adatelemek **helyzete** a lényeges
- minden adatelemhez különböző egészszámsorozat (ennek számjait *indexnek* nevezzük – ezek darabszáma határozza meg a tömb *dimenzióját*) tartozik, így az asszociativitást biztosító részhalmazok egyeleműek és diszjunktak
- folytonos reprezentációnál leképezése lehet **sorfolytonos** vagy **oszlopfolytonos**

- **típusai:** felsőháromszög-mátrix, alsóháromszög-mátrix, szimmetrikus-mátrix, ritka-mátrix, egység-mátrix, null-mátrix
- 

**Fa:**

- homogén, dinamikus és hierarchikus
- **fogalmak:** csúcs, csomópont, gyökérelem, levélelem, közbenső elem, él, út, részfa, szint, magasság

**Kupac:**

- a gyermek elem(ek) értéke  $\leq$  szülő elem(ek) értéke
- speciális kupac: **bináris kupac**
- **beszúrás:** balról jobbra feltöltéssel történik, esetenként az elemek értékének cseréjével
- **törlés:** a max elem törlésekor elhelyezzük a kupac legutolsó elemét, és kupacosítunk
- **kupac építés:** balról jobbra felépítünk egy fát a tömb elemeivel, majd a levélelemtől felfele kupacosítunk balról jobbra haladva szintenként
- **kupacosítás:** a gyökértől haladva minden gyermekére ellenőrizzük a *kupac tulajdonságot*, s ha nem felel meg valamelyik, azon az ágon tovább megyünk, cseréljük az adatelemeket és tovább folytatjuk. (?)
- **kupac rendezés:** balról jobbra felépítünk egy bináris fát a tömb elemeivel, **max-kupacot** készítünk belőle, majd a gyökérelemet a kezdetlegesen üres, rendezett tömb elejére tesszük, majd mindig a kupac utolsó levélelemével felülírjuk a gyökérelemet és folytatjuk a fenti lépéseket.

**Rendezetlen és rendezett fák:** A **rendezetlen** fáknál nem lényeges az ugyanazon csúcsból kiinduló élek sorrendje, **rendezett** fáknál viszont igen.

**Minimális magasságú fa:** ha adott számú elemet nem lehetne kisebb magasságú fában

elhelyezni.

**Kiegyensúlyozott fa:** bármely elemére igaz, hogy az elem bal és jobb oldali részfájának magasságkülönbsége legfeljebb 1.

- **Tökéletesen kiegyensúlyozott fa:** bármely elemének bal és jobb oldali részfájában az elemek darabszáma legfeljebb 1-gyel tér el.

**Kifejezésfa:** a levélelemek egy kifejezés operandusait, a nem levél elemek pedig ugyanazon kifejezés operátorait tartalmazzák.

**Bináris fa:** minden adatelemnek *maximum két rákövetkezője* van.

- **típusai:**
  - **Szigorú értelemben vett bináris fa:** minden adatelemnek *0 vagy 2 rákövetkezője* van
  - **Rendezett bináris fa**
- **bejárások:**
  - **Preorder:** gyökér, bal, jobb
  - **Inorder:** bal, gyökér, jobb
  - **Postorder:** bal, jobb, gyökér

**Bináris keresőfa:** a szülőelem jobb gyermekének oldalán nagyobbak, a bal oldalán kisebbek az értékek.

- **bővítés:** ha nincs még elem benne beszúrjuk, ha van akkor olyan irányba megyünk mindig, amerre kell (balra ha kisebb, jobbra ha nagyobb). Mindaddig folytatjuk, amég levélelemhez nem jutottunk.
- **törlés:**
  - Ha levélelem, probléma nélkül kitörölhetjük.
  - Ha egy gyermeke van, akkor felülírjuk a gyermekelem értékével.

- Egyébként, felülírjuk a jobb oldali részfa legkisebb (*legbaloldalibb*) elemével, vagy a bal oldali részfa legnagyobb (*legjobboldalibb*) elemével.

**AVL fa:** vagyis kiegyensúlyozott keresőfa.

- **bővítés:** amennyiben nem romlik el a kiegyensúlyozottság, beszúrjuk levélelemként a fába. Egyébként pedig:
  - **LL**<sup>1</sup> eset: jobbra forgatással
  - **LR**<sup>2</sup> eset: fel + jobbra forgatás
  - **RR**<sup>3</sup> eset: balra forgatással
  - **RL**<sup>4</sup> eset: fel + balra forgatás
- **törlés:** hasonlóan a keresőfákhoz, annyi különbséggel, hogy előfordulhat hogy többször is forgatni kell a fa kiegyensúlyozásához (a bővítéssel ellentétben)

**Piros-fekete fa:** egy olyan bináris keresőfa, amely a következő tulajdonságokkal rendelkezik:

1. Minden csomópontja **piros** vagy **fekete**.
2. A gyökere fekete.
3. Minden (NIL értékű) levele fekete.
4. Ha egy csomópont piros, mindkét rákövetkezője fekete (vagyis, nincs benne két egymást követő piros csomópont).
5. Minden csomópont esetén a bal és jobb oldalon lévő fekete csomópontok száma egyenlő.

---

<sup>1</sup>bal oldali részfa bal oldala

<sup>2</sup>bal oldali részfa jobb oldala

<sup>3</sup>jobb oldali részfa jobb oldala

<sup>4</sup>jobb oldali részfa bal oldala

- **beszúrás:**

- **Okasaki-módszer:** úgy bővítjük mint egy keresőfát, mégpedig piros színezéssel. A következő eshetőségek fordulhatnak elő a beszúrást követően:
  1. Ha a fa továbbra is rendelkezik a piros-fekete tulajdonsággal nincs teendők.
  2. Nem teljesül a 2. tulajdonság, miszerint a gyökérelem fekete. Ez csak akkor fordulhat elő ha eredetileg üres volt a fa. Ekkor csak egyszerűen feketére színezzük az újonnan beszúrandó elemet.
  3. Nem teljesül a 4. tulajdonság, miszerint nem lehet két egymás utáni piros csomópont. Ez csak akkor fordulhat elő, ha piros a beszúrt elem szülője. Ekkor **forgatásokat** és **átszínezéseket** kell végrehajtani.
- **CLRS-módszer:** hasonló az *Okasaki-módszerhez* azzal a különbséggel, hogy másképp kezeli azt az esetet, amikor a beszúrást követően nem teljesül a 4. tulajdonság.

- **törlés:** hasonlóan a keresőfákhoz, forgatással és színezéssel keverve.

**B-fa** (*Bayer-fa*): lapokból (*adatelemeket* és *mutatókat* tartalmaz) felépülő keresőfa.

- a lapokon az adatelemek számát a B-fa **rendje** határozza meg. Ha ez  $N$ , akkor a gyökérlap kivételével, minden lapon **min.**  $N$ , **max.**  $2N$  adatelem helyezkedhet el.
- az adatelemek rendezve szerepelnek a lapokon
- **bővítés:**
  - Ha a beszúrást követően, az adott lapon az adatelemek száma nem haladja meg a  $2N$ -t, akkor rendben vagyunk.
  - Ellenkező esetben, felfele visszük az adott lap középső elemét és ketté osszuk az adott lapot a felvitt betű mentén. Ha a fenti lapon kevesebb mint  $N + 1$  adatelem lenne (és nem gyökérlap), akkor fentebb visszük (?).

- **törlés:**

- Ha levéllapról törölünk, és az elemek mennyisége nem csökken  $N$  alá rendben vagyunk. Ellenben, ha a levéllapon  $N$  alá csökken az elemek mennyisége, a szülőlapból veszünk kölcsön egy elemet vagy egyesítünk, ha nincs elég elem a szülőlapon. (?)
- Ha nem levéllapról törölünk, akkor a törlendő elem **baloldali részfájának legjobboldalibb elemével** írjuk felül, vagy a **jobb oldali részfa legbaloldallibb** elemével. Ekkor két eshetőség történhet:
  - \* Ha van egy szomszédos testvérlap, amin minimum  $N + 1$  elem van, akkor kölcsönveszünk egyet.
  - \* Ha mindkét szomszédos testvérlap  $N$  elemet tartalmaz, akkor **lapösszevonást** kell végrehajtani.



## 2. Tétel

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

Fogalmak:

- **Elemi esemény:** egy kísérlet lehetséges kimenetelei.
- **Esemény:** elemi eseményekből álló halmazok, **jele:**  $A, B, C, \dots$
- **Eseménytér:** egy kísérlethez tartozó összes elemi esemény, **jele:**  $\Omega$
- **Valószínűség:** **(definíció)** Ha  $n$  független kísérletet végzünk egy  $A$  esemény megfigyelésére és  $A$   $k$ -szor következett be, akkor  $k$ -t az  $A$  esemény gyakoriságának, a  $\frac{k}{n}$  értéket pedig  $A$  **relatív gyakoriságának** nevezzük és  $P(A)$ -val jelöljük.

Axiómák:

1.  $P(A) \geq 0$ . (Minden  $A$  eseményre)
2.  $P(\Omega) = 1$ . ( $A$  biztos esemény mindig bekövetkezik)
3.  $P(A + B) = P(A) + P(B)$ . (Ha  $A$  és  $B$  egymást kizáró események)

**Klasszikus kiszámítási módja:**  $P(A) = \frac{k}{n} = \frac{\text{kedvező esetek száma}}{\text{összes esetek száma}}$

**Kombinatorikus kiszámítási módszerek:**

- **Permutáció:** az  $A$  halmaz önmagára vett bijektív leképezése (=sorba rendezés, nem számít a sorrend)
  - **ismétlés nélküli:** **(definíció)**  $n$  különböző elem lehetséges sorbarendezéseinek a száma:  $P_n = n!$
  - **ismétléses:** **(definíció)** Ha  $n$  elemünk van  $k$  különböző fajtából, az 1. fajtából  $l_1$ , a 2.-ből  $l_2$  stb. (azaz  $l_1 + l_2 + \dots + l_k = n$ ), akkor az  $n$  elem lehetséges sorrendjeinek a száma:  $P_n^{l_1, \dots, l_k} = \frac{n!}{l_1! \cdot \dots \cdot l_k!}$

- **Variáció:**  $n$  elemű halmazból kiválasztott  $k$  hosszúságú sorozatok (=kiválasztás és sorba rendezés, számít a sorrend)

- **ismétlés nélküli:** **(definíció)** Egy  $n$  elemű halmaz  $k$ -ad osztályú **ismétlés nélküli variációi** alatt, a halmaz elemeiből *visszatevés nélkül* kiválasztott  $k$  hosszúságú sorozatokat értjük.

Ezek száma:  $V_{n,k} = \frac{n!}{(n-k)!} = n \cdot (n-1) \cdot \dots \cdot (n-k+1)$ , ahol  $n \geq k$

- **ismétléses:** **(definíció)** Egy  $n$  elemű halmaz  $k$ -ad osztályú **ismétléses variációi** alatt, a halmaz elemeiből *visszatevéssel* kiválasztott  $k$  hosszúságú sorozatot értjük.

Ezek száma:  $V_{n,k}^i = n^k$

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

- **ismétlés nélküli:** **(definíció)** Egy  $n$  elemű halmaz  $k$  elemű részhalmazait a halmaz  $k$ -ad osztályú **ismétlés nélküli kombinációinak** nevezzük.

Számuk:  $C_{n,k} = \frac{n!}{k!(n-k)!} =: \binom{n}{k}$ , ahol  $n \geq k$

- **ismétléses:** **(definíció)** Ha egy  $n$  elemű halmaz elemeiből úgy képezzük a  $k$  elemű részhalmazt, hogy egy elemet többször is választhatunk (azaz *visszatevéssel*), akkor az  $n$  halmaz  $k$ -ad osztályú **ismétléses kombinációjáról** beszélünk.

Számuk:  $C_{n,k}^i = \binom{n+k-1}{k} =: \left( \binom{n}{k} \right)$

---

**Feltételes valószínűség:** **(definíció)** Az  $A$  eseménynek a  $B$  eseményre vonatkozó feltételes valószínűsége megadja az  $A$  esemény bekövetkezésének a valószínűségét, feltéve hogy a  $B$  esemény már bekövetkezett vagy bekövetkezik.

$$P(A|B) = \frac{P(AB)}{P(B)}$$

**Függetlenség:** **(definíció)** Két esemény független, ha  $P(AB) = P(A) \cdot P(B)$ . Vagyis, ha az egyik esemény bekövetkezése nem függ a másik bekövetkezésétől.

**Bayes-formula:** **(definíció)** 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.

$$\textbf{Ezen formula: } P(B|A) = \frac{P(A|B) \cdot P(B)}{P(A)}$$

**Tétel:** Legyen  $A$  egy esemény,  $B_1, B_2, \dots$  teljes eseményrendszer,  $P(A) > 0, P(B_i) > 0, i = 1, 2, \dots$  Ekkor

$$P(B_i|A) = \frac{P(A|B_i) \cdot P(B_i)}{\sum_{j=1}^{\infty} P(A|B_j) \cdot P(B_j)}, \text{ minden } j\text{-re.}$$

**2.2. Algoritmusok lépésszáma: beszűrásos rendezés, összefésülésses rendezés, keresések lineáris és logaritmikus lépésszámmal. Gyorsrendezés, az összehasonlítások minimális száma. Rendezés lineáris lépésszámmal: radix rendezés, vödör rendezés.**

**Beszűrásos rendezés:** Mechanizmusa hasonló ahhoz, amikor a kezünkben lévő kártyákat rendezzük. Felveszünk őket egyenként, és beszűrjük a megfelelő helyre. A beszűrő rendezés is hasonlóképpen működik, melynél a kezdő elem az első tömbbeli elem lesz.

Lépésszám, időbonyolultság:

- **legrosszabb eset:**  $O(n^2)$  (ha pont fordítva van rendezve a kiindulási tömb)
- **legjobb eset:**  $O(n)$  (ha a kiinduló tömb eleve rendezve van)

**Összefésülő rendezés:** 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 párossával 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:**

```

5 2 4 6 1
[5 2 4] [6 1]
[5 2] [4] [6] [1]
[5] [2] [4] [6] [1]
[2 5] [4] [1 6]
[2 4 5] [1 6]
[1 2 4 5 6]
```

- "oszd meg és uralkodj" elven működik
- **Lépésszám, időbonyolultság:**  $O(n \log n)$ , ahol  $\log n = \frac{\text{felosztások}}{\text{szintek}}$  száma (?).

- Bizonyos esetekben gyorsabb is lehet mint a *gyorsrendezés*, viszont hátránya a nagy tárterület a felosztások miatt (**nem helyben rendez**).

### Gyorsrendezés:

- "*oszd meg és uralkodj*" elven működik
- **Lépései:**
  1. Kiválasztunk a tömbből egy tetszőleges elemet. Ez lesz az úgynevezett vezérellem (**pivot**). Ez lehet például a tömb *utolsó eleme* is.
  2. Az ennél kisebbeket a tömb elejére, az ennél nagyobbakat a tömb végére rendezzük. A vezérelemmel megegyező elemek mehetnek bármelyik oldalra.
  3. Ezután az így keletkező két tömbrészt külön rendezzük, az algoritmus rekurzív hívásával.
- Vegyük észre, ha a **pivot** előtt csakis kisebb, utána pedig csakis nagyobb elemek vannak, azt jelenti, hogy a **pivot** már a **végleges helyén van**.
- **Lépésszám, időbonyolultság:**
  - **legrosszabb eset:**  $O(n^2)$  (ha a **pivot** *mindig* a tömb legnagyobb eleme)
  - **legjobb eset:**  $O(n \log n)$  (ha a felosztás közel, vagy teljesen egyenlő)

**Radix rendezés** (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*).

43 12 97 10

10 12 43 97

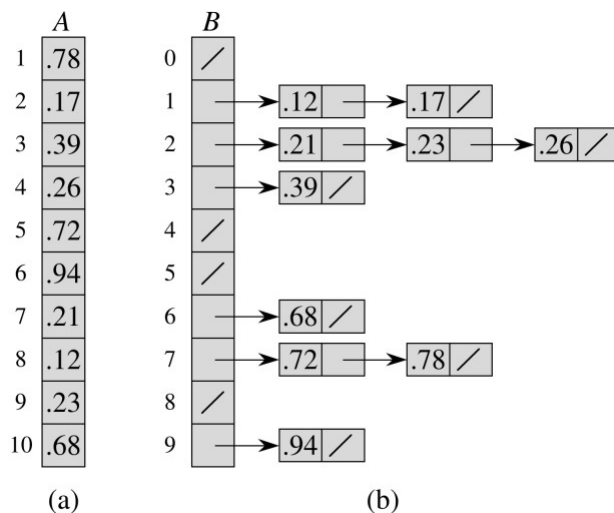
10 12 43 97

- ha nem ugyan olyan hosszúak a számjegyek, akkor át lehet őket alakítani bináris számrendszerbe és úgy *radix rendezni*

- **időbonyolultság:**  $O(d \cdot (n + k))$ , 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 tömb elemeire  $(n_0, n_1, \dots, n_k)$  igaz, hogy:  $0 \leq n_i < 1$  ( $i = 0, 1, \dots, k$ ) 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.



**6. ábra.** Vödör (edény) rendezés láncolt listával

**Időbonyolultság:**

- *lineáris idejű* egészen addig, amíg az edényméretek négyzeteinek összege lineáris a teljes elemszámban
- **legrosszabb esetben:**  $O(n^2)$  (ha vödrök elemszáma nagyban eltér, és a szétválogatást követően a vödrökben lévő elemek fordított sorrendben vannak)
- **legjobb esetben:**  $O(n)$  (egyenletes eloszlású számokkal, és ha a szétválogatást követően már eleve rendezett vödröket kapunk)

### Teljes keresés:

- A tömb elemeinek iterálása az elejétől, egészen a keresett elem megtalálásáig.
- **Időbonyolultság:**  $O(n)$  (ezért is hívják lineáris keresésnek, mert a lépésszám 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, még hozzá rekurzív módon.
- **Időbonyolultság:**  $O(\log n)$  (nagy elemszámú tömbök esetén lényegesen gyorsabb lehet, mint a lineáris)

### 3. Téma

#### 3.1. Függvények szélsőértéke, függvényvizsgálat. A legkisebb négyzetek módszerre

A függvényvizsgálat lépései:

##### 1. Alapvető vizsgálatok:

- Értelmezési tartomány ( $D_f$ ) és szakadási pontok
- Zérushelyek (ha vannak)
- Paritás
- Periodicitás (korlátosság)
- Határértékek ( $+\infty$  és  $-\infty$ -ben, ha van értelme)

##### 2. Derivált függvény vizsgálata ( $f'$ ):

- Monotonitás
- Szélsőértékhelyek

##### 3. Kétszeresen (másodrendű) derivált függvény vizsgálata ( $f''$ ):

- Konvexitás (avagy *konvex* vagy *konkáv*)
- Inflexiós pontok

---

**Szakadási pont:** **(definíció)** Azt mondjuk, hogy a valós számok egy  $D$  részhalmazán értelmezett  $f : D \rightarrow \mathbb{R}$  függvénynek, a  $D$  lezártja egy  $u$  pontjában **szakadása** van, ha

- $u \in D$ , de  $u$ -ban  $f$  nem folytonos vagy
- $u \notin D$

**Zérushely:**  $f(x) = 0$

**Paritás:** 
$$\begin{cases} f(-x) = f(x), \text{ ha } f \text{ páros (a függvény tükrös az } y \text{ tengelyre)} \\ f(-x) = -f(x), \text{ ha } f \text{ páratlan (a függvény tükrös az origóra)} \end{cases}$$



**Periodicitás** (korlátosság):

- **(definíció)** Az  $f : D_f \rightarrow \mathbb{R}$  függvény **alulról korlátos**, ha van olyan  $k \in \mathbb{R}$ , hogy az értelmezési tartomány minden elemére  $k \leq f(x)$ .
- **(definíció)** Az  $f : D_f \rightarrow \mathbb{R}$  függvény **felülről korlátos**, ha van olyan  $K \in \mathbb{R}$ , hogy az értelmezési tartomány minden elemére  $f(x) \leq K$ .
- **(definíció)** Az  $f : D_f \rightarrow \mathbb{R}$  függvény **alulról és felülről is korlátos**, ha van olyan  $k \in \mathbb{R}, K \in \mathbb{R}$ , hogy az értelmezési tartomány minden elemére  $k \leq f(x) \leq K$ .

**Határérték:**

1. **(definíció)** Legyen  $f$  egy olyan függvény, amelyre  $[N, \infty) \subseteq D(f)$  valamely alkalmas  $N \in \mathbb{R}$ . Ekkor az  $f$  függvény **határértéke a végtelenben  $L$**  (jelölés  $\lim_{x \rightarrow +\infty} f(x) = L$ ), ha

$$\forall \varepsilon > 0, \exists M, \forall x [x > M \Rightarrow |f(x) - L| < \varepsilon]$$

2. **(definíció)** Legyen  $f$  egy olyan függvény, amelyre  $(-\infty, N] \subseteq D(f)$  valamely alkalmas  $N \in \mathbb{R}$ . Ekkor az  $f$  függvény **határértéke a mínusz végtelenben  $L$**  (jelölés  $\lim_{x \rightarrow -\infty} f(x) = L$ ), ha

$$\forall \varepsilon > 0, \exists m, \forall x [x < m \Rightarrow |f(x) - L| < \varepsilon]$$

**Függvény deriváltja:** Legyen  $D \subset \mathbb{R}, f : D \rightarrow \mathbb{R}$  és  $x_0 \in D$ . Ekkor a

- **differenciahányados függvény:**  $\phi(x, x_0) = \frac{f(x) - f(x_0)}{x - x_0}$
- **differenciálhányados:**  $\lim_{x \rightarrow x_0} \phi(x, x_0) = \frac{f(x) - f(x_0)}{x - x_0} = f'(x_0)$

**Monotonitás:** Legyen  $x_1, x_2 \in D_f$ .

- **(definíció)** Az  $f$  **monoton nő**, ha bármilyen  $x_1 < x_2$  esetén  $f(x_1) \leq f(x_2)$ .  
**Szigorúan monoton nő**, ha az egyenlőség sehol sem teljesül, vagyis  $f(x_1) < f(x_2)$ .

- **(definíció)** Az  $f$  **monoton csökken**, ha bármilyen  $x_1 < x_2$  esetén  $f(x_1) \geq f(x_2)$ .  
**Szigorúan monoton csökken**, ha az egyenlőség sehol sem teljesül, vagyis  $f(x_1) > f(x_2)$ .

**Szélsőértékhelyek:**

- **(definíció)** Ha  $f$  **felülről korlátos** és van olyan  $x_0 \in D_f$  szám, hogy  $f(x_0) \geq f(x)$  minden  $x \in D_f$  esetén, akkor azt mondjuk, hogy az  $f$  függvénynek **maximuma** van az  $x_0$  pontnál.  
Továbbá,  $f'(x_0) = 0$ .
- **(definíció)** Ha  $f$  **alulról korlátos** és van olyan  $x_0 \in D_f$  szám, hogy  $f(x_0) \leq f(x)$  minden  $x \in D_f$  esetén, akkor azt mondjuk, hogy az  $f$  függvénynek **minimuma** van az  $x_0$  pontnál.  
Továbbá,  $f'(x_0) = 0$ .

**Konvexitás:**

- **(definíció)** Az  $f : D_f \rightarrow \mathbb{R}$  függvény az  $[a, b]$  intervallumon **konvex**, ha a grafikonja feletti tartomány konvex:  $(x, y) \in \mathbb{R}^2 | x \in [a, b] \text{ és } y \geq f(x)$
- **(definíció)** Az  $f : D_f \rightarrow \mathbb{R}$  függvény az  $[a, b]$  intervallumon **konkáv**, ha a grafikonja alatti tartomány konkáv:  $(x, y) \in \mathbb{R}^2 | x \in [a, b] \text{ és } y \leq f(x)$

**Inflexiós pont:** Az  $f : [a, b] \rightarrow \mathbb{R}$  differenciálható függvénynek az  $x_0 \in [a, b]$  pont pontosan akkor inflexiós pontja, ha  $x_0$  **szélsőértékhelye az  $f$  függvénynek** (illetve, ahol a függvény **konvexitást vált**  $f''(x_0) = 0$  – ahol a másodrendű derivált **előjelet vált**).

---

**Legkisebb négyzetek módszere:** (pl. *lineáris regresszió*)

- a megfigyelési pontokra  $(t_1, t_2, \dots, t_n)$  időpillanatok és  $f_1, f_2, \dots, f_n$  megfigyelések) keressük a legjobban illeszkedő modellt (pl. *polinomot*)
- célunk a **hibák négyzetösszegének**  $\left(\sum_{i=1}^n (f(x_i) - y_i)^2\right)$  **minimalizálása**

- minimum csak abban a pontban lehet, ahol a **parciális deriváltja**  $x_k$  szerint  $\frac{\sigma J(x)}{\sigma x_k} = 0$ . Ez a következő lineáris egyenletrendszerhez vezet:  $A^T A x = A^T f$   
(*Gauss-féle normálegyenlet*)

**3.2.** Az elsőrendű logika nyelvének szintaxisa. Változók kötött és szabad előfordulása. A nyelv interpretációja, változókiértékelés. Termek és formulák értéke interpretációban, változókiértékelés mellett. Törvény, ellentmondás, ekvivalencia, következmény. Normálformák, prenex formulák. Logikai kalkulások.

**Klasszikus elsőrendű nyelv:**(**definíció**)Klasszikus elsőrendű nyelven az  $L^{(1)} = \langle LC, Var, Con, Term, Form \rangle$  rendezett ötöst értjük, ahol

- $LC = \{\neg, \wedge, \vee, \supset, \equiv, \forall, \exists, =, (, )\}$  (a **nyelv logikai konstansainak** halmaza – erőségi/kötési sorrend szerint **csökkenően**)
- $Var = \{x_n | n = 0, 1, 2, \dots\}$  (a **nyelv változóinak** megszámlálhatóan végtelen halmaza)
- $Con = \bigcup_{n=0}^{\infty} (F(n) \cup P(n))$  a **nyelv nemlogikai konstansainak** legfeljebb megszámlálhatóan végtelen halmaza.
  - $F$  - névkonstansok,  $P$  - állításkonstansok
- $Term$  (a nyelv **terminusainak** halmaza)
  - $F(0) \cup Var \subseteq Term$
- $Form$  (a nyelv **formuláinak** halmaza)
  - $P(0) \subseteq Form$
- Az  $LC, Var, F(n), P(n)$  halmazok páronként diszjunktak

**Például:**  $\exists x \forall y \text{ szeret}(x, y)$  (Létezik valaki, aki mindenkit szeret.)

---

**Változók kötött és szabad előfordulása:**

1. **Szabad változók:** ha nem köti kvantor a változókat (a formula *paramétere*)
  - Nem szabad átnevezni őket.
  - Egy **atomi formulában** az összes változó szabad előfordulású.

- Ha az  $A$  formula  $\neg B$  alakú, akkor  $FreeVar(A) = FreeVar(B)$ .
- Ha az  $A$  formula  $(B \supset C), (B \wedge C), (B \vee C)$  vagy  $(B \equiv C)$  alakú, akkor  $FreeVar(A) = FreeVar(B) \cup FreeVar(C)$ .
- Ha viszont már kvantorokat  $(\forall, \exists)$  használunk, akkor már **nincs** szabad előfordulású változó.

## 2. Kötött változók:

- Szabadon át lehet nevezni őket (de akkor **egységesen**).
- Egy **atomi formulában nincs kötött előfordulású változó**.
- Ha az  $A$  formula  $\neg B$  alakú, akkor  $BoundVar(A) = BoundVar(B)$ .
- Ha az  $A$  formula  $(B \supset C), (B \wedge C), (B \vee C)$  vagy  $(B \equiv C)$  alakú, akkor  $BoundVar(A) = BoundVar(B) \cup BoundVar(C)$ .
- Ha viszont már kvantorokat  $(\forall, \exists)$  használunk, akkor már **csakis** kötött előfordulású változókról beszélünk.

---

**Interpretáció:** **(definíció)** Az  $\langle U, \varrho \rangle$  párt az  $L^{(1)} = \langle LC, Var, Con, Term, Form \rangle$  **elsőrendű nyelv** egy interpretációjának nevezzük, ha

- $U \neq \emptyset$ , azaz  $U$  nemüres halmaz
- $Dom(\varrho) = Con$ , azaz a  $\varrho$  a  $Con$  halmazon értelmezett függvény, amelyre teljesülnek a következők:
  - Ha  $a \in F(0)$ , akkor  $\varrho(a) \in U$
  - Ha  $f \in F(n), n \neq 0$ , akkor  $\varrho(f) : U^{(n)} \rightarrow U$
  - Ha  $p \in P(0)$ , akkor  $\varrho(p) \in \{0, 1\}$
  - Ha  $P \in P(n), n \neq 0$ , akkor  $\varrho(P) \subseteq U^{(n)}$

**Értékelés:** **(definíció)** Legyen a fenti elsőrendű nyelv az interpretációjával. Ekkor az  $\langle U, \varrho \rangle$  interpretációra támaszkodó  $v$  értékelésen egy olyan **függvényt** értünk, amely a következőképpen néz ki:

$$v : Var \rightarrow U$$

**Termek és formulák értékelése interpretációban:** **(definíció)** Legyen  $I$ , egy  $L^{(1)}$  nyelvbeli interpretáció. Jelölje továbbá a  $|\dots|^{I,k}$  egy term vagy formula kiértékelését az  $I$  interpretációban. Ekkor:

1.  $||^{I,k} \Rightarrow \{i, \text{ ha } P^I(|t_1|^{I,k}, \dots, |t_k|^{I,k}) = i, \text{ egyébként } h\}$  ( $i$  - igaz,  $h$  - hamis)
2.  $|\neg A|^{I,k} \Rightarrow \neg |A|^{I,k}$
3.  $|A \wedge B|^{I,k} \Rightarrow |A|^{I,k} \wedge |B|^{I,k}$
4.  $|A \vee B|^{I,k} \Rightarrow |A|^{I,k} \vee |B|^{I,k}$
5.  $|A \supset B|^{I,k} \Rightarrow |A|^{I,k} \supset |B|^{I,k}$
6.  $|\forall x A|^{I,k} \Rightarrow \{i, \text{ ha } |A|^{I,k^*} = ik \text{ minden } k^*x\text{-variánsra}\}$  (?)
7.  $|\exists x A|^{I,k} \Rightarrow \{i, \text{ ha } |A|^{I,k^*} = ik \text{ valamely } k^*x\text{-variánsra}\}$  (?)

**Megjegyzés:** egy formula kiértékelése nagyban függ a predikátumok jelentésétől (hogyan pontosan milyen exakt relációt jelentenek)

---

**Törvény, ellentmondás, ekvivalencia, következmény:**

Egy elsőrendű logikai nyelv egy  $A$  formulája:

- **kielégíthető:** ha van a nyelvnek olyan  $I$  interpretációja és  $I$ -ben van olyan  $k$  változókiértékelés, amelyre  $|A|^{I,k} = i$  (igaz)
- **kielégíthetetlen** (ellentmondás): ha a nyelv minden  $I$  interpretációjában és  $I$  minden  $k$  változókiértékelése mellett  $|A|^{I,k} = h$  (hamis)
- **törvény:** ha a nyelv minden  $I$  interpretációjában és  $I$  minden  $k$  változókiértékelése mellett  $|A|^{I,k} = i$  (igaz)

**Ekvivalencia:** **(definíció)** Az  $A$  és  $B$  elsőrendű formulák **logikailag ekvivalensek**, ha minden  $I$  interpretációban és  $k$  változókiértékelés mellett

$$|A|^{I,k} = |B|^{I,k}$$

**Megjegyzés:** formulahalmazoknál is hasonlóképpen igaz és fennáll.

**Következmény:** **(definíció)** A  $B$  elsőrendű formula **következménye**  $\Gamma$ -nak, ha a nyelv minden olyan interpretációja és változókiértékelése amely kielégíti  $\Gamma$ -t, az kielégíti  $B$ -t is.

**Tautologikus következmény:** **(definíció)** Azt mondjuk, hogy a  $B$  **tautologikus következménye**  $\Gamma$ -nak, ha  $\Gamma$  és  $B$  közös igazságtáblázatában azon sorokban, ahol  $\Gamma$  kielégíthető,  $B$  oszlopában is csupa  $i$  (igaz) igazságérték van.

---

**Prenex normálforma:** **(definíció)** Legyen  $L^{(1)} = \langle LC, Var, Con, Term, Form \rangle$  egy tetszőleges **elsőrendű nyelv**.

Az  $A \in Form$  formulát **prenex alakúnak** nevezzük, ha az alábbi két feltétel valamelyike teljesül:

1. Az  $A$  formula **kvantormentes**, azaz sem a  $\forall$  sem a  $\exists$  kvantor nem szerepel benne, vagy
2. az  $A$  formula  $Q_1x_1Q_2x_2\dots Q_nx_nB$ ,  $(n = 1, 2, \dots)$  alakú, ahol
  - $B \in Form$  kvantormentes formula
  - $x_1, x_2, \dots, x_n \in Var$  különböző változók
  - $Q_1, Q_2, \dots, Q_n \in \{\forall, \exists\}$  kvantorok

**Megjegyzés:** minden formulát, az eredetivel ekvivalens prenex alakúra lehet hozni.

**Prenex normálformára-hozás lépései:**

1. Ha van materiális ekvivalencia a formulában, azt fel kell oldani.
  - $(A \equiv B) \Leftrightarrow ((A \supset B) \wedge (B \supset A))$
2. Változóiban tiszta alakra hozás.
  - kötött változók szabályos átnevezésével
3. *De Morgan* törvényeinek alkalmazása.
  - $\neg \forall x A \Leftrightarrow \exists x \neg A$
  - $\neg \exists x A \Leftrightarrow \forall x \neg A$

#### 4. Kvantorkiemelés.

- ha implikáció van, és kvantor a bal oldalon, akkor a kiemelésnél fordítjuk a kvantort, egyébként pedig csak egyszerűen kiemeljük a kvantorokat

#### Normálformák nulladrendű nyelvek esetén:

- **diszjunktív normálforma:** formailag  $(\neg a \wedge b \wedge c) \vee (a \wedge b \wedge c)$
- **konjunktív normálforma:** formailag  $(\neg a \vee b \vee c) \wedge (a \vee b \vee c)$
- **normálformára-hozás lépései:**
  1. A **materiális ekvivalenciát** ( $\equiv$ ) átalakítjuk vele ekvivalens **diszjunkcióra** (?).
    - implikáció esetében pedig alkalmazzuk a következő összefüggést:  $A \supset B \Leftrightarrow \neg A \vee B$
  2. A **kettős tagadás** és **De Morgan** törvényeivel elérjük, hogy a negáció csak atomi formulákra vonatkozzon.
    - $\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$  és  $\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B$
  3. A **disztributivitást** felhasználva addig alakítjuk a formulát, amég a **konjunktciók** és **diszjunkciók** megfelelő sorrendben nem lesznek.
  4. Egyszerűsítünk.

---

#### Logikai kalkulusok (*Gentzen kalkulus*)

**Szekvent:** **(definíció)** Legyen  $A_1, A_2, \dots, A_n$  és  $B_1, B_2, \dots, B_m$  tetszőleges formulák. A

$$T \wedge A_1 \wedge A_2 \wedge \dots \wedge A_n \supset B_1 \vee B_2 \vee \dots \vee B_m \vee \perp$$

alakú formulát **szekventek** nevezzük. **Jelölése:**  $A_1, A_2, \dots, A_n \vdash B_1, B_2, \dots, B_m$

**Levezethetőség:** **(definíció)** Egy szekvent **levezethető** a szekventkalkulusban, ha vagy

- $A, \Gamma \vdash \Delta, A$  alakú vagy



- van olyan levezetési szabály, amelyben a **szekvent** a **vonallalatti szekventsémával azonos alakú**, és elkészítve a szekventünkből, ezen levezetési szabály vonal feletti szekventsémájával azonos alakú szekventet(ek), és ez(ek) levezethetőek a szekventkalkulusban. (???)

**Helyesség:** **(definíció)** A szekventkalkulus **helyes**, mert ha  $A_1, A_2, \dots, A_n$  és  $B_1, B_2, \dots, B_m$  szekvent levezethető a szekventkalkulusban, akkor a  $T \wedge A_1 \wedge A_2 \wedge \dots \wedge A_n \supset B_1 \vee \dots \vee B_m \vee \perp$  formula **logikai törvény**.

**Teljesség:** **(definíció)** A szekventkalkulus **teljes**, mert ha  $B$  formula **logikai törvény**, akkor a " $\vdash B$ " szekvent levezethető a szekventkalkulusban.

**Szekvent kalkulus levezetési szabályai:**

$$\begin{array}{c}
 A, \Gamma \rightarrow \Delta, A \\
 \\
 \begin{array}{ll}
 (\rightarrow \supset) & \frac{A, \Gamma \rightarrow \Delta, B}{\Gamma \rightarrow \Delta, (A \supset B)} \quad (\supset \rightarrow) \quad \frac{\Gamma \rightarrow \Delta, A \quad B, \Gamma \rightarrow \Delta}{(A \supset B), \Gamma \rightarrow \Delta} \\
 (\rightarrow \wedge) & \frac{\Gamma \rightarrow \Delta, A \quad \Gamma \rightarrow \Delta, B}{\Gamma \rightarrow \Delta, (A \wedge B)} \quad (\wedge \rightarrow) \quad \frac{A, B, \Gamma \rightarrow \Delta}{(A \wedge B), \Gamma \rightarrow \Delta} \\
 (\rightarrow \vee) & \frac{\Gamma \rightarrow \Delta, A, B}{\Gamma \rightarrow \Delta, (A \vee B)} \quad (\vee \rightarrow) \quad \frac{A, \Gamma \rightarrow \Delta \quad B, \Gamma \rightarrow \Delta}{(A \vee B), \Gamma \rightarrow \Delta} \\
 (\rightarrow \neg) & \frac{A, \Gamma \rightarrow \Delta}{\Gamma \rightarrow \Delta, \neg A} \quad (\neg \rightarrow) \quad \frac{\Gamma \rightarrow \Delta, A}{\neg A, \Gamma \rightarrow \Delta}
 \end{array}
 \end{array}$$

**7. ábra.** Szekvent kalkulus levezetési szabályai

**Megjegyzések:**

- negációnál csak simán átdobjuk a másik oldalra a megfelelő tagokat
- az  $(\wedge r)$ ,  $(\vee l)$  és  $(\supset l)$  ketté bontják a relációt
- az összes többi pedig helyben tartja, csak szétszedi bal és jobb oldalra a relációt és tagjait

## 4. Tétel

### 4.1. Függvények, görbék, felületek leírása és számítógépes ábrázolása.

#### Függvények

- függvények segítségével jeleníthető meg egy ábra
- csak közelíteni tudjuk a számítógépes ábrázolásnál (gyakorlatilag tehát **szakaszokból áll**)
  - a szakaszok sűrűsége a pontok mennyiségétől függ

#### Függvények típusai:

1. **Explicit függvények:** pl.  $f : \mathbb{R} \rightarrow \mathbb{R}$ ,  $f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_0 x^0$   
(polinomiális függvény)
    - **egyértelmű hozzárendelés**
    - $a_n, a_{n-1}, \dots, a_0$  - a polinom együtthatói,  $n$  - fokszám,  $x$  - változó
    - minden polinomiális függvény egyértelmű meghatározásához/ábrázolásához legalább  $n + 1$  pontra van szükség
  2. **Implicit függvények:**  $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ 
    - ha **nincs egyértelmű hozzárendelés**
    - **például:** kör, ellipszis, stb.
    - $(x, y)$ -hoz rendelünk egy  $F(x, y)$  függvényt (?)
    - minden eddig függvényt át lehet alakítani implicit alakúra
  3. **Vektorfüggvények:** (vektor értékű, paraméteres alakban megadott)
    - az idő függvényében egy **vektort** írunk le:  $v(t) : [a, b] \rightarrow V^2$
    - a mozgó helyvektor végpontjai rajzolják a görbét (?)
-

## Görbék típusai:

### 1. Explicit görbék: pl. polinommal megadott görbe

- ahányad fokú polinom  $\Leftrightarrow$  akkorad rendű görbe

### 2. Implicit görbék:

- lassú kiértékelésű, és **nem egyértelmű**, ezért ritkán használatos a számítógépes grafikában
- az értékek behelyettesítése a **távolságot** is mutatja (minél nagyobb az érték, annál távolabb van a pont)

### 3. Vektorgörbék (paraméteres): mozgó helyvektor végpontjai rajzolják a görbét, tehát **idő függvényében** vektort írunk le.

### 4. Térbeli görbék: rajzolására alkalmatlanok az implicit és explicit függvények. Csak vektorfüggvénnyel rajzolhatók.

Egy  $m$ -ed és egy  $n$ -ed rendű görbének legfeljebb  $m \times n$  darab **látható** metszéspontja lehet. (*Bezont-tétel*)

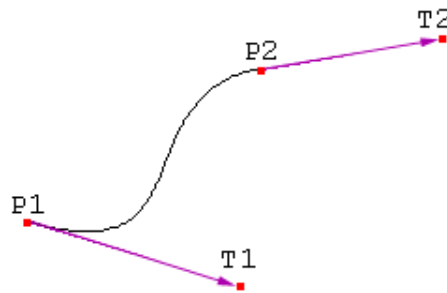
## Nevezetes görbék:

### 1. Hermit ív:

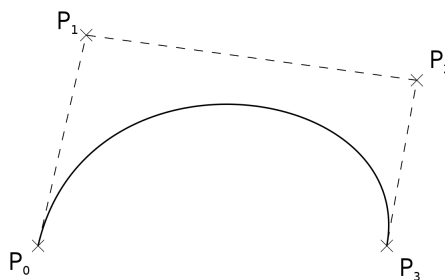
- a következők adottak: **2 pont és 2 érintő**, vagyis kezdő ( $P_1$ ) és végpont ( $P_2$ ), illetve ezekben a pontokban az érintők ( $T_1, T_2$ )

### 2. Bézier görbe: $c(t) = \sum_{i=0}^n P_i \cdot B_i^n(t)$ (*Bernstein polinom*)

- általában **4 (kontroll)pontra** ( $P_0, P_1, P_2, P_3$ ) **illesztendő**
- de akármennyi pontra illeszthető
- *Hermit ív* továbbfejlesztése



8. ábra. Hermite ív



9. ábra. Bézier görbe

---

### Felületek típusai:

#### 1. Explicit felületek: $f : \mathbb{R}^2 \rightarrow \mathbb{R}, f(x, y) = z$

- **dróthálós megjelenés:** vagyis a felületi görbék ábrázolása
- **előny:** könnyű ábrázolás
- **hátrány:** maga alá görbülő felületet nem tud ábrázolni

#### 2. Implicit felületek: $F(x, y, z) = 0$

- minden  $(x, y)$  értékhez több érték is társulhat, tehát **maga alá görbülhet a felület** (pontosan ez az egyik előnye is)
- **előny:** maga alá görbülő felületet tud ábrázolni
- **hátrány:** pontonkénti kiértékelés
- **metszéspontok:** felület + görbe, felület + felület

- **(definíció)** Egy  $n$ -edrendű felületnek és egy  $m$ -edrendű görbének  $n \times m$  lát-ható (=valós koordinátákkal rendelkező) metszéspontja lehet.

Tehát egy felület egyenletének a foksámát eldönthetem úgy, hogy egye-nessel metszem, és a metszéspontok száma megadja a felület egyenletének a foksámát.

- **(definíció)** Egy  $n$ -edrendű és egy  $m$ -edrendű felület **metszésvonala** egy  $n \times m$ -edrendű **görbe**

3. **Paraméteres felületek:** míg a görbék ábrázolása során pontokat kötünk össze, itt **paraméter vonalakkal** rácsozzuk be.

Legyen  $I^2 = [a, b] \times [c, d]$  a két időintervallum *Descartes szorzata*.

Ekkor a  $\bar{v}(u, v)$  három koordináta függvénnyel számolódik, melyek:

- $x(u, v) : I^2 \rightarrow \mathbb{R}$
- $y(u, v) : I^2 \rightarrow \mathbb{R}$
- $z(u, v) : I^2 \rightarrow \mathbb{R}$

## Összefoglalás

	Explicit	Implicit	Paraméteres
Görbék	$f(x) = y$	$F(x, y) = 0$	$\bar{v}(t)$
Felületek	$f(x, y) = z$	$F(x, y, z) = 0$	$\bar{v}(u, v)$ (?)

**4.2. Problémák reprezentálása állapottéren. A megoldás keresése visszalépéssel. Szisztematikus és heurisztikus gráfkereső eljárások: a szélességi, a mélységi és az A algoritmusok. Kétszemélyes játékok és reprezentálásuk. A nyerő stratégia. Lépésajánló algoritmusok.**

**Az állapottér fogalma:** A mesterséges intelligenciában a problémák megoldása a probléma megfogalmazásával kezdődnek. A problémát leírjuk/reprezentáljuk. Egyik legelterjedtebb reprezentációs technika az **állapottér-reprezentáció**. (*state-space representation*)

$$P = \langle A, K_{ezd}, C_{él}, O_{perátorok} \rangle \text{ (javítani)}$$

**Megjegyzés:** egy probléma megoldható, ha van olyan célállapot, ami elérhető a kezdőállapottól.

**Állapottér gráf:** az állapottér (irányított) gráfot alkot, aminek csomópontjai az **állapotok**, a csomópontok közötti élek pedig a **cselekvések**. Jellemzői:

- **Elágazási tényező:** tetszőleges állapotból közvetlenül elérhető állapotok maximális száma, jele:  $b$
- **A legsekélyebb megoldás:** a legkevesebb operátoralkalmazás segítségével elérhető célállapot eléréséhez szükséges operátoralkalmazások száma, jele:  $d$
- **A csomópontok maximális mélysége:** jele:  $m$

---

**Visszalépéses kereső (backtracking):**

- ha elakadunk, egyszerűen **vissza lehet lépni**
- **nem informált** kereső (csak a probléma definícióját ismeri)
- **időbonyolultság:**  $O(b^m)$
- **tárkonyolultság:**  $O(m)$
- **teljesség:** csak véges, körmentes gráfban teljes
- **optimalitás:** nem garantál optimális megoldást

**Megjegyzés:** A visszalépéses kereső alkalmazásával a tárkonyoltság csökkenthető tovább.

---

### Gráfkereső algoritmusok

**Adatbázisa:** a keresés során tárolt elemek. (?)

Egyetlen **művelete a kifejtés:**

1. Távolítsuk el a csomópontot a peremből.
2. Ha a csomópont által reprezentált állapot még nem eleme a zárt listának,
  - Adjuk hozzá a zárt listához,
  - Állítsuk elő a csomópont követőit, a csomópont által reprezentált állapotból közvetlenül elérhető állapotokhoz, új csomópontot(kat) készítve a perembe.

**Kifejtések:** a fa-kifejtésnél lehetnek duplikátumok, a gráf-kifejtésnél nem.

**Szélességi kereső:** egy VIS tömböt és egy P peremet használunk a bejárásra. Sorba megyünk a gráf csomópontjain. Ha meglátogattunk egyet, akkor elhelyezzük a VIS tömbben, a közvetlen rákövetkezőit pedig a P peremben helyezzük el, mégpedig hátulra.

A következő meglátogandó csomópont, a **perem első eleme** lesz.

- **nem informált** kereső (csak a probléma definícióját ismeri)
- **időbonyolultság:**  $O(b^{d+1})$
- **tárkonyolultság:**  $O(b^{d+1})$
- **teljesség:** teljes, ha  $b$  véges
- **optimalitás:** *optimális*, ha minden élköltség egységnyi

**Mélységi kereső:** egy VIS tömböt és egy P peremet használunk a bejárásra. Sorba megyünk a gráf csomópontjain. Ha meglátogattunk egyet, akkor elhelyezzük a VIS tömbben, a közvetlen rákövetkezőit pedig a P peremben helyezzük el, mégpedig hátulra.

A következő meglátogandó csomópont, a **perem utolsó eleme** lesz.

- **nem informált** kereső (csak a probléma definícióját ismeri)

- időbonyolultság:  $O(b^m)$
- tárbonyolultság:  $O(b \cdot m)$
- teljesség: csak véges, körmentes gráfban teljes
- optimalitás: *nem garantál* optimális megoldást

Egyéb keresők:

- **egyenletes költségű kereső**: a legkisebb össz úti-költségű csomópontot válassza.
- **legjobbát először kereső**: a heurisztika is jelen van már. A legkisebb heurisztikájú csomópontot válassza.

**A\* keresés**: A legjobbát-először keresés legismertebb fajtája az A\* keresés. Mindíg a legkisebb, **teljes úti költség + heurisztikát** válassza.

**Cél**: a teljes becsült útiköltség minimalizálása:

$$f(n) = g(n) + h(n), \text{ ahol}$$

- $f(n)$  – a legolcsóbb, az  $n$  csomóponton keresztül vezető megoldás becsült költsége
- $g(n)$  – az aktuális csomópontig megtett út költsége
- $h(n)$  – az adott csomóponttól a célhoz vezető út költségének becslőjét (az adott csomópont *heurisztikája*)

**Optimális költség**: a cél csomópontban az összesített útiköltséget **optimális költségnek** nevezzük.

---

**Kétszemélyes játékok és reprezentálásuk**:

- **formálisan**:  $\langle B, b_0, J, \hat{v}, L \rangle$ , ahol
  - $B$ : a játékállások halmaza
  - $b_0 \in B$ : a kezdőállás
  - $J$ : a játékosok halmaza, ahol  $|J| = 2$



- $\hat{v}$ : egy  $V \rightarrow \{-1, 0, 1\}$ -be képző függvény (ez határozza meg, hogy melyik játékos nyert):
  - \* 1 – a következő játékos nyer
  - \* -1 – a következő játékos veszít
  - \* 0 – egyébként (döntetlen)
- $L$ : a lépések halmaza

**Játékfa:** a játékrepresentációból készíthető

---

**Nyerő stratégia:**

- **játszma:** **(definíció)** Olyan lépéssorozat, ami a játék kezdőállapotából, valamilyen olyan állapotba vezető összefüggő lépéssorozat, ami esetén végálláshoz érkezünk.
- **stratégia:** **(definíció)** Olyan leképezés, ami egy adott játékos számára, minden olyan helyzetben, ami esetében lépni következik, előírja, hogy melyik lépést tegye meg.

**Nyerő stratégia:** ha minden stratégia mellett lejátszható játszmában ő nyer.

- Az egyik játékos garantáltan rendelkezik nyerő stratégiával, amennyiben döntetlen nem állhat elő, illetve, ha döntetlen előállhat, akkor az egyik játékosnak garantáltan van nem veszteső stratégiája.

---

**Lépésajánló algoritmusok:**

1. **MinMax algoritmus:** a fa felépítésénél a levélelemektől haladunk és ha a gyermekelemek szülője a MAX, akkor a nagyobb gyermekelemet válasszuk ki, ha pedig a MIN a szülője, akkor a kisebbet. Ezen lépéseket folytatjuk amíg a gyökérelemhez nem értünk.

- **hasznosságfüggvény:**  $h : A \rightarrow [-m, m]$ , ahol  $m \in \mathbb{R}^+$  és

$$h_i(a, j) = \begin{cases} h(a, j), & \text{ha } i = j \\ -h(a, j), & \text{egyébként} \end{cases}$$

- **időbonyolultság:**  $O(b^m)$

- tárbonyolultság:  $O(m)$

2. **Alfa-Béta nyesés:** ( $\alpha = -\infty, \beta = +\infty$ ) párokat visszük minden elemre/ágra, és ha az aktuális  $\alpha$ -nál nagyobbbat találtunk (és MAX a szülő) akkor  $\alpha$ -t cseréljük, ha pedig az aktuális  $\beta$ -nál kisebbet (és MIN a szülő), akkor pedig  $\beta$ -t cseréljük. Ha az  $\alpha > \beta$  feltétel teljesül, a következő ágat(kat) **lenyessük**.

- a *MinMax* algoritmus továbbfejlesztett változata, ezért ezt szokták a gyakorlatban használni
- teljesen úgy működik mint a *MinMax* algoritmus annyi különbséggel, hogy ha ezt egy standard minimax fára alkalmazzuk, ugyanazt az eredményt adja vissza, mint a minimax, a **döntésre hatással nem lévő ágakat** azonban **lenyesi** (sőt, akár teljes *részfákat* is lenyeshet)
- tetszőleges mélységű fákra lehet alkalmazni

3. **Negamax algoritmus:** *kifejtendő*

## 5. Tétel

### 5.1. Mátrix fogalma, műveletek, determináns, rang. Speciális mátrixok, inverz.

Mátrix, mint lineáris transzformáció. Sajátérték, sajátvektor.

**Mátrix:** **(definíció)** Egy  $m$  sorral és  $n$  oszloppal rendelkező **számtáblázatot**  $m \times n$ -es **mátrixnak** nevezünk.

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

Az összes  $m \times n$ -es mátrix halmazát  $\mathcal{M}_{m \times n}$ -nel jelöljük.

**Jellemzők:**

- Ha  $n = m$ , akkor a mátrix **négyzetes** vagy **kvadratikus**
- A mátrix **főátlója** alatt az  $(a_{11}, a_{22}, \dots)$  szám  $k$ -ast értjük.
- Két mátrix **egyenlő**, ha azonos típusúak (azaz ugyanannyi soruk és oszlopuk van), és a megfelelő elemeik megegyeznek.
- Azon  $n \times n$ -es mátrixot, melynek a főátlójában csupa 1-es áll, minden más eleme pedig 0,  **$n$ -edrendű** vagy  **$n$ -dimenziós egység mátrixnak** nevezzük.

**Jele:**  $E_n$  vagy  $I_n$  az egység mátrix esetében, illetve  $O_n$  a null-mátrix esetében (melynek pedig minden eleme 0).

**Műveletek:**

1. **Összeadás:** Csak **azonos típusú** mátrixokat tudunk összeadni.

Legyenek  $A = (a_{ij}), B = (b_{ij}), C = (c_{ij})$   $m \times n$ -es mátrixok.

Ekkor  $C = A + B$ , ha  $c_{ij} = a_{ij} + b_{ij}$ , ahol  $i = 1, \dots, m$  és  $j = 1, \dots, n$ .

2. **Skalárral való szorzás:** Elemenként végezzük, azaz ha  $\lambda \in \mathbb{R}, A = (a_{ij}) \in \mathcal{M}_{m \times n}$ , akkor  $\lambda A = (\lambda a_{ij}) \in \mathcal{M}_{m \times n}$ .

3. **Mátrixszorzás:** Legyenek  $A = (a_{ij})m \times k$  és  $B = (b_{ij})k \times n$  típusú mátrixok. Ekkor  $A$  és  $B$  szorzata az a  $C = (c_{ij})m \times n$  típusú mátrix, amelyre

$$\sum_{r=1}^k a_{ir}b_{rj}$$

**Mátrix inverze:** **(definió)** Azt mondjuk az  $A$   $n$ -edrendű négyzetes mátrixról, hogy **invertálható**, vagy **létezik az inverze**, ha létezik olyan  $B$   $n$ -edrendű kvadrátikus mátrix, hogy  $AB = BA = E_n$ .

**Tétel:** Ha  $A$  invertálható, akkor az inverze *egyértelmű*. **Jele:**  $A^{-1}$ .

**Mátrix rangja:** **(definió)** Egy **mátrix rangja** alatt a mátrix sorai (vagy oszlopai), mint vektorok által alkotott vektorrendszer rangját értjük. **Jelölés:**  $\text{rang}(A)$ .

**Megjegyzés:** ha a mátrix rangja  $k$ , akkor az oszlopaiból vagy soraiból kiválasztható  $k$  darab **lineárisan független** vektor.

**Mátrix determinánsa:** Legyen  $A = (a_{ij})$  egy  $n \times n$ -es kvadrátikus mátrix. Az  $A$  mátrix  $n^2$  eleméből válasszunk ki úgy  $n$  elemet, hogy minden sorból és oszlopból pontosan egyet válasszunk. A kiválasztott elemek alakja:

$$a_{1\sigma(1)}, a_{2\sigma(2)}, \dots, a_{n\sigma(n)}$$

Ekkor az  $A$  mátrix determinánsa:

$$\det(A) = |A| = \sum_{\sigma} \varepsilon(\sigma) a_{1\sigma(1)} \cdot a_{2\sigma(2)} \cdot \dots \cdot a_{n\sigma(n)}, \text{ ahol}$$

$$\varepsilon(\sigma) = \begin{cases} 1, & \text{ha } \sigma \text{ páros} \\ -1, & \text{ha } \sigma \text{ páratlan} \end{cases}$$

Ez az összeg  $n!$  tagú.

**Speciális mátrixok:** alsó- és felsőháromszög mátrix, ritka mátrix, egység-mátrix, null-mátrix, szimmetrikus mátrix.

**Lineáris leképezés:** **(definió)** Legyenek  $V$  és  $W$  vektorterek  $\mathbb{R}$  felett. Ekkor a  $\varphi : V \rightarrow W$  **lineáris leképezés**, ha

- **additív**, azaz  $\forall u, v \in V : \varphi(u + v) = \varphi(u) + \varphi(v)$ .
- **homogén**, azaz  $\forall v \in V, \lambda \in \mathbb{R} : \varphi(\lambda v) = \lambda \varphi(v)$ .

**(definíció)** Ha  $V = W$ , akkor a lineáris leképezést, **lineáris transzformációnak** hívjuk.

**Lineáris transzformáció sajátértékei, sajátvektorai:** **(definíció)** Legyen  $\varphi : V \rightarrow V$  lineáris transzformáció.

Egy nem-nulla  $v \in V$  vektort  $\varphi$  **sajátvektorának** hívunk, ha  $\exists \lambda \in \mathbb{R} : \varphi(v) = \lambda v$ . Ekkor  $\lambda$ -t a  $\varphi$  lineáris transzformáció  $v$ -hez tartozó **sajátértékének** nevezzük.

**Megjegyzések:**

- **karakterisztikus egyenlet/polinom:**  $\det(A - \lambda E_n) = 0$
- a **sajátvektor** kiszámításánál vissza kell helyettesíteni  $\lambda$ -ként a karakterisztikus polinomba, és onnan kiszámolni a sajátvektort, melynek alakja:

$$\bar{v} = t \cdot \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, t \in \mathbb{R}$$

**5.2. A problémaredukciós reprezentáció és az ÉS/VAGY gráfok. Ismeretrepresentációs technikák, bizonytalanság-kezelés (fuzzy logika). A rezolúciós kalkulus. A logikai program és az SLD rezolúció. A logikai programozás alapvető módszerei.**

Ha egy problémát mindaddig egyszerűsítünk, bontogatunk, amég elemi, egyszerű problémákhoz jutunk, amiket már immáron egyszerűen meg tudunk oldani, azt **problémaredukciónak** nevezzük.

**Problémaredukciós reprezentáció:**

- először is le kell írni az eredeti problémát
- **egyszerű probléma:** amit meg tudunk oldani, vagy ismerjük a megoldást
- meg kell adni a problémát egyszerűsítő **redukciós operátorokat** is
  - egy redukciós operátor egy problémához azokat a részproblémákat rendeli hozzá, melyek megoldásával az eredeti is megoldásra kerül
  - egy **operátorsorozattal** egy problémahalmazt egy másik problémahalmazzá redukálhatunk
- **megoldhatóság:** egy probléma megoldható, ha csupa egyszerű problémából álló problémahalmazzá redukálható
- a **feladatok** lehetnek:
  - megoldható-e a probléma a problémaredukciós reprezentációban
  - egy- vagy az összes megoldás előállítás
- **például:** *Hanoi tornyai*

**ÉS/VAGY gráfok:**

- egy reprezentáció egy irányított gráfot (*ÉS/VAGY gráf*) határoz meg
- a **gráf csúcsai:** a problémahalmaz elemei (maguk a problémák)

- a **gráf élei**: a közvetlenül elérhető problémákhoz húzzuk őket (**ÉS élköteget** vagy **hiperélt** alkotnak).

Továbbá, ezek az élek **összetartozónak** tekinthetők.

- **megoldhatóság**: ha van egy startcsúcsból kiinduló olyan hiperút, melynek levelei **terminális csúcsok**
- **megoldáskereső módszerek**: visszalépéses- vagy keresőfával megoldáskereső

## Ismeretreprezentációs technikák

Fajtái:

- procedurális reprezentáció
- logikai alapú reprezentáció (logikai nyelvek, szabályalapú reprezentációk)
- strukturált reprezentáció (szemantikus hálók és keretek, döntési fák)
- hibrid reprezentációk

**Bizonytalanság kezelés** (fuzzy logika): **(definíció)** A fuzzy logika célja, a nem egyértelmű állítások<sup>5</sup> matematikai kezelhetőségének a megkönnyítése. Azért van minderre szükség, mivel ezen állítások szubjektívek.

A fuzzy logika bevezeti, hogy az állítások a  $\{0, 1\}$  értékeket vehetik fel<sup>6</sup>, amik ezek között vannak, azok pedig bizonytalanságot reprezentálnak.

Továbbá, mindez lehetővé teszi komplexebb problémák esetén az **alacsony számítási bonyolultságú** modellek és algoritmusok alkalmazását.

---

## A rezolúciós kalkulus

Fogalmak:

- **literál**: az atomi formulák és negáltjaikat **literáloknak** nevezzük
- **klóz**: literálok halmaza (implicit diszjunkció) (?)
- **üres klóz**: literálok üres halmaza, jele:  $\square$  (kielégíthetetlen)

---

<sup>5</sup>pl. meleg van, valaki magas stb.

<sup>6</sup>0 - hamis, 1 - igaz

- **formula:** klózek halmaza (implicit diszjunkció) (?)
- **üres klózhalmaz:** jele:  $\emptyset$  (érvényes)
- **például:**

Formula:  $(p \vee q) \wedge (\neg p \vee \neg q)$

$\Downarrow$

Klózhalmaz:  $\{\{p, r\}, \{\neg p, \neg q\}\}$

**A rezolúciós szabály:** **(definíció)** Ha  $C_1 = \{\dots, l, \dots\}$  és  $C_2 = \{\dots, l^c, \dots\}$ <sup>7</sup> klózek **komplementens literálpárt** tartalmaznak, akkor **rezolválhatók**.

- az "eredmény" a **rezolvens klóz:**  $C = C_1 - \{l\} \cup C_2 - \{l^c\}$
- **például:**  $C_1 = ab\bar{c}$  és  $C_2 = bc\bar{e}$ , akkor az eredménye  $C = (ab\bar{c} - \{\bar{c}\}) \cup (bc\bar{e} - \{c\}) = ab \cup b\bar{e} = ab\bar{e}$

**A rezolúciós eljárás:** **(definíció)** Egy klózhalmaznál eldönti, hogy **kielégíthető-e**.

Legyen az  $S$  halmaz. Ekkor:

1. Vegyünk  $S$ -ből két rezolválható klózt.
2. Készítsük el a rezolvenst.
3. Ha a rezolvens az üres klóz ( $\square$ ), akkor a formula **kielégíthetetlen**. Különben adjuk hozzá a rezolvenst a klózhalmazhoz.
4. Álljunk meg, ha nincs több rezolválható klóz.

---

**A logikai program és az SLD rezolúció. A logikai programozás alapvető módszerei:**

Imperatív program: olyan mint egy recept. Meghatározza, hogy mit és milyen sorrendben kell megtenni a programot futtató gépnek.

---

<sup>7</sup> ahol  $(l, l^c)$  komplementens literálpár



- **logikai program:**
  - **tudást ír le**, melyet a gép a neki feltett kérdésekre való válaszoláshoz felhasznál
  - logikai állítások halmaza
  - **futása:** következtetési folyamat
- **legismertebb logikai nyelv**<sup>8</sup>: *prolog*
  - a programnak meg kell adnunk egy **célformulát** (céklózt), ezután ellenőrzi, hogy a céklóz a logikai program logikai következményei között van-e
  - **döntési eljárás:** SLD (*Linear resolution with Selection function for Definitive clauses*) elsőrendű rezolúciós kalkulus (??kifejteni)
- gyakori **példaprogram:** rokoni kapcsolatot adunk meg, melyből további rokonsági kérdésekre kaphatunk választ

---

<sup>8</sup>a '70-es évektől kezdve

## 6. Tétel

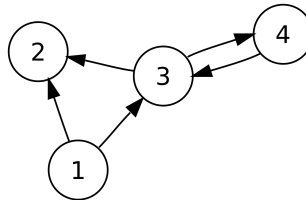
6.1. Gráf fogalma és megadásának módjai. Egyszerű, irányított és irányítatlan gráfok. Séta, út, összefüggőség. Nevezetes gráfok: páros gráf, teljes gráf, fa, kör, súlyozott gráf.

A gráf fogalma:

- **hálós adatszerkezet**: minden adatelemnek **tetszőleges számú megelőzője és rákövetkezője** lehet
- **csúcsok** és **élek** (két csúcs közötti kapcsolat) halmaza
- egy gráfot az határoz meg, hogy mely csúcsai vannak élekkel összekötve

A gráf megadási módjai:

1. Ábrával:



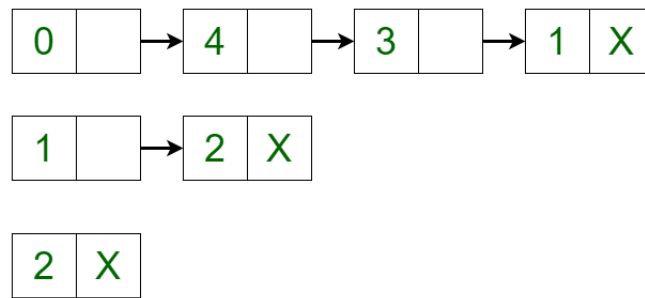
10. ábra. Irányított gráf reprezentációja ábrával

2. Az  $N$  pontthalmaz (csúcshalmaz) és az  $A$  élhalmaz tételes felsorolásával

$$N = \{0, 1, 2, 3, 4\}$$

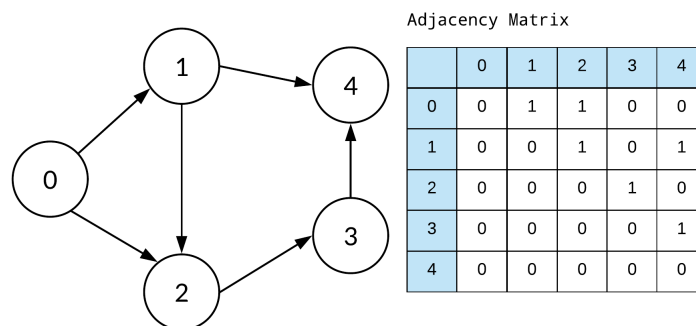
$$A = \{(0, 4), (0, 3), (0, 1), (1, 2)\}$$

3. **Szétszort reprezentáció**: szomszédsági (*multi*)**listával**: A kezdő csúcspontból listaszerűen felsoroljuk az onnan elérhető csúcspontokat.



11. ábra. Gráf szétszort reprezentációja

4. Folytonos reprezentáció: szomszédsági (*csúcs*)mátrixsal:



12. ábra. Gráf folytonos reprezentációja

A mátrix **sorai**: az adott indexű csúcsból kiinduló **csúcs-oszlopokhoz** 1-est rakunk, a többi pedig 0-nak hagyjuk.

A mátrix **oszlopai**: az adott indexű csúcsba menő **csúcs-sorokhoz** 1-est rakunk, a többi pedig 0-nak hagyjuk.

$$\text{Képletesen: } c[i, j] = \begin{cases} 1, & \text{ha } (i, j) \in E \\ 0, & \text{ha } (i, j) \notin E \end{cases}$$

---

**Egyszerű gráf:**

- bármely két csúcsa között **legfeljebb egy él** lehet (kizárjuk a többszörös éleket, illetve a hurkokat)

**Írányított gráf:** ebben az esetben az éleknek irányuk van.

**Írányítatlan gráf:** az élekhez nincs irány rendelve, vagyis nincs különbség  $A \rightarrow B$  és

$B \rightarrow A$  között.

---

**Séta:** **(definíció)** A gráf csúcsainak és éleinek egy olyan **sorozata**, melyben (?) minden végpontja megegyezik a következő él kezdőpontjával – feltéve ha létezik következő él.

A sétában a csúcsok és az élek tetszés szerint ismétlődhetnek.

**Út:** **(definíció)** Útnak nevezzük a csúcsok és élek olyan sorozatát, amelyben nem ismétlünk sem éleket, sem csúcsokat.

**Összefüggőség:** **(definíció)** Egy gráf összefüggő, ha (élei esetleges irányításáról megfeledkezve/eltekintve) bármely két csúcsa között van út.

---

**Nevezetes gráfok:**

1. **Páros gráf:** Egy gráf páros, ha *nincs benne páratlan hosszúságú kör*.

- körnek nevezzük azt az utat, amelynek kezdő- és végpontja azonos

2. **Teljes gráf:** Olyan gráf, melynek *bármely két csúcsa között van él*.

3. **Fa:** Olyan gráf, amely *összefüggő és körnélküli*.

$$\text{A fa csúcsainak száma} = \text{élek száma} + 1$$

4. **Súlyozott gráf:** Súlyozott gráf esetében egy  $w$  súlyfüggvény is rendelhető az élekhez.

## 6.2. Generatív nyelvtanok, nyelvosztályok, a Chomsky-hierarchia. Véges automáták, lineáris idejű felismerés, veremautomaták.

Generatív nyelvtanok:  $G = (N, \Sigma, S, P)$ , ahol

- $N$  – a **nemterminális** abécé
- $\Sigma$  – a **terminális** abécé
- $S$  – a **kezdőszimbólum**
- $P$  – a **helyettesítési szabályok**

Továbbá:

- $L(G)$ , a  $G$  grammatika által generált nyelv.
- A  $G$  grammatika generálja a  $w$  szót, ha  $S \Rightarrow^* w$  (vagyis ha  $S$ -ből levezethető  $w$ )

**Például:**  $N = \{S\}, \Sigma = \{a, b\}, S \in N, P = \{S \rightarrow \lambda \mid S \rightarrow aSb\}$  ( $\lambda$  az **üres szó**)

**Nyelvosztályok:** a nyelvtanok bizonyos formai tulajdonságok alapján **nyelvosztályokba** sorolhatóak.

Noam Chomsky alkotta meg az alábbi 4 nyelvosztályt:

1. **Rekurzívan felsorolható nyelvtanok:**  $\alpha \rightarrow \beta$  alakú, ahol

- $\alpha$  és  $\beta$  nemterminálisokból és terminálisokból álló szavak, és  $\alpha$  tartalmaz legalább egy nemterminális szimbólumot.

2. **Környezetfüggő nyelvtanok:**  $\alpha A \beta \rightarrow \alpha \gamma \beta$  vagy  $S \rightarrow \lambda$ , ahol

- ahol  $A$  nemterminális,  $\gamma$  egy nemterminálisokból és terminálisokból álló, akár üres szó,  $\alpha$  és  $\beta$  nemterminálisokból és terminálisokból álló szavak

3. **Környezetfüggetlen nyelvtanok:**  $A \rightarrow \alpha$ , ahol

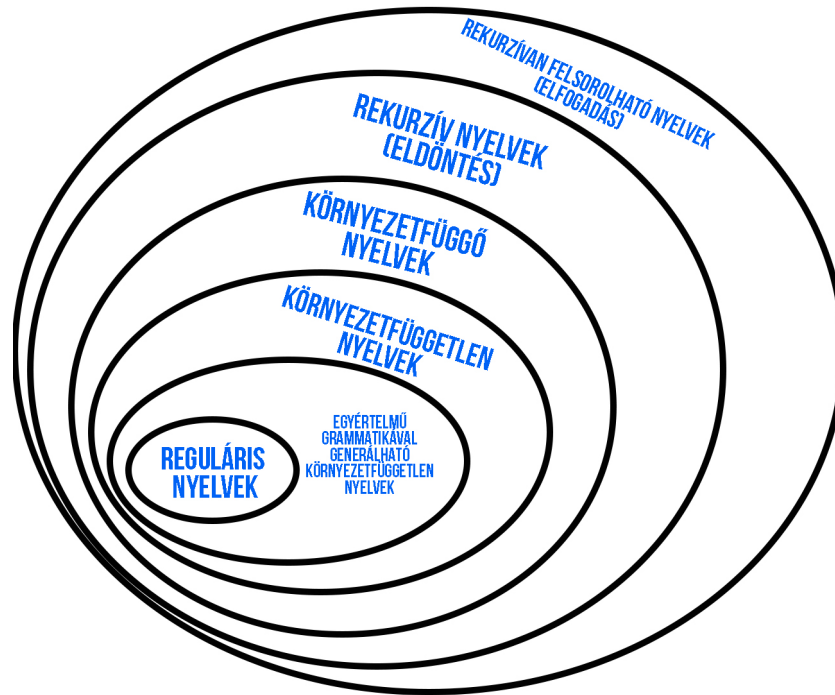
- ahol  $A$  nemterminális,  $\alpha$  egy nemterminálisokból és terminálisokból álló, akár üres szó

4. Reguláris/szabályos nyelvtanok:  $A \rightarrow aB|a|\lambda$ , ahol

- $A, B$  – nemterminálisok,  $a$  – terminális és  $\lambda$  üres szó
- reguláris nyelv<sup>9</sup> –  $\{a, b\}^* \Leftrightarrow (a + b)^*$  – reguláris kifejezés

---

Chomsky-hierachia:



13. ábra. Chomsky-hierarchia

**Chomksy normálforma:** **(definíció)** Egy grammatika *Chomsky normálformában* van, ha csak  $A \rightarrow BC$  és  $A \rightarrow a$  alakú szabályokat tartalmaz.

---

**Rekurzív vs. rekurzívan felsorolható nyelv:**

- **rekurzív nyelv:** ha van olyan  $T$  Turing gép, ami minden bemeneten megáll
  - $w \in L$  szavak esetén **elfogadó állapotban** áll meg
  - $w \notin L$  szavak esetén pedig **nem elfogadó állapotban** áll meg

Ekkor a  $T$  Turing gép **eldönti**  $L$ -et.

---

<sup>9</sup>vagyis reguláris kifejezéssel leírható

- **rekurzívan felsorolható nyelv:** ha van olyan  $T$  Turing gép, ami
  - minden  $w \in L$  szó esetén **elfogadó állapotban** áll meg
  - $w \notin L$  szavak esetén pedig **nem elfogadó állapotban** áll meg vagy **egyáltalán nem áll meg**

Ekkor a  $T$  Turing gép **elfogadja**  $L$ -et.

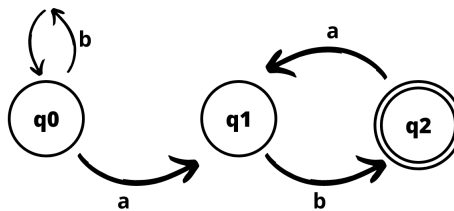
---

**Determinisztikus véges automata:**  $M = (Q, \Sigma, q_0, A, \delta)$ , ahol

- $Q$  – véges állapothalmaz
- $\Sigma$  – véges bemeneti ábécé
- $q_0 \in Q$  – kezdőállapot
- $A \subseteq Q$  – vég/elfogadási állapotok
- $\delta : Q \times \Sigma \rightarrow Q$  – állapotátmenet függvény

**Például:**  $M = (\{q_0, q_1, q_2\}, \{a, b\}, q_0, q_2, \delta)$ , ahol  $\delta$  a következő:

$$\delta(q_0, b) = q_0, \delta(q_0, a) = q_1, \delta(q_1, b) = q_2, \delta(q_2, a) = q_1$$



14. ábra. Determinisztikus véges automata

**Nemdeterminisztikus véges automata:** ugyan az a definíciója mint a determinisztikus véges automatának, a következő eltéréssel:

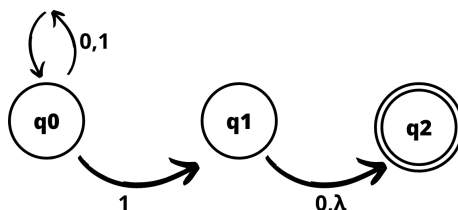
$$\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$$

Vagyis több lehetőség van ugyanarra a bemenetre, és megjelenik az **üresszó átmenet**.

**Például:**  $M = (\{q_0, q_1, q_2\}, \{0, 1\}, q_0, q_2, \delta)$ , ahol  $\delta$  a következő:

	0	1
q0	q0	q0q1q2
q1	q2	-
q2	-	-

1. Táblázat.  $\delta$  szabályai



15. ábra. Nemdeterminisztikus véges automata

---

**Lineáris idejű felismerés:** **(definíció)** A reguláris nyelvek esetén a szóprobléma nagyon hatékonyan megoldható.

Ha megszerkesztünk egy, az adott nyelvet elfogadó determinisztikus véges automatát, akkor annak segítségével a szót betűnként elolvasva végig követve az automata futását (legkésőbb) a szó végére érve megkapjuk a választ a kérdésre:

- ha végállapotba jutottunk a szó végén, akkor a szó benne van az adott reguláris nyelvben
- ha nem végállapotba jutottunk, vagy (*parciális automata* esetén) időközben elakadtunk a feldolgozással, akkor a keresett szó nincs a nyelvben

Tehát a probléma **valós időben megoldható**, ahány betűből áll az input szó, annyi lépés után tudjuk a választ.

---

**Veremautomaták:** **(definíció)**  $M = (Q, T, \Gamma, q_0, Z_0, \delta, F)$ , ahol

- $Q$  – állapothalmaz
- $T$  – bemeneti ábécé
- $\Gamma$  – veremábécé
- $q_0 \in Q$  – kezdőállapot
- $Z_0 \in \Gamma$  – kezdeti veremtartalom



- $\delta$  – állapotátmenet reláció/függvény
- $F \subseteq Q$  – végállapotok halmaza

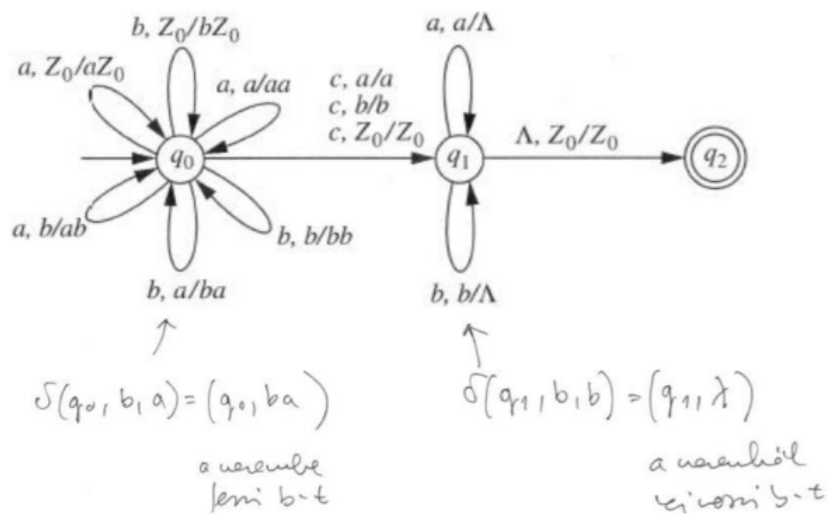
Megjegyzések:

- *véges automata + verem-memória*  $\rightarrow$  *veremautomata*
- a veremautomaták által megadható nyelvek azok, amelyek **környezetfüggetlen grammatikával generálhatóak**

Megjegyzések: Legyen a  $b, Z_0/a$  szabály. Ekkor ezt a következőképpen értelmezzük:

- ha  $b$ -t olvasok és  $Z_0$  van a verem tetején, akkor  $a$ -t írok a helyébe
- ennek segítségével el lehet dönteni egy adott szóról, hogy elfogadja-e a veremautomata (vagy akár az adott környezetfüggetlen nyelv) vagy sem

Például:



16. ábra. Veremautomata

Move number	State	Input	Stack symbol	Move(s)
1	$q_0$	$a$	$Z_0$	$(q_0, aZ_0)$
2	$q_0$	$b$	$Z_0$	$(q_0, bZ_0)$
3	$q_0$	$a$	$a$	$(q_0, aa)$
4	$q_0$	$b$	$a$	$(q_0, ba)$
5	$q_0$	$a$	$b$	$(q_0, ab)$
6	$q_0$	$b$	$b$	$(q_0, bb)$
7	$q_0$	$c$	$Z_0$	$(q_1, Z_0)$
8	$q_0$	$c$	$a$	$(q_1, a)$
9	$q_0$	$c$	$b$	$(q_1, b)$
10	$q_1$	$a$	$a$	$(q_1, \Lambda)$
11	$q_1$	$b$	$b$	$(q_1, \Lambda)$
12	$q_1$	$\Lambda$	$Z_0$	$(q_2, Z_0)$
	(all other combinations)			none

17. ábra. Veremautomata táblázatos reprezentációja

## 7. Tétel

### 7.1. Lineáris egyenletrendszer fogalma és megoldása Gauss eliminációval.

Lineáris egyenletrendszerek: **(definíció)** Az

$$\begin{aligned}a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &= b_1 \\a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &= b_2 \\&\vdots \\a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n &= b_m\end{aligned}$$

alakú egyenletrendszert, ahol  $a_{ij} (i \in \{1, \dots, m\}, j \in \{1, \dots, n\})$  és a  $b_k (k \in \{1, \dots, m\})$  valós számok ismertek,  $x_1, \dots, x_n$  ismeretlenek, **lineáris egyenletrendszernek** nevezzük, ha:

1.  $a_{ij}$ : az **egyenletrendszer együtthatói**
2.  $b_k$ : **szabad tagok**, vagy **konstansok**
3. az egyenletrendszer **alaplátrixa**, illetve **kibővített mátrixa**:

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix} \quad \text{és} \quad A|b = \left( \begin{array}{ccc|c} a_{11} & \dots & a_{1n} & b_1 \\ a_{21} & \dots & a_{2n} & b_2 \\ \vdots & & \vdots & \vdots \\ a_{m1} & \dots & a_{mn} & b_m \end{array} \right)$$

**Megjegyzés:** a lineáris egyenletrendszer mátrixos alakja  $Ax = b$ . (?)

**Egyenletrendszer megoldhatósága:** **(definíció)** A lineáris egyenletrendszer

- **megoldható:** ha van megoldása, azaz létezik olyan  $(x_1, \dots, x_n)$  vektor, hogy  $Ax = b$  fennáll. Továbbá, ezen megoldás lehet:
  - **határozott**, ha pontosan 1 megoldása van
  - **határozatlan**, ha több megoldása van
- **ellentmondásos:** ha nincs megoldása

Továbbá, egy lineáris egyenletrendszer pontosan akkor oldható meg, ha  $\text{rang}(A) = \text{rang}(A|b)$ . Egyébként pedig, ha  $\text{rang}(A) < \text{rang}(A|b)$ , akkor határozatlan.

**Egyenletrendszer homogenitása:** Ha  $b = 0$  akkor **homogén**, egyébként **inhomogén**.

**Gauss elimináció:** Az alábbi ekivalens állítások nem változtatják meg a lineáris egyenletrendszer megoldáshalmazát:

- Egyenlet szorzása  $\lambda \neq 0$ -val. (Egyenletek  $\leadsto$  kibővített mátrix sorai)
- Egy egyenlethez hozzáadni egy másik egyenlet  $\lambda$ -szorosát.
- Egyenletek sorrendjének megváltoztatása.
- Elhagyni olyan egyenletet, amely egy másik egyenlet  $\lambda$ -szorosa.
- Ismeretlenek felcserélése együtthatóikkal együtt, minden egyenletben. (alaplátrixban: *oszlopcsere*)

Ezen átalakítások segítségével az egyenletrendszer kibővített mátrixát (ahol a mátrix sorai felelnek meg 1-1 egyenletnek) **trapéz alakúra** hozzuk (főátló alatt csupa nulla), ahonnan **visszahelyettesítéssel** adódnak a megoldások.

- Ha az eljárás közben  $(0 \dots 0 | \neq 0)$  sor adódik, akkor az egyenletrendszer **ellentmondásos**.
- Ha az eljárás végén  $n$  sor marad, akkor az egyenletrendszer **határozott**, ha kevesebb, akkor **határozatlan**.

**7.2. Determinisztikus Turing-gépek, lineárisan korlátozott automáták, eldönthetetlen problémák, tár és idő korlátok. Nemdeterminisztikus Turing-gépek, nevezetes nyelvosztályok, P, NP.**

**Determinisztikus Turing-gép:** **(definíció)** A  $T = (Q, \Sigma, \Gamma, q_0, \#, \delta, F)$  rendezett hetest *Turing-gépnek* nevezzük, ahol

- $Q$ : a gép **állapotainak** véges halmaza
- $\Sigma$ : a **bemeneti ábécé**
- $\Gamma$ : a **szalagábécé**
- $q_0 \in Q$ : a **kezdő állapot**
- $\# \in (\Gamma \setminus Q)$ : a **szóköz betű**
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{\text{Bal, Jobb, Helyben}\}$ : a gép **átmenetfüggvénye/mozgásfüggvénye**
- $F \subseteq Q$ : **végállapotok** halmaza

**Jellemzői:**

1. A gép írhat és olvashat is a szalagról.
2. Az író-olvasó fej balra és jobbra is mozoghat, vagy akár egy helyben is maradhat.
3. A szalag végtelen.
4. Két speciális állapot: elfogadás és elutasítás

**Konfiguráció:** állapot + szalag tartalma + a fej helye

- **például:**  $\Delta q_1 01 \Delta$ 
  - melynek **jelentése:** a fej a 0-án van, a  $q_1$  állapotban, illetve a  $\Delta$  az üres betűt jelöli

**Lineárisan korlátozott automaták:** **(definíció)** Az  $LBA = (Q, \Sigma, \Gamma, q_0, \#, \delta, F)$ -t **lineárisan korlátozott automatának** hívjuk, ahol

- $Q$ : állapotok halmaza
- $\Sigma$ : bemeneti ábécé
- $\Gamma \supseteq \Sigma$ : a szalagábécé
- $q_0 \in Q$ : kezdő állapot
- $\# \in (\Gamma \setminus \Sigma)$ : a szóköz betű
- $\delta : \begin{cases} Q \times (\Gamma \setminus \{\#\}) \rightarrow Q \times \Gamma \times \{\text{Bal}, \text{Jobb}, \text{Helyben}\} \\ Q \times \{\#\} \rightarrow Q \times \{\#\} \times \{\text{Bal}, \text{Jobb}, \text{Helyben}\} \end{cases}$
- $F \subseteq Q$ : **végállapotok** halmaza

**Megjegyzések:**

- Tulajdonképpen a Turing-gép egy olyan változata, ahol a számításra fordítható szalagterület az input által lefoglalt területre korlátozódik.
- A lineárisan korlátozott automatákkal elfogadott nyelvek osztálya megegyezik a **környezetfüggő nyelvek** osztályával.

**Eldönthetetlen problémák:** **(definíció)** Azt mondjuk, hogy ha egy probléma algoritmikusan megoldható, akkor egy Turing gép is meg tudja oldani. Ha pedig egy problémát nem lehet Turing géppel megoldani, akkor algoritmussal sem. **(Church-Turing tézis)** Továbbá, egy probléma algoritmikusan megoldható, ha a hozzá tartozó nyelv **rekurzív**. Egy ilyen megoldhatatlan probléma **Hilbert 10. problémája**:

- **maga a probléma:** *Létezik-e olyan eljárás, ami eldönti, hogy egy egész együtthatós polinomnak van-e egész gyöke?*
- **állítás/válasz:** Nem létezik olyan eljárás, ami képes eldönteni, hogy egy egész együtthatós polinomnak van-e egész gyöke. Vagyis, az ezt reprezentáló nyelv **nem rekurzív**, hanem csak **rekurzívan felsorolható**!

---

**Tár és idő korlátok:** **(definíció)** Ahhoz, hogy egy pontos fogalmat tudjunk adni az algoritmus (Turing gép) gyorsaságáról, **le kell korlátoznunk** annak *számolási idejét* vagy a felhasznált *tárcellák számát*.

**Időkorlátosság:**

- **Determinisztikus Turing gép:** **(definíció)** A  $TM$  Turing gép  $t(n)$  **időkorlátos**, ha  $n$  hosszú inputokon, **legfeljebb  $t(n)$  lépést tesz**.
- **Nemdeterminisztikus Turing gép:** **(definíció)** Egy  $NTM$  nemdeterminisztikus Turing gép  $t(n)$  **időkorlátos**, ha  $n$  hosszúságú inputon,  $NTM$  minden számítási út mentén **legfeljebb  $t(n)$  lépést téve megáll**.

**Tárkorlátosság:** Legyen  $s : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$  egy függvény, melyre  $\forall n \in \mathbb{Z}^+$  számmal igaz, hogy  $s(n) \geq \log_2(n)$ .

**(definíció)** A  $TM$  Turing gép  $s(n)$  **tárkorlátos**, ha  $n$  hosszú inputokon **legfeljebb  $s(n)$  tárcellát használ a munkaszalagon** (azaz  $STM(n) \leq s(n)$ ).

---

**Nemdeterminisztikus Turing gép:** A nemdeterminisztikus Turing gépek esetén az átmenet függvény a következő:

$$\delta : Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{\text{Bal, Jobb, Helyben}\}} \quad (?)$$

ahol a  $Q \times \Gamma \times \{\text{Bal, Jobb, Helyben}\}$  halmaz összes részhalmazának a halmaza.

- **Továbbá, (definíció)** Az  $NTM$  nemdeterminisztikus Turing gép elfogadja az  $x$  inputot, ha a  $NTM$ -et  $x$  bemenettel a kiinduló helyzetből indítva, van legalább egy elfogadó (egy elfogadó állapotban véget érő) számítási út.
- Illetve, egy nemdeterminisztikus Turing gép szimulálható egy **3 szalagos, determinisztikus Turing géppel**.

---

**Nevezetes nyelvosztályok:** lásd a **6.2-es tételt**.

---

**Bonyolultsági osztályok (P, NP):**

1. **P osztály:** **(definíció)** Azon nyelvek (problémák) unióját, amelyekhez van olyan **determinisztikus Turing gép**, ami ezek szavait valamilyen polinomfüggvénnyel megadható időben eldönti (kiszámítja), **P bonyolultsági osztálynak** nevezzük.

2. **NP osztály:** **(definíció)** Azon nyelvek (problémák) unióját, amelyekhez van olyan **nemdeterminisztikus Turing gép**, ami ezek szavait valamilyen polinomfüggvénnyel megadható időben eldönti (kiszámítja), **NP** bonyolultsági osztálynak nevezzük.

**Megjegyzés:**  $P$  és  $NP$  viszonya  $P \subseteq NP$ , viszont egyelőre nem ismert olyan feladat (nyelv) ami  $NP$ -ben van és bizonyítottan nincs  $P$ -ben.

**Példák NP problémára:** 3-szín probléma, maximális méretű klikk, Hamilton-út probléma stb.



## 8. Tétel

### 8.1. Statisztikai minta és becslések, átlag és szórás. Konfidenciaintervallumok.

Az  $u$ -próba.

**Statisztikai minta:** **(definíció)** Statisztikai mintának nevezünk  $n$  független, azonos eloszlású  $X_1, X_2, \dots, X_n$  valószínűségi változót, amelyek közös eloszlása megegyezik a vizsgált  $X$  változó eloszlásával.

A minta numerikus értékeit ezen változók felvett értékeinek tekintjük.

**Átlag:** ...

**Szórás:** a szórás egy valószínűségi változó értékeinek a **várható értéktől való eltéréseinek a mértéke**.

---

**Becslések:**

1. **Torzítatlan becslés:** **(definíció)** A  $T$  statisztikát a  $t$  paraméter **torzítatlan becslésének** nevezzük, ha  $\mathbb{E}T = t$ . (E?)
  - a **torzítatlanság** azt jelenti, hogy a becslés a becsülendő paraméter körül ingadozik
2. **Konzisztens becslés:** **(definíció)** A  $T_n$  sorozatot a  $t$  paraméter **konzisztens becslésének** nevezzük, ha  $T_n \rightarrow t$ -be képez *sztochasztikusan*.
3. **Maximum-likelihood becslés:** **(definíció)** Legyen  $X_1, X_2, \dots, X_n$  minta egy **diszkrét eloszlásból**,  $x_1, x_2, \dots, x_n$  pedig a **minta realizáció**. Legyen  $\vartheta$  az ismeretlen paraméter. Az

$$L(x_1, \dots, x_n; \vartheta) = P(X_1 = x_1, \dots, X_n = x_n) = \prod_{i=1}^n P(X_i = x_i)$$

függvényt **likelihood-függvénynek** nevezzük.

---

**Konfidenciaintervallumok:** **(definíció)** Legyen  $\vartheta$  ismeretlen paraméter,  $T_1$  és  $T_2$  két statisztika. Azt mondjuk, hogy a  $[T_1, T_2]$  intervallum  $1 - \alpha$  **megbízhatósági szintű konfi-**

**dencia intervallum**  $\vartheta$ -ra, ha  $P(\vartheta \in [T_1, T_2]) \geq 1 - \alpha$ .

---

**Az  $u$ -próba** (vagy másnéven  $Z$ -teszt): A statisztikai hipotézisek vizsgálatára próbákat (teszteket) alkalmazunk. A legegyszerűbb ilyen próba az  $u$ -próba.

**Egymintás  $u$ -próba:** **(definíció)** Legyen  $X_1, \dots, X_n$  minta  $\mathcal{N}(0, 1)$  eloszlásból. Tegyük fel, hogy a  $\sigma^2$  ismert. Az  $m$  várható értékre az előírás  $m_0$ . Tehát a

$$H_0 : m = m_0$$

nullhipotézist kell vizsgálnunk a

$$H_1 : m \neq m_0$$

alternatív hipotézissel (*ellenhipotézis*) szemben.  $H_0$  fennállása esetén az

$$u = \frac{\bar{X} - m_0}{\sigma} \sqrt{n}$$

statisztika **standard normális eloszlású**.

**Kétmintás  $u$ -próba:** Legyen az  $X_1, \dots, X_n$  minta  $\mathcal{N}(m_1, \sigma_1)$  eloszlásból és az  $Y_1, \dots, Y_n$  minta  $\mathcal{N}(m_2, \sigma_2)$  eloszlásból.

A várható értékre végzünk hipotézisvizsgálatot.

A nullhipotézis:  $H_0 : m_1 = m_2$ .

Az alternatív vagy ellen hipotézis:  $H_1 : m_1 \neq m_2$ .

Vagyis **kétoldali alternatív hipotézisünk van**. Ekkor a **próbastatisztika a következő**:

$$u = \frac{\bar{X}_n - \bar{Y}_n}{\sqrt{\frac{\sigma_1^2}{n} + \frac{\sigma_2^2}{m}}}, \quad \text{ahol } u \sim \mathcal{N}(0, 1).$$

**8.2. Az informatikai biztonság fogalma, legfontosabb biztonsági célok. Fizikai, emberi, technikai fenyegetések és ellenük való védekezés. Algoritmikus védelem eszközei: titkosítás, digitális aláírás, hash függvények. Az AES és RSA algoritmusok.**

**Az informatikai biztonság fogalma:** **(definíció)** Az informatikai biztonság az informatikai rendszer olyan, a védő számára **kielégítő mértékű állapota**, amely az informatikai rendszerben kezelt **adatok bizalmassága, sértetlensége és rendelkezésre állása**, illetve a **rendszerelemek sértetlensége és rendelkezésre állása** szempontjából: *zárt, teljes körű, folytonos és a kockázatokkal arányos*.

- **zárt védelem:** olyan védelem, amely az összes releváns fenyegetést figyelembe veszi
- **teljes körű:** olyan védelem, amely a rendszer összes elemére kiterjed
- **folytonosság:** a védelem az időben változó körülmények és viszonyok ellenére is folyamatosan megvalósul
- **kockázatokkal arányosság:** egy kellően nagy intervallumban a **védelem költségei arányosak a potenciális kárértékkel**

**Biztonsági célok - CIA hármas:**

- **Confidentiality** (bizalmasság): Titkos vagy személyes információk (privacy) nem kerülhetnek jogosulatlanok kezébe. A bizalmasságot az adatok tárolásánál, feldolgozásánál és továbbításánál is garantálni kell.
- **Integrity** (sértetlenség): Két fogalom:
  - **Data integrity** (*adatintegritás*): Teljesülésekor az adat jogosulatlanul nem módosult tárolása, feldolgozása vagy továbbítása során.
  - **System integrity** (*rendszer sértetlensége*): A rendszer működése az elvártak megfelelő, jogosulatlan módosításoktól mentes.
- **Availibilty** (*rendelkezésre állás*): Biztosítja, hogy a szolgáltatás az arra jogosultak számára a szükséges időben és időtartamra használható.

**További biztonsági célok:** nyomon követhetőség és garancia, biztosíték.

**Sérülési szintek:** alacsony, közepes és magas

---

**Fizikai védelem:** azon rendszerek védelme, amik az adatok **tárolását, feldolgozását** és **továbbítását** biztosítják. Többségük *preventív* vagy *detektív*.

**Fizikai infrastruktúra:**

- Informatikai rendszer hardver elemei
- Épületek
- Kiszolgáló rendszerek
- Személyzet

**Fizikai fenyegetések és az ezekre irányuló védelmi intézkedések**

• **Környezeti fenyegetések, természeti csapások:**

- Nem megfelelő hőmérséklet és/vagy páratartalom  $\Leftrightarrow$  mérőeszközök installálásával
- Víz  $\Leftrightarrow$  vízérzékelők installálása
- Por  $\Leftrightarrow$  szűrővel felszerelt ventillátor használata
- Tűz, füst  $\Leftrightarrow$  automata tűzoltó rendszer telepítése és/vagy egyebek

• **Technikai fenyegetések:**

- Elektromos teljesítmény, elektromágneses interferencia  $\Leftrightarrow$  szünetmentes tápegység használata

• **Emberi fizikai fenyegetések:**

- Jogosulatlan fizikai hozzáférés, lopás, vandalizmus, visszaélés  $\Leftrightarrow$  csak az arra jogosult léphessen be az épületbe, zárható tárolók használata és egyéb intézkedések.

---

## Algoritmusok védelme, avagy titkosítási sémák

**Szimmetrikus titkosítási séma:** *titkosító kulcs* = *visszafejtő kulcs* (vagy a visszafejtő kulcs a titkosító kulcsból polinomiális időben könnyen kiszámítható):

- **(definíció)** Az  $SE = (Key, Enc, Dec)$  hármas egy **szimmetrikus titkosítási séma**, ahol
  - **Key: kulcsgeneráló algoritmus**, amely egy  $k$  biztonsági paraméterhez (kulcs méretére utal) megad egy  $K \in \mathcal{K}$  titkos kulcsot.
  - **Enc: titkosító algoritmus**, amely  $\forall m \in \mathcal{P}$  nyílt üzenethez és  $\forall K \in \mathcal{K}$  titkos kulcshoz generál egy  $c \in \mathcal{C}$  titkosított üzenetet. (vagyis  $^{10}c = Enc_K(m)$ )
  - **Dec: visszafejtő algoritmus**, amely egy  $c \in \mathcal{C}$  titkosított üzenethez és egy adott  $K \in \mathcal{K}$  kulcshoz megad egy  $m \in \mathcal{P}$  nyílt üzenetet. (vagyis  $m = Dec_K(c)$ )

**Aszimmetrikus titkosítási séma:** *titkosító kulcs*  $\neq$  *visszafejtő kulcs* (vagyis, a visszafejtő kulcs a titkosító kulcsot felhasználva csakis nehezen számítható ki – nem ismerünk rá polinomiális idejű algoritmus):

- **(definíció)** Az  $AE = (Key, Enc, Dec)$  hármas egy **aszimmetrikus titkosítási séma**, ahol
  - **Key: kulcsgeneráló algoritmus**, amely egy  $k$  biztonsági paraméterhez (kulcs méretére utal) megad egy  $(PK, SK) \in \mathcal{K}$  nyilvános és titkos kulcsból álló párt.
  - **Enc: titkosító algoritmus**, amely  $\forall m \in \mathcal{P}$  nyílt üzenethez és  $PK$  nyilvános kulcshoz generál egy  $c \in \mathcal{C}$  titkos üzenetet. (vagyis  $c = Enc_{PK}(m)$ )
  - **Dec: visszafejtő algoritmus**, amely egy  $c \in \mathcal{C}$  titkosított üzenethez és egy adott  $SK$  titkos kulcshoz megad egy  $m \in \mathcal{P}$  nyílt üzenetet. (vagyis  $m = Dec_{SK}(c)$ )

---

<sup>10</sup> $c \Leftrightarrow cyphertext$

### Szimmetrikus és aszimmetrikus titkosítás összehasonlítása:

	Szimmetrikus	Aszimmetrikus
Kulcsok titkossága	kulcsok titkosak ( $K$ )	$(PK, SK)$ nyilvános és titkos kulcs
Kulcsok kezelése	kulcsere algoritmus	Nyilvános Kulcs Infrastruktúra ( <i>Public Key Infrastructure</i> )
Időigény	gyors algoritmusok	lassú algoritmusok
Üzenetek mérete	nagy méretű	kis méretű
Példák	TDES, AES	<b>RSA</b> , ElGamal, elliptikus görbe titkosítás

**Digitális aláírás:** **(definíció)** A digitális aláírási séma egy  $DNS = (Key, Sign, Ver)$  hármas, ahol

- **Key:** A **Key kulcsgeneráló algoritmus** a  $k$  biztonsági paraméterre kiszámítja a  $(PK, SK)$  kulcspárt, ahol  $PK$  nyilvános és  $SK$  titkos.
- **Sign:** A **Sign aláíró algoritmus** az  $SK$  titkos kulcshoz és az  $m \in \{0, 1\}^*$  üzenetre generál egy  $s = Sign_{SK}(m)$  aláírást.
- **Ver:** A **Ver ellenőrző algoritmus** a  $PK$  nyilvános kulcsra, az  $m$  üzenetre, és az  $s$  aláírássra **IGAZ** vagy **HAMIS** értéket ad vissza. **IGAZ** esetén az aláírás érvényes, **HAMIS** esetén érvénytelen.

**Hash függvények:** **(definíció)** A  $H : \{0, 1\}^* \rightarrow \{0, 1\}^n, n \in \mathbb{N}$  függvényt **hash függvénynek** nevezzük.

- a hash értéket **lenyomatnak** is hívjuk
- **lavinahatás:** Egy bit változása az inputban, jelentős változást eredményez az outputban. (pl. az output fele) (?)
- a hash függvények **nem injektívek**

**AES** (Advanced Encryption Standards – eredeti nevén **Rijndael**)

Szimmetrikus titkosítási séma, amely  $4 \times 4$ -es mátrixokat használ a titkosítás során.

Az eljárás részletes leírása:

1. A **tényleges kulcsok előállítása** a nyers kulcsból a *Rijndael-féle módszerrel*

2. **Előkészítés:**

(a) AddRoundKey

3. **Ciklusonként ismétlődő lépések:**

(a) SubBytes

(b) ShiftRows

(c) MixColumns

(d) AddRoundKey

4. **Utolsó ciklus** (nincs MixColumns lépés)

(a) SubBytes

(b) ShiftRows

(c) AddRoundKey

---

**Euler-féle  $\varphi$  függvény:**  $\varphi(n) = n \prod_{p|n} \frac{p-1}{p}$ , ahol  $p$  prím.

**Euler-Fermat tétel:** **(definíció)** Ha  $^{11}(a, m) = 1$ , akkor  $a^{\varphi(m)} \equiv 1 \pmod{m}$ .

• **kis Fermat tétel:**

1. Ha  $p$  prímszám,  $a \in \mathbb{Z}$  és  $^{12}p \nmid a$ , akkor  $a^{p-1} \equiv 1 \pmod{p}$ .

2. Ha  $p$  prímszám és  $a \in \mathbb{Z}$ , akkor  $a^p \equiv a \pmod{p}$ .

---

<sup>11</sup>vagyis ha  $a$  és  $m$  **relatív prímek**

<sup>12</sup> $\nmid$  – nem osztja

## **RSA (Rivest, Shamir, Adleman)**

Aszimmetrikus titkosítási séma:  $AE = (Key, Enc, Dec)$ , ahol

- *Key*:

1. Véletlenül választunk két nagy prímet:  $p, q \rightarrow$  **Prímtesztek**, pl. Miller-Rabin prímteszt
2. Kiszámítjuk az RSA modulust:  $n = p \cdot q$ .
3. Kiszámítjuk az Euler-féle  $\phi$  függvényt:  $\phi(n) = (p - 1)(q - 1)$ .
4. Választunk egy véletlen  $e$  egészt:  $1 < e < \phi(n)$  és  $(e, \phi(n)) = 1$ . ( $e$  titkosító kitevő/komponens)  $\rightarrow$  **Euklideszi algoritmus**
5. Kiszámítjuk  $d$ -t:  $1 < d < \phi(n)$  és  $e \cdot d \equiv 1 \pmod{\phi(n)}$ . ( $d$  visszafejtő kitevő/komponens)  $\rightarrow$  **Kibővített Euklideszi algoritmus**

$PK = (n, e)$ ,  $SK = d$  és  $\phi(n)$ ,  $p$ ,  $q$  titkos paraméterek.

Illetve,  $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$

- $Enc_{PK}(m) = m^e \pmod{n}$ ,  $\forall m \in \mathcal{P}$  és  $PK = (n, e)$  mellett.  $\rightarrow$  **Gyors hatványozás**
- $Dec_{SK}(c) = c^d \pmod{n}$ ,  $\forall c \in \mathcal{C}$  és  $SK = d$  mellett.  $\rightarrow$  **Kínai Maradéktétel alkalmazása**



# Informatikai ismeretek

9. **Tétel:** Adatbázisrendszerek. Adatbázis, adatbázisrendszer, adatbázis-kezelő rendszer (DBMS) fogalma és jellemzői. Egyed, tulajdonság és kapcsolat fogalma és tulajdonságai. Relációs, objektum-relációs és NoSQL adatbázisok jellemzése. A funkcionális függés fogalma. Koncepcionális adatbázis-tervezés, az ER modell és leképezése relációs modellre. Az SQL elemei: DDL, DML, DCL, egyszerű lekérdezések és táblák összekapcsolása.

**Adatbázis:** **(definíció)** Egymással logikailag összefüggő, egymáshoz kapcsolódó adatok összessége.

**Adat:** **(definíció)** Olyan ismert **tény**, amely *számszerűsíthető* és *implicit* (magától értetődő) *jelentése* van.

**Adatbázis-kezelő rendszer (DBMS<sup>13</sup>):** **(definíció)** Olyan szoftvercsomag/rendszer, amely számítógépes adatbázisok *létrehozását* és *karbantartását* támogatja.

Feladatai:

- adatbázis **definiálása** adatípusai, szerkezete és megszorításai révén
- adatbázis tartalom **betöltése** a (másodlagos) tároló eszközön
- adatbázis **kezelése**: kinyerés (keresés), módosítás, adatbázis elérése web alkalmazásokon keresztül
- feldolgozás és megosztás konkurens felhasználók és alkalmazói programok egy halmaza között úgy, hogy az **összes adat érvényes és konzisztens** marad.

**Adatbázisrendszer:** DBMS + adatok (illetve néha az alkalmazásokat is beleértjük)

---

<sup>13</sup>Database Management System

## Az adatbázis megközelítés főbb jellemzői:

### 1. Az adatbázis önleíró természete:

- Egy DBMS **katalógus** egy önálló adatbázis leírását (**metaadatokból**<sup>14</sup> áll) tárolja.
- Lehetővé teszi a különböző adatbázis alkalmazásokkal való együttműködést.

### 2. A programok és az adatok elszigetelése:

- **Program-adat függetlenségnek** nevezzük
- Lehetővé teszi a módosításokat a DBMS program megváltoztatása nélkül

### 3. Adat absztrakció: Egy adatmodellt használunk arra, hogy a tárolási részleteket elrejtjük.

### 4. Az adatok többféle nézetének támogatása: minden felhasználó különböző képet láthat az adatbázisról.

### 5. Adatok megosztása és többfelhasználós tranzakció feldolgozás: tranzakciók és konkurens felhasználók jelenléte.

---

**Egyed:** **(definíció)** A valós világnak az az eleme (tárgy, jelenség, személy stb.), amely a modellezés tárgyát képezi. (pl. Gipsz Jakab végzős PTI-s hallgató)

**Tulajdonság:** **(definíció)** Az egyednek a modellezés szempontjából lényeges **jellemzője**. (pl. a Gipsz Jakab név)

**Kapcsolat** **(definíció)** A két vagy több egyedtípus egyedei között fennálló **viszony**.

---

## Relációs adatbázis

Részei:

- **Reláció:** értékek egy táblázata, amely sorok egy halmazából áll.
- **Sor:** a modellezett kisvilág **egyed-előfordulásáról** vagy egy **kapcsolat-előfordulásáról** tartalmaz tényeket (információt).

---

<sup>14</sup>az adatra vonatkozó adat

- **Oszlopok:** minden oszlop egy **oszlop fejléccel** (címmel) rendelkezik, amely az illető oszlopban lévő adatok jelentéséről ad információt.
- **Reláció kulcsa:** minden sor rendelkezik egy olyan adatelem értékkel (vagy azok egy halmazával), amely egyértelműen azonosítja a sort a táblázatban. Ezt **kulcsnak** nevezzük.

---

**Objektum-relációs adatbázis:** a relációs adatbázis **kibővítése**. Belül minden **relációsan** működik. Erre egy **ráépülő réteggént** alakítják ki az objektum-orientált felületet.

- **Főbb bővítések:** kollekción típusok, metódusok, hivatkozások
- **Objektumok:** sorobjektum, oszlopobjektum, objektumazonosító
- **Táblák:** sorobjektumok halmaza

---

**NoSQL adatbázis:** gyorsan változó, **struktúrátlan adatok** nagy tömegének kezelésére szolgál. Típusai:

- **Kulcs-érték:** pl. Redis
- **Dokumentum:** pl. MongoDB
- **Oszlopalapú:** pl. Cassandra
- **Gráf:** pl. Neo4j

---

**A funkcionális függés fogalma:** **(definíció)** Az  $R$  két attribútumhalmaza,  $X$  és  $Y$  között  $X \rightarrow Y$ -nal jelölt **funkcionális függés** előír egy **megszorítást**, amelyek egy  $R$  fölötti  $r$  relációt alkothatnak. (alkothatnak?)

A megszorítás az, hogy  $\forall t_1, t_2 \in r$  rekordok esetén amelyekre  $t_1[X] = t_2[X]$  teljesül, teljesülnie kell  $t_1[Y] = t_2[Y]$ -nak is.

---

**Koncepcionális adatbázis-tervezés:**

- Minivilág meghatározása
- Követelményrendszer meghatározása (funkcionális analízis)
- Koncepcionális tervezés








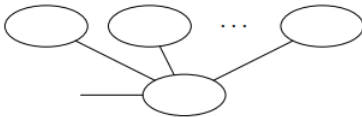
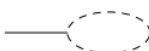

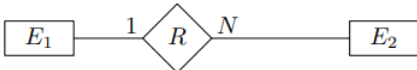
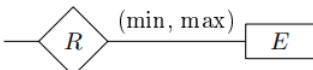
- Logikai tervezés
- Fizikai tervezés
- Tranzakcióimplementáció
- Alkalmazói programok

**ER modell** (Entity Relationship):

- **Gyenge egyedtípus:** azon egyedtípusok, amelyek nem rendelkeznek saját *kulcsattribútumokkal*:
  - csak részleges kulcsuk (*diszkriminátor*) van amelyek egyértelműen azonosítják azokat a gyenge egyedeket, amelyek **ugyanazon tulajdonos egyedekhez kapcsolódnak**
- **Erős egyedtípus:** rendelkezik saját kulcsattribútummal

**ER séma leképezése:**

1. Erős egyedtípusok leképezése.
2. Gyenge egyedtípusok leképezése.
3. Bináris 1 : 1 számosságú kapcsolattípusok leképezése (külső kulcs használata, összevonás, kereszthivatkozás vagy kapcsoló reláció használata)
4. Bináris 1 :  $N$  számosságú kapcsolattípusok leképezése.
5. Bináris  $M$  :  $N$  számosságú kapcsolattípusok leképezése.
6. Többértékű attribútumok leképezése.
7.  $N$ -edfokú kapcsolattípusok leképezése.

Szimbólum	Jelentés
	egyedtípus
	gyenge egyedtípus
	kapcsolattípus
	azonosító kapcsolattípus
	attribútum
	kulcsattribútum
	többértékű attribútum
	összetett attribútum
	származtatott attribútum
	az $E_2$ egyedtípus totális résztvevője a $R$ kapcsolatnak
	az $E_1$ és $E_2$ egyedtípusok $1 : N$ számosságú $R$ kapcsolata
	az $E$ egyedtípus $R$ -beli részvételére vonatkozó strukturális megszorítás (min, max)

18. ábra. ER séma jelölései

---

Az SQL elemei (Structured Query Language): strukturált adatbázis-lekérdező nyelv

- DDL (*Data Definition Language*): create, alter, drop, truncate → **táblákra vonatkozóan**
- DML (*Data Manipulation Language*): insert, update, delete → egy **tábla soraira vonatkozóan**
- DCL (*Data Control Language*): tranzakciókezelés (commit, rollback, savepoint) és jogosultságkezelés (grant, revoke)

### Lekérdezések részei:

- SELECT: attribútumok megadásához
  - FROM: táblák megadásához
  - WHERE: feltételek megadásához
  - GROUP BY: csoportképzés megadott attribútumok alapján
  - HAVING: csoportlekérdezésre vonatkozó feltételek megadásához
  - ORDER BY: rendezés megadásához
- 

### Táblák összekapcsolásának módjai:

#### 1. Belső kapcsolat:

- INNER JOIN: megegyező, **előre meghatározott**, attribútumú sorokból áll
- NATURAL JOIN: **automatikusan** történik az összecsatoláshoz használt attribútum kiválasztása (nem mindig szolgál optimális megoldásként)

#### 2. Külső kapcsolat:

- LEFT OUTER JOIN: A bal oldalon megadott táblából azok a sorok is az eredménybe kerülnek, amelyekhez nincs a jobb oldalon megadott táblában megfelelő érték, NULL értékek kerülnek az eredményben ezek helyére.
- RIGHT OUTER JOIN: A jobb oldalon megadott táblából azok a sorok is az eredménybe kerülnek, amelyekhez nincs a bal oldalon megadott táblában megfelelő érték, NULL értékek kerülnek az eredményben ezek helyére.
- FULL OUTER JOIN: Mindkét megadott táblából az eredménybe kerülnek azok a sorok is, amelyeknek nincs a másik táblában megfelelő sora.

## 10. Tétel: Lexikális egységek. Adattípusok. Nevesített konstans. Változó. Kifejezések. Utasítások. Programegységek. Paraméterkiértékelés, paraméterátadás. Blokk. Hatáskörkezelés, láthatóság. Absztrakt adattípus. Kivételkezelés.

**Lexikális egységek:** **(definíció)** A program szövegének azon elemei, melyeket a fordító a lexikális elemzés során **felismer** és **tokenizál**<sup>15</sup>.

Elemei:

- **Kulcsszavak**
  - fentartott, foglalt angol szavak
  - sosem használhatóak azonosítóként
  - pl. `for`, `if`, `int` stb.
- **Azonosítók**
  - kötelezően betűvel kezdődik
  - használatukkor ajánlatos **névkonvenciókat** alkalmazni: `CamelCase`, `snake_case` stb.
  - **standard azonosító:** pl. `NULL`
- **Konstans szövegliterálok:** pl. `"Hello"`
- **Konstansok:** pl. `INT_MAX`
- **Operátorok:** `+`, `-`, `++`, `--`, stb.
- **Szeparátorok:** pl. zárójelek, vessző stb.
- **Megjegyzések:** ezek használata nem befolyásolja a program futását, csupán a programozónak szolgál útmutatásként.

---

<sup>15</sup>közbenső formára hoz

---

**Adattípusok:** egy **absztrakt programozási eszköz**, amely egy konkrét programozási eszköz egy komponenseként jelenik meg.

Egy adattípus három dolgot határoz meg: *tartomány*, *műveletek* és *reprezentáció*.

Típusai:

1. **Egyszerű típusok:** szerkezetileg nem bonthatóak tovább.

**Fajtái:** egész, valós (lebegőpontos), logikai, karakter

2. **Összetett típusok:** az összetett adattípusnál az értékhalmoz mellett a **szerkezet** is lényeges, egyik-másik típus ábrázolása már magasabb szintet igényel, mert elég bonyolult.

**Fajtái:** tömb, vektor, rekord, szöveg (**string**), sor, verem, lista stb.

---

**További adattípusok:** mutatók (memóriacímre mutatnak)

---

**Nevesített konstans:**

- mindig deklarálni és inicializálni kell (?)
- pl. Java-ban: `final double PI = 3.14;` (mely egyben lehet `public static` is)

---

**Változó:** négy komponense van:

- **Név:** azonosító
  - **Attribútumok:** típus körülhatároláshoz használatos, deklaráció használatával, melyek lehetnek:
    - **explicit, implicit, automatikus**
  - **Cím:** tárkiosztásra, hatáskörre vonatkozik (hogyan legyen tárolva a változó a memóriában), mely lehet:
    - **statikus, dinamikus** vagy a **programozó által vezérelt tárkiosztás**
  - **Érték:** a változó által felvett, és az attribútumok által behatárolt érték.
-



**Kifejezések:** szintaktikai eszközök, melyek formálisan a következő összetevőkből állnak:

- **operandusok** és **operátoraik**, melyek sorrendje lehet:

- **prefix:** ++a
- **infix:** a+1
- **postfix:** a++

- **kerek zárójelek**

---

**Utasítások:** a fordító ezeknek a segítségével generálja a **tárgyprogramot**.

Fajtái:

- Értékadó utasítás: `a = 2;`
- Üres utasítás
- Ugró utasítás: általánosan használt alakja a `GOTO`
- Elágazó utasítás: `if, else`
- Ciklusvezérlő utasítás: `for, while`
- Hívó utasítás
- Vezérlésátadó utasítások: `BREAK, CONTINUE, RETURN`

---

**Programegység:** **(definíció)** egy olyan **szintaktikus egység**, mely a program tagolásának, programrészek újrafelhasználhatóságának az eszköze.

Ilyenek például az: alprogramok, blokkok, csomagok, osztályok stb.

---

**Paraméterek:** függvényeknél használatosak, melyek átadása lehet:

1. **Érték-alapú:** az objektum egy másolatát adjuk át
  2. **Referencia-alapú:** az eredeti objektumot adjuk át
-

**Blokk:** A blokk olyan **programegység**, amely csak másik programegység belsejében helyezkedhet el, külső szinten nem állhat.

- van kezdete (`{`), törzse és vége (`}`)

---

**Hatáskörkezelés, láthatóság:** ezek határozzák meg, hogy a változók a program mely részein használhatóak.

---

**Absztrakt adattípus:** információ elrejtésre használatos, tehát nem ismerjük sem a **representációt**, sem a **műveletek implementációját**.

---

**Kivételkezelés:** a programban előforduló hibák kezelésére használatos. Ezen kivételek lehetnek:

- **Checked:** fordítási időben előjönnek, pl. a program nem talál egy használni kívánt fájlt vagy osztályt
- **Unchecked:** futás közben jönnek elő, pl. aritmetikai kivétel (nullával való osztás)
- **User defined:** a felhasználó által definiált

**Szintaxisa:** `try {...} catch (...) finally {...}`

**11. Tétel:** Az objektumorientált paradigma alapfogalmai. Osztály, objektum, példányosítás. Öröklődés, osztályhierarchia. Polimorfizmus, metódustúlterhelés. A bezárási eszközrendszer. Absztrakt osztályok és interfészek. Típus-tagok.

**Programozási paradigma (PP):**

- A program felépítésére használatos **eszközkészlet**, vagyis hogy milyen egységek képezik a program alkotóelemeit.
  - moduláris programozás, objektumorientált programozás stb.

**Objektumorientált paradigma (OOP):**

- Minden entitás **osztályokkal** van leírva/megadva.
- **Objektumok:** osztályok példányai, amelyek **állapotokat** (mezők, attribútumok, stb.) és **viselkedést** (metódusok) zárnak magukba.

**Egységbezárás (encapsulation):**

- Olyan részletek elrejtése, amelyekre csak az adott osztálynak van szüksége.
- Egy egyszerű, tiszta **interfész** biztosítása.

**Osztály:** Ez egy felhasználói típus, amelynek alapján példányok (objektumok) hozhatóak létre. **Adat** és **metódus** definíciókat tartalmaz.

**Példányosítás:** Egy adott sablonnal meghatározott adatszerkezet **műveletvégzésre alkalmas** példányba történő lemásolása, létrehozása.

---

**Öröklődés:** egy olyan mechanizmus, amely alkalmazásával egy osztályt (*gyerekosztály*) **származtathatunk** egy másiktól (*szülő osztály*).

- általában a többszörös öröklődés nincs megengedve a legtöbb nyelvben, a konfliktusok elkerülése végett

- az öröklődés által **osztályhierarchiát** hozhatunk létre, amelyekben az osztályok megosztanak egymással különböző *állapotokat* és *viselkedéseket*
- ezáltal a kód olvashatóbb és könnyedén újra felhasználható marad

**Osztályhierarchia:** Egy osztályhierarchia alatt azt értjük, hogy van egy *ősosztály* (*superclass*), amelyből minden későbbi osztály örököl tulajdonságokat.

- az *ősosztály* olyan alapvető tulajdonságokkal rendelkezik, amikre a leszármazott osztályoknak szükségük lesz
- minél mélyebben helyezkedik el egy *ősosztályból* leszármazott osztály, annál specifikusabb állapotokkal és viselkedéssel rendelkezik

---

**Polimorfizmus** (többalakúság): A polimorfizmus egy **egységes interfészre** utal, amit különböző típusok valósítanak meg.

Ezáltal egy *ősosztály* típusú változó hivatkozhat **ugyanazon közös őosztályból származó** osztályok példányaira.

A polimorfizmus többféleképpen is megvalósítható:

- **Ad hoc polimorfizmus** (statikus): A függvények különböző implementációival való **túlterhelése**.
- **Paraméteres polimorfizmus** (statikus): A kódot **általánosan írják meg** különböző típusok számára, amely alkalmazható az összes típusra. Objektumorientált környezetben **sablonnak** vagy **generikusnak** nevezik.
- **Altípusosság** (dinamikus): A név több **különböző osztály példányait jelöli**, amelyeknek a függvényt deklaráló **közös őse van**. Objektumorientált környezetben többnyire erre gondolnak, amikor polimorfizmusról beszélnek.

Továbbá, a polimorfizmus lehet **statikus** vagy **dinamikus**:

- **statikus polimorfizmus:** metódusok túlterhelése, függvénysablonok, osztállysablonok
  - statikus, fordításidejű kötés

- **előnye:** gyorsabb végrehajtású, mivel nem kell dinamikus kötéseket figyelni
- **hátránya:** a fordítónak többet kell dolgoznia, illetve több memóriát használ
- **dinamikus polimorfizmus:** metódusok felülírása
  - dinamikus, futásidejű kötés
  - **előnyei és hátrányai:** dinamikusabb, de lassítja a futást

**Metódustúlterhelés:** A metódustúlterhelés lehetővé teszi számunkra, hogy több, **ugyanazzal a névvel** ellátott metódus létezzon, amennyiben **eltérő paraméterekkel** rendelkeznek.

Háromféleképpen tudunk túlterhelni egy metódust:

1. A paraméterek megegyeznek, de számuk eltérő.
2. A paraméterek eltérőek.
3. A paraméterek sorrendje eltérő.

**Megjegyzés:** mindez nem vonatkozik a metódus visszatérési értékére.

**A bezárási eszközrendszer:** a **láthatóság szabályozásához** használatos. A Java programozási nyelvben 4 áll rendelkezésünkre:

- **private:** csak az adott osztályban látható, amelyben létrehozták
- **default/package private:** csak az adott csomagban használható (ha nem adunk meg láthatóságot, akkor ez az alapértelmezett)
- **protected:** csak a (csomagbeli, vagy csomagon kívüli) leszármazott osztályokban látható
- **public:** mindenhol láthatóak

---

**Absztrakt osztályok:**

- **nem példányosíthatóak**, de örökölni lehet belőlük
- metódusait vagy implementáljuk, vagy csak definiáljuk a szignatúrájukat

- ezeket tudják majd a leszármazott osztályok opcionálisan felülírni (amennyiben nem **abstract** metódusokról beszélünk, mivel ezeket kötelező felülírni a leszármazott osztályban)

### Interfészek:

- **szereződést határoznak meg**
- absztrakció + egységbezárás
- bármennyi osztály implementálhatja őket
- az interfészben csak a metódus szignatúráját adjuk meg
  - ám bár újabban (Java 8-tól kezdődően) lehetőségünk van **default** implementációt megadni egy interfészbeli metódushoz

---

### Típustagok:

- **összetevők:** név + hozzáférhetőség
- ötféle egyedtípus lehet típustag: mező, metódusok, beágyazott típus, tulajdonságok és események
- **típusrendszer:** különféle *díszítőelemeket* definiál, amelyeket tagokhoz rendelhetünk hozzá
  - pl. a Python-beli díszítőelemeket **dekorátoroknak** nevezzük, ilyen például a metódusokra opcionálisan helyezett **@staticmethod** dekorátor

## 12. Tétel: Operációs rendszerek fogalma, felépítése, osztályozásuk. Fájlok és fájlrendszerek. Speciális fájlok Unix alatt. Átírányítás, csővezetékek. Folyamatkezelés. Jelzések, szignálok. Ütemezett végrehajtás

**Operációs rendszer fogalma:** **(definíció)** egy programrendszer, amely **közvetítő szerepet** játszik a számítógép felhasználója és a számítógép hardvere között.

**Céljai:**

- felhasználói programok végrehajtása
- a számítógép rendszer használatának megkönnyítése és hatékonyabbá tétele

**Szerkezeti felépítése:**

- **Kernel** (*mag*): az OS központi része, melynek feladata az erőforráskezelés, memóriakezelés, folyamatok ütemezése stb.
  - **kernel típusok:** monolitikus kernel, hibrid kernel, mikrokernél, exokernél
- **Rendszermag** (*system core*): kiegészíti a kernel funkcióit, amely magasabb szintű feladatokat lát el, mint például a hálózati kommunikáció
- **Rendszerszolgáltatások:** fájlkezelés, hálózati szolgáltatások stb.
- **Fájlrendszer:** a fájlok és könyvtárak tárolására és szervezésére szolgáló rendszer
- **Felhasználói felület:** mindez lehet parancssori felület (*CLI*), vagy grafikus felület (*GUI*)
- **Eszközillesztők:** lehetővé teszik az operációs rendszer és a hardvereszközök közötti kommunikációt és együttműködést

**Az operációs rendszer osztályozása:**

- Felhasználói felület szerint
  - **karakteres:** UNIX, DOS

- **grafikus:** Apple Mac OS
  - Felhasználók száma szerint
    - **egy-felhasználós:** BeOS
    - **több-felhasználós:** Microsoft Windows XP
    - **hálózati:** Novell Netware
  - Folyamatkezelés szerint
    - **kötegelt:** Microsoft MS DOS
    - **multiprogramozott**
    - **valós idejű:** BeOS
    - **időosztásos:** UNIX
- 

**Fájlok:** információstárolási egység

**Komponensei:** fájlnev, kiterjesztés, méret, dátum, idő, hely stb.

**Műveletek:** create, write, read, seek, delete, truncate, open, close

**Elérési útvonal:** relatív (./myFolder) vagy abszolút (C://Users//MyUser//myFolder)

**Fájltrendszerek:** A lemezek **partíciókra** bonthatóak, amik vagy nyerssek, vagy formázottak egy fájlrendszerrel.

A fájlrendszer az alábbi részekből áll:

- Fájlok
- Katalógusok (könyvtárak)
- Fájlmetaadatok
- Blokkok és szektorok
- Jogosultságkezelés: **rw**x (read, write, execute) jogosultságok kezelése

**Példák:** NTFS, FAT32, extFAT, ext3, hálózati fájlrendszerek stb.

---



## Speciális fájlok UNIX alatt:

- **link**: hivatkozás másik fájlra
  - a linket egy 'l' betű azonosítja (pl. `lrwxrwxrwx termcap`)
  - továbbá, ez lehet **hard-link** (csak az adott fájlrendszerbe hivatkozhat fájlokra) vagy **soft-link** (felrúgja a hard-link szabályait)
- **nevesített csővezeték** (named pipe): folyamatok közötti kommunikációra ad lehetőséget, oly módon hogy az egyik alkalmazás kimenetét egy másik alkalmazás bemenetére köti
  - a nevesített csővezeték egy 'p' betű azonosítja (pl. `prw-rw----- mypipe`)
- **socket**: folyamatok közötti kommunikációra használatosak, de immáron **hálózatos környezetben**
- **eszköz** (device) fájlok: a hardver elemeit reprezentálják
  - ezáltal az utasításokban **közvetlenül** használhatjuk az eszközöket

---

## Átírányítás: három komponense van:

- **bemenet** (*stdin*): ahonnan a program a futás során a beérkező **adatokat olvassa**
- **kimenet** (*stdout*): ahova a program **ír**
- **hibakimenet** (*stderr*): ahova a program a futás során fellépő **hibaüzeneteket írja**

## Példák:

- `<...: stdin` átírányítása (a megadott fájlból olvas)
- `>...: stdout` átírányítása (a megadott fájlba ír, ha már létezik az adott fájl, felülírja a tartalmát)
  - `>>...: ez esetben hozzáfűzi a fájlhoz ha az nem üres`

- `2>...`: *stderr* átirányítása (a megadott fájlba írja a hibaüzeneteket)
- `&>...`: *stdout* és *stderr* átirányítása ugyanazon fájlba
- `2>&1...`: az *stderr*-t ugyan oda irányítja, ahova az *stdout* irányítva lett
- `1>&2...`: az *stdout*-ot ugyan oda irányítja, ahova az *stderr* irányítva lett

### Csővezetékek:

- programok egy olyan **sorozata**, amelyek folyamatokkal vannak összekötve
  - azaz a program1 kimenete a program2 bemenetére
- több programból álló csővezeték is létrehozható
- megadása `'|'` vonalakkal történik
- leggyakrabban az OS **I/O alrendszerén keresztül** kerül megvalósításra

### Folyamatkezelés:

- processzus  $\Leftrightarrow$  egy **futó program**
- a batch rendszerek  $\Leftrightarrow$  **job**-okat, míg az időosztásos rendszerek  $\Leftrightarrow$  **felhasználói programokat** vagy **task**-okat futtatnak
- processzusok **állapotai**: új, futó, várakozó, futásra kész, befejezett
- **processzus részei**: programkód, adatok, statisztikai információk, regiszterek, verem, fájlok és fájlleírók, csatlakoztatott erőforrások
- **futása**: előtérben vagy háttérben
- **speciális processzusok**:
  - **zombi** processzus: ha a szülő nem vár a gyerekekre. Egy processzus **árva** lesz, ha a szülője hamarabb törlődik mint ő.
  - **együttműködő** processzus: tudnak egymással kommunikálni, IPC<sup>16</sup> segítségével

<sup>16</sup>Inter Process Communication

A **processzusvezérlő blokk** (PCB<sup>17</sup>): minden futó processzust **létrehoz** és **karbantart**.

Ezeket használja fel az OS a processzusok *kezeléséhez* és *végrehajtásához*.

Elemei:

- Processzusazonosítás
  - Processzorállapot információ
  - Processzusvezérlő információ
- 

Jelzések, szignálok:

- **céljuk** hogy jelezzék egy processzusnak, hogy egy bizonyos esemény bekövetkezett
  - **menete**: generálódik → eljut a processzushoz → átadódik vagy a **gyári**, vagy a **user-defined** jelzéskezelőnek
- 

Ütemezett végrehajtás:

- **kritériák**: CPU kihasználtság, processzus végrehajtási ideje, várakozási idő, válasz-idő
- a **rövidtávú ütemező** választja ki a következő processzust
- a **diszpécser modul** adja a processzus kezébe a CPU irányítását, amely *kontextusváltással jár*

Ütemezési algoritmusok:

1. **First-Come, First-Served (FCFS)**: ahogyan vannak a sorban, úgy is választódnak ki
  - **konvoj effekt** alakulhat ki (?kifejteni)
2. **Shortest-Job First (SJF)**: CPU burst alapú ütemezés
3. **Prioritás alapú**: prioritásszám alapú ütemezés

---

<sup>17</sup>Process Control Block

4. **Round Robin:** minden processzus kap egy kicsi processzoridőt. Miután ez lejár neki, a sor végére kerül.
5. **Többosztintű sor:** két sor használata az ütemezéshez. Az első soron Round Robin ütemezés folyik, a másodikon pedig FCFS.

### 13. Tétel: Verziókezelés, verziókezelő rendszerek. Szoftvertesztelési alapfogalmak (tesztszintek, tesztípusok, teszttervezési módszerek). Objektum orientált tervezési alapelvek (GoF, SOLID). Függőségbefecskendezés. Architektúrális minták (MVC). Tervezési minták. Szabad és nem szabad szoftverek. Szoftverlicencek, szabad és nyílt forrású licencek fajtái

**Verziókövetés(kezelés):** olyan eljárások összesége, amelyek lehetővé teszik egy adathalmaz változatainak (verzióinak) együttes kezelését.

Ezen rendszerek lehetővé teszik:

- ugyanazon fájl különböző, időben eltérő változatainak a megtekintését
- egy állomány két változata közötti különbségeinek megtekintését
- elágazásokat (branch)-eket tudunk létrehozni (lásd **dev** és **feature** ágak)

Továbbá, manapság a rendszerek már **nem centralizáltak**, hiszen minden fejlesztő rendelkezik a saját lokális változatával, ezért nincs szükség centralizálásra.

**Verziókövető (kezelő) rendszerek:** Subversion, Git, Visual Studio Services stb.

---

**Tesztelés:**

- a szoftverfejlesztési életciklus minden részéhez kapcsolódó, **statikus** vagy **dinamikus** folyamat
- kapcsolatban áll a szoftvertermékek *tervezésével*, *elkészítésével* és *kiértékelésével*, hogy megállapítsa, hogy a szoftvertermék teljesíti-e a meghatározott követelményeket
- **célja:** a hibák megtalálása

**Tesztszintek:**

#### 1. **Egységtesztek** (unit tests):

- pontosan egy programegységet tesztel
- egymástól függetlenül működnek

- nincs mellékhatásuk

2. **Integrációs tesztek** (integration tests):

- programegységeket, modulokat, azok egymással és a környezettel történő együttműködését teszteli

3. **Rendszertesztek** (system tests):

- az integrációs tesztek speciális fajtája

4. **Átvételi tesztek** (acceptance tests):

- utolsó tesztelési szint, amely elválasztja a rendszerünket a való életben történő működéstől
- független fejlesztő végzi

5. **Alfa és béta tesztek** (alpha and beta tests):

- előre kiválasztott felhasználók végzik

**Teszt típusok:**

1. **Funkcionális**

2. **Nem-funkcionális:** teljesítmény, stressz, használhatóság

3. **Regressziós:** ha a szoftver egyáltalán nem működik

**Teszttervezési módszerek:**

- Tesztelő tapasztalata
- Specifikáció alapú
- Struktúra alapú (ismerjük a program felépítését)
- Hiba alapú (hibasejtés)
- Valószínűség (determinisztikus módon is lehet)

---

**GoF** (Gang Of Four - "négyek bandája"):

1. **Interfészre programozunk, ne implementációra!**

- a laza csatoltság alapelve

2. **Részesítsük előnyben az objektum-összetételt az öröklődéssel szemben!**

- mivel öröklődés esetében a szülőosztályból örökölt megvalósításokon nem tudunk változtatni
- ellenben az objektum-összetétel dinamikusan, futásidőben történik

**SOLID elvek:** Robert C. Martin ("Bob bácsi") által megfogalmazott elvek.

1. **Single-responsibility principle** (egyszeres felelősség elve): ha egy osztálynak egy-nél több felelőssége van, akkor egynél több oka van a változásra. Másszóval, egy osztálynak csak egy felelőssége lehet.
2. **Open-closed principle** (nyitva-zárt elv): a szoftvereknek nyitottnak kell lenniük a bővítésre, de zártnak kell lenniük a módosításra.
3. **Liskov substitution principle** (Liskov-féle szétválasztási elv): ha egy  $S$  típus  $T$  típus altípusa, nem változhat meg egy program működése, ha benne a  $T$  típusú objektumot  $S$  típusú objektummal helyettesítjük.
4. **Interface segregation principle** (interfész szétválasztási elv): nem szabad arra kényszeríteni az osztályokat, hogy olyan metódusoktól függjenek, melyeket nem használnak.
5. **Dependency inversion principle** (függőség megfordítási elv): magas szintű modulok ne függjenek alacsony szintű moduloktól.

**Egyéb objektum-orientált tervezési alapelvek:** DRY<sup>18</sup>, YAGNI<sup>19</sup>, KISS<sup>20</sup>, Demeter törvénye.

---

<sup>18</sup>Dont Repeat Yourself

<sup>19</sup>You Aren't Gonna Need It

<sup>20</sup>Keep It Simple Stupid

---

### Függőség befecskendezés:

- a **vezérlés megfordítása** (IoC<sup>21</sup>) nevű *architekturális minta* alkalmazásának egy speciális esete
- **lazán csatolt kód** fejlesztését teszi lehetővé
  - a lazán csatoltság **kiterjeszthetővé** teszi a kódot, a kiterjeszthetőség pedig **karbantarthatóvá**
- egy objektumra olyan **szolgáltatásként** tekintünk, melyet más objektumok, **kliensek** használnak

### Fogalmak:

- **Függőség** (dependency): egy kliens által igényelt **szolgáltatást** jelent, mely a feladatának ellátásához szükséges
- **Függő** (dependent): egy **kliens objektum**, melynek egy vagy több függőségre van szüksége a feladatának ellátásához
- **Objektum gráf** (object graph): a függő objektumok és függőségeik összessége
- **Befecskendezés** (injection): egy kliens függőségeinek megadását jelenti
- **DI konténer** (DI container):
  - függőségbefecskendezést támogató **programkönyvtár**
  - **IoC konténer** kifejezést is használják rá
  - a függőség befecskendezés történhet DI konténer nélkül is, ekkor **tiszta DI**-ről beszélünk

**Függőség befecskendezés előnyei:** kiterjeszthetőség, karbantarthatóság, tesztelhetőség stb.

---

<sup>21</sup>Inversion Of Control



**Minták:** *"A minta egy olyan ötlet, mely egy gyakorlati környezetben már hasznosnak bizonyult, és várhatóan más környezetben is hasznos lesz"* - Martin Fowler

- **részei:** környezet, probléma és megoldás

**Architekturalis minták:** szoftverrendszerek **alapvető szerkezeti felépítésére** adnak *sémákat*.

Ehhez előre definiált **alrendszereket** biztosítanak, melyeknek meghatározzák a felelősségi köreit, illetve, a közöttük lévő kapcsolat megszorításait.

**MVC** (*Model-View-Controller*):

- **Környezet:** rugalmas ember-gép felülettel rendelkező **interaktív alkalmazások**
- **Probléma:** igény a felhasználói felület gyakori változására
- **Megoldás:** az interaktív alkalmazás három részre osztása

**Megjegyzések:**

- a nézet elválasztása a vezérlőtől kevésbé fontos, sőt akár több vezérlő is tartozhat egy nézethez
- a modell **objektumokat csomagol be**

---

**Tervezési minták**

- **középszintű minták**, kisebb léptékűek az architekturalis mintáknál
- alkalmazásuknak **nincsen hatása** egy szoftverrendszer alapvető felépítésére
- függetlenek az adott programozási nyelvtől vagy programozási paradigmától

**Osztályozásuk céljuk szerint:**

- **Létrehozási minták:** az objektumok létrehozásával foglalkoznak
- **Szerkezeti minták:** azzal foglalkoznak, hogy hogyan alkotnak osztályok és objektumok nagyobb szerkezeteket

- **Viselkedési minták:** az osztályok vagy objektumok egymásra hatását valamint a felelősségek elosztását írják le
- 

#### **Létrehozási minták:**

1. **Egyke** (*singleton*): egy osztályból csak egy példányt engedélyez, és ehhez globális hozzáférési pontot ad meg
2. **Építő** (*builder*): az összetett objektumok felépítését függetleníti az ábrázolásuktól, így ugyanazzal az építési folyamattal különböző ábrázolásokat hozhatunk létre

#### **Szerkezeti minták:**

1. **Díszítő** (*decorator*): az objektumokhoz dinamikusan további felelősségi köröket rendel
2. **Illesztő** (*adapter*): az adott osztály interfészét az ügyfelek által igényelt interfészé alakítja

**Viselkedési minták:** bejáró, felelősséglánc, látogató, sablonfüggvény.

---

#### **Szabad és nem szabad szoftverek:**

- **Szabad szoftverek:** az alábbiakat jelenti:
    1. A szoftver tetszőleges célra szabadon **felhasználható**.
    2. A szoftver működése szabadon **tanulmányozható**.
    3. A szoftver másolatait szabadon **lehet terjeszteni**.
    4. A szoftver szabadon **módosítható**.
  - **Nem szabad szoftverek:** a használat, terjesztés és módosítás tilos, korlátozott, vagy engedélyhez kötött.
-

**Szoftverlicenc:** szoftverek **használatának** és **terjesztésének** módját szabályozó **jogi eszköz**.

**Szabad és nyílt forrású licencek fajtái:**

- **GNU General Public License (GPL):** ezen szoftverek szabadon felhasználhatóak és módosíthatóak.  
Viszont, a készített szoftver is GPL licenc alatt kell hogy kiadásra kerüljön.
- **MIT License:** ezen szoftverek szabadon felhasználhatóak és módosíthatóak.  
Továbbá, nem kötelező az eredeti forráskód közzététele.
- **Apache License:** ezen szoftverek szabadon felhasználhatóak és módosíthatóak.  
Továbbá, kötelező közzétenni az eredeti forráskódot.
- **BSD License:** hasonló a GPL-hez, azzal a különbséggel, hogy kevésbé korlátozó feltételeket tartalmaz.
- **Creative Common License**

14. **Tétel:** Hagyományos szoftverfejlesztési módszertanok: vízesés modell, V-modell, spirális fejlesztési modell, prototípus alapú fejlesztés, iteratív és inkrementális módszertanok, gyors alkalmazásfejlesztés. Agilis szoftverfejlesztési módszertanok: az agilis szoftverfejlesztés alapjai, az agilis kiáltvány, valamint egy szabadon választott agilis módszertan részletes bemutatása

Vízesés modell:

- **strukturált**, vagyis az elvégzendő feladatokat **modulokra bontja**
- ideális **nagy projektekhez**, melyek rugalmatlanok
- az életciklus **lineáris**, vagyis nincs lehetőség a követelmények módosítására
- részletes dokumentációt feltételez ez a módszertan

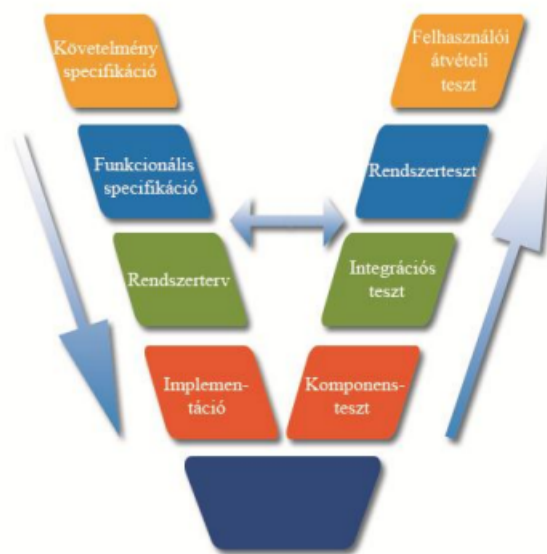


19. ábra. Vízesés modell

---

V-modell:

- a vízesés model továbbfejlesztése
- **részei:** fejlesztési- és teszt szár
- ha a teszt hibát talál, akkor a vele egy szinten lévő fejlesztési lépéshez kell visszatérni



20. ábra. V-modell

#### Fázisai:

1. **Követelmény specifikáció – Acceptance teszt:** üzleti elemzést foglal magába
2. **Funkcionális specifikáció – Rendszerteszt:** a program működését foglalja magába
3. **Rendszerterv – Integrációs teszt:** a rendszer konkretizált leírása (milyen osztályokkal, metódusokkal valósítanánk meg a megoldást stb.), illetve a komponensek együttműködését írja le
4. **Implementáció – Komponensteszt:** maga az implementáció

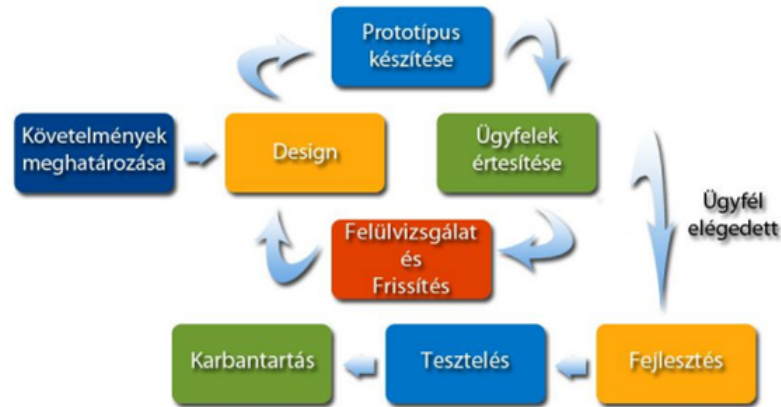
---

#### Prototípus modell:

- ez a **legelterjedtebb módszertan**
- a vízesés modell sikertelenségére válaszol, mely abból eredt, hogy a felhasználók a csak a fejlesztés végén találkoztak a szoftverrel, így nem derülhetett ki időben, hogy a felek esetleg félreértették egymást
- **célja:** a végső átadás előtt **több prototípust** kell leszállítani

- a fejlesztés során lehetőség van a változó követelményekhez alkalmazkodni

**Változatai:** eldobható prototípus, evolúciós prototípus.



21. ábra. Prototípus modell

---

**Spirális fejlesztési modell:**

- A **vizesés modell** és a **prototípus modell** egyes részeit kombinálja.
- Jelen van benne az **iteráció**.
- Minden ciklus végére egy működő prototípust kell előállítani.
- Nagy, bonyolult rendszerek esetén ajánlott.

**Fázisai:**

1. Célok meghatározása
2. Kockázatok azonosítása és feloldása
3. Fejlesztés és tesztelés
4. Következő iteráció tervezése

---

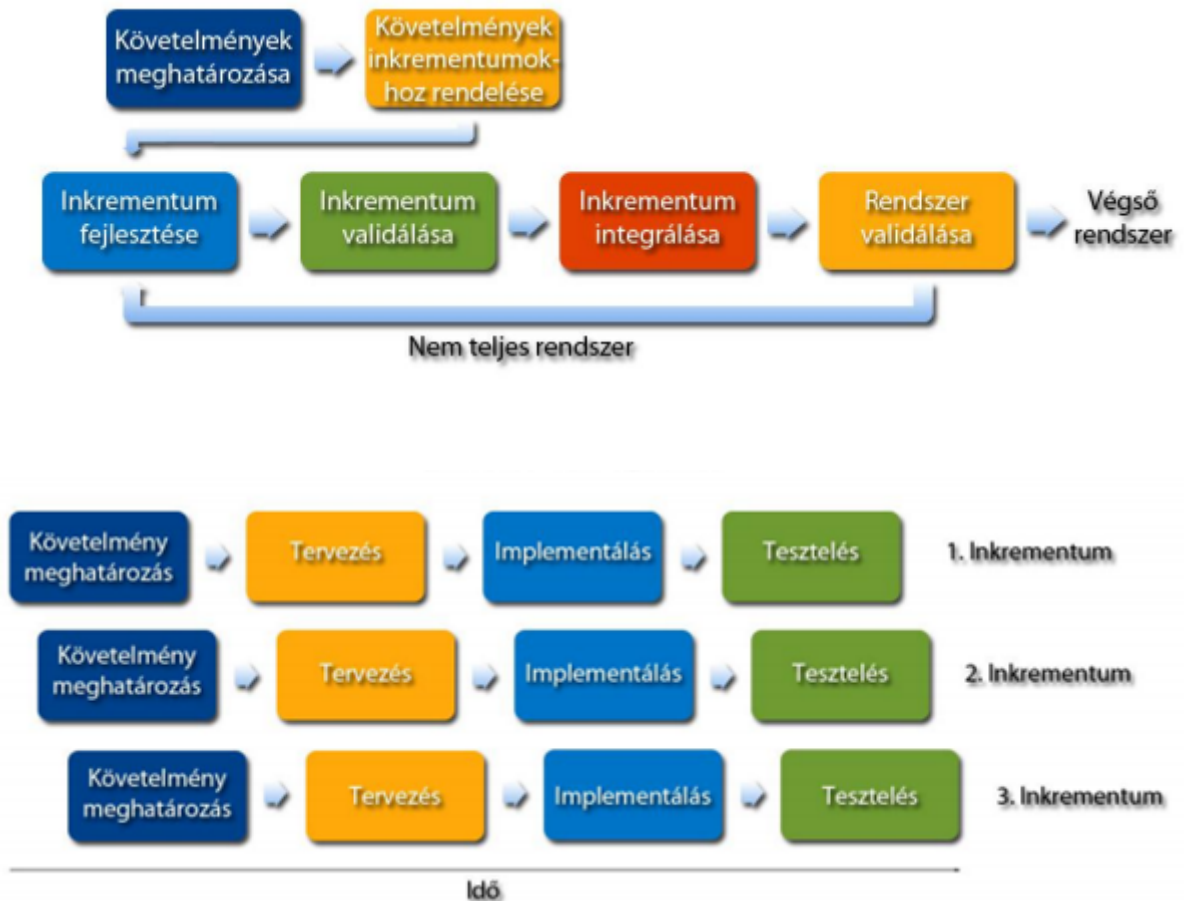
**Iteratív és inkrementális módszertanok:**

- A fejlesztést **iterációkra bontjuk**, melyek mindegyikében **tervezés és implementáció** megy végbe.

- Minden iteráció kiegészíti az előző iterációban elkészült prototípust.  
Ezt a kiegészítést nevezik **inkrementumnak**.

**Iteratív módszertan:** A hangsúlyt a folyamatra/iterációra helyezi.

**Inkrementális módszertan:** : A hangsúlyt az iterációk végtermékére helyezi.

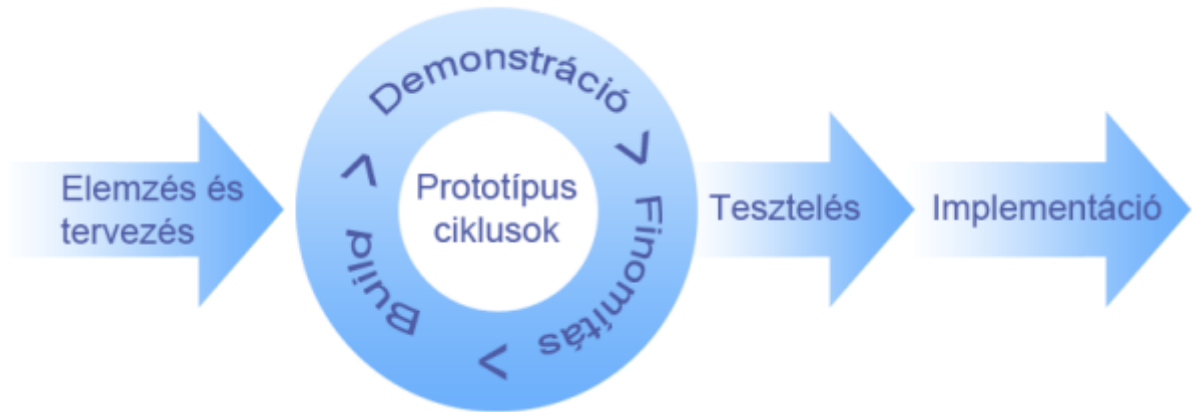


22. ábra. Inkrementális módszertan

**Gyors alkalmazásfejlesztés (RAD – Rapid Application Development):**

- Lényege, hogy a szoftvert **gyorsabban és jobb minőségben** készítjük el.
- Ehhez a következők szükségesek: korai prototípus készítése, jó csapat, szigorú ütemterv, jó időbeosztás stb.
- A RAD esetén előfordul, hogy a fejlesztők már **jól bevált komponenseket** használnak fel.

- **Ciklusokban** történik a fejlesztés.
- Ha a projekt nem modularizálható, akkor nem lehet alkalmazni.



**23. ábra.** Gyors alkalmazásfejlesztés – RAD

**Agilis szoftverfejlesztés** (adaptív): iteratív szoftverfejlesztési módszerek egy csoportjára utal. Alapelvei:

1. A legfontosabb a **megrendelő kielégítése**. (a szoftver gyors és folyamatos áttadásával)
2. Még a **követelmények kései változtatása** sem okoz problémát.
3. A **működő szoftver / prototípus átadása rendszeresen**, a lehető legrövidebb időn belül.
4. Az előrehaladás **alapja a működő szoftver**.
5. **Napi együttműködés** a megrendelő és a fejlesztők között.
6. A leghatékonyabb kommunikáció a **szemtől-szembeni megbeszélés**.
7. A projektek **motivált egyének köré épülnek**, akik megkapják a szükséges eszközöket és támogatást a legjobb munkavégzéshez.
8. **Önszervező csapatok** készítik a legjobb terveket.
9. Rendszeres időközönként a **csapatok reagálnak a változásokra**, hogy még hatékonyabbak legyenek.



10. **Egyszerűség**, a minél nagyobb **hatékonyságért**.
11. Folyamatos **figyelem a technikai kitűnőségnek**.
12. Az agilis folyamatok általi **fenntartható fejlesztés állandó ütemben**.

**Előnyei:** kevesebb dokumentáció, növekvő rugalmasság és csökkenő kockázat, könnyebb kommunikáció és javuló együttműködés, a megrendelő bevonása a fejlesztésbe.

---

**Scrum:** két fogalom: *sprint* (fejlesztési időszak) és *akadály* (olyan tényező, amely akadályozza a fejlesztést).

**Fejlesztési folyamat:**

1. A **Product Owner** létrehoz egy **Product Backlog**-ot, amelyre felhasználói **sztoriként** felviszi a teendőket.
2. A **Sprint Planning Meeting**-en a csapat tagjai megbeszélik, hogy mely sztorikat vállalják, továbbá elkészítik az időbeosztást.
3. A sprint folyamán a csapat és a **Scrum Master** *naponta* megbeszélik a történeteket.
4. A sprint végén van a **Sprint Review**, ahol a csapat bemutatja a sprint alatt elkészült sztorikat.
5. Ezután következik a **Sprint Retrospective**, ahol a sprint során felmerült problémákat vitatja meg a csapat, és konkrét javaslatokat tesznek.
6. Ezt követően újra a **Sprint Planning Meeting** következik.

**Szerepkörök:**

- **Scrum Master:** Felügyeli és segíti a csapat munkáját, ő a csapat menedzsere.
  - ő felel a csapat teljesítményéért
- **Product Owner:** A megrendelő szerepét tölti be.
  - ő felel a sztorik fontossági sorrendjének felállításáért

A Product Owner és a Scrum Master **nem lehet ugyanaz a személy**.

- **Csapat:** ideálisan 5-9 fő akik a bevállalt feladatokat végzik.
- **Üzleti szereplők:** megrendelők, forgalmazók, tulajdonosok. A Sprint Review során kapnak szerepet.
- **Menedzsment:** a megfelelő **környezetet biztosítja** a csapatok számára.

#### Megbeszélések:

- **Sprint Planning Meeting:** Itt dönti el a csapat, hogy pontosan mely sztorikat vállalja el a Backlog-ból.
- **Backlog Grooming:** Itt zajlik a Product Backlog finomítása.
- **Daily Meeting:** melynek témái:
  - Mit csináltál a tegnapi megbeszélés óta?
  - Mit fogsz csinálni a következő megbeszélésig?
  - Milyen akadályokba ütköztél az adott feladat megoldása során?
- **Sprint Review Meeting:** a sztorik ellenőrzésére és minősítésére.
- **Sprint Retrospective:** a sprint során felmerülő problémákat vitatják meg a tagok.

#### Termékek:

- **Product Backlog:** ebbe helyezi el a Product Owner az elvégzendő **sztorikat** (a sztorik pedig **task**-okra vannak bontva), melyek mindegyikéhez prioritásokat is rendel.
- **Sprint Backlog:** az aktuális sprintre bevállalt munkák, sztorik vannak felsorolva ezen dokumentumban.
- **Burn-down chart:** a munka haladását hivatott illusztrálni egy ábra segítségével (aktuális haladás és elvárt haladás kapcsolata).

**További agilis módszertanok:** XP (Extreme Programming – extrém programozás)

## 15. Tétel: A web működésének alapjai. Web szabványok és szabványügyi szervezetek. URI-k és felépítésük. HTTP: kérések és válaszok felépítése, metódusok, állapotkódok, tartalomgyeztetés, sütik. A web jelölőnyelvei: XML és HTML dokumentumok felépítése. Stíluslap nyelvek. JSON

### Alapfogalmak

**Világháló:** egy információs tér, melynek elemeit (erőforrás) URI (*Uniform Resource Identifier*) azonosítják.

**Tartalomgyeztetés** (content negotiation): egy erőforráshoz több reprezentáció kínálása és ezek közül a legmegfelelőbb kiválasztása, amikor egy reprezentációt kell szolgáltatni.

**Hivatkozás-feloldás** (dereferencing): egy URI használata a hivatkozott erőforrás eléréséhez. Az elérésnek számos formája van mint:

- letöltés, létrehozás, módosítás, törlés

### Ágensek:

1. **web ágens** (*web agent*): a Weben egy személy, entitás vagy folyamat nevében cselekvő személy vagy szoftver. (pl. egy keresőrobot – web crawler)
2. **felhasználói ágens** (*user agent*): a web ágensek egy fajtája, egy személy nevében cselekvő szoftver. (pl. egy webböngésző)

---

**Web szabványok:** olyan dokumentumok, melyek **követelményeket, előírásokat és irányelveket** fogalmaznak meg. Ezek lehetnek:

- **De facto:** a gyakori használatból vagy a piaci elfogadottságból származnak.
  - **például:** QWERTY billentyűzetkiosztás, TeX, PDF (2008 előtt)
- **De jure:** helyi, állami és/vagy nemzetközi szintű szabályozók által kötelezőként előírt szabványok
  - **például:** Nemzetközi Mértékegységrendszer (SI), PDF (2008-tól)

- **Önkéntes közmegegyezékes szabványok:** különböző magánintézmények által meghatározott szabványok
  - **például:** az Internet protokollkészletet (közismert nevén TCP/IP), HTML, CSS

#### Szabványügyi szervezetek:

##### 1. **IANA** (*Internet Assigned Numbers Authority*):

- **feladatai:** IP címek kiosztása, DNS-gyökérzóna felügyelete

##### 2. **IETF** (*Internet Engineering Task Force*):

- internet **szabványokat fejlesztő** nemzetközi szabványügyi szervezet
- ez fejleszti az internet protokollkészletet (TCP/IP)
- nincs formális tagás, így bárki beléphet
- az Internet szabványokhoz kötődő specifikációkat az **RFC dokumentumsorozatban** publikálja

##### 3. **W3C** (*World Wide Web Consortium*):

- **alapelvei:** web mindenkinek és web mindenhol

##### 4. **WHATWG** (*Web Hypertext Application Technology Working Group*):

- böngészőkben implementálható szabványokat fejleszt (pl. HTML, DOM, URL stb.)
- „*élő szabványoknak*” nevezett specifikációkat fejleszt, melyek folyamatosan frissülnek

#### Az RFC dokumentumsorozat:

- az internetről szóló **műszaki és szervezeti dokumentumokat** tartalmaz
- négy folyamatra osztható:

- Internet Engineering Task Force (*IETF*)
- Internet Architecture Board (*IAB*)
- Internet Research Task Force (*IRTF*)
- A független beadványok folyama

---

**URI-k és felépítésük:** absztrakt vagy fizikai erőforrást azonosító tömör karaktersorozat.  
**Szintaxisa, felépítése:**

séma:hierarchikus-rész[?lekérdezés] [#erőforrásrész]

**URI sémák:** file, http/https, mailto, about

**URI-k összehasonlítása:**

- URI-k akkor **ekvivalensek**, ha ugyanazt az erőforrást azonosítják, viszont ez gyakorlati szempontból használhatatlan.
- A gyakorlatban az ekvivalencia megállapítása az URI karakterláncok összehasonlításán alapul.
- Az összehasonlítás során normalizálás is történhet.
  - **például** nagybetű karakterek kisbetű karakterekké alakítása a kisbetű-nagybetű érzéketlen komponensekben

---

**Állapotnélküliség, munkamenet fogalma**

**Munkamenet** (*session*): kérések és válaszok egy kliens és egy szerver közötti sorozata.

**Állapotnélküliség:** az egymást követő kérések egymástól függetlenként kezeltek. (pl. ilyen a HTTP protokoll is)

---

### Kérések:

- egy kérés kezdősora az alábbi felépítésű:

metódus kérés-cél HTTP-verzió CRLF

### Válaszok:

- az üzenet első sora (*status line*) az alábbi felépítésű:

HTTP-verzió állapotkód indok\_fázis CRLF

---

### HTTP metódusok:

- GET: a cél erőforrás egy reprezentációjának átvitelét kéri.  
Információlekérésre használatos.
- HEAD: azonos a GET metódussal, de a szerver nem küldhet üzenettörzset a válaszban.  
Metaadatok megszerzésére használatos.
- POST: erőforrások létrehozására
- PUT: erőforrások módosítására
- DELETE: erőforrások törlésére
- TRACE: a kérés visszaküldését kéri
- OPTIONS: a cél erőforrás kommunikációs opcióiról kéri információkat

### Állapotkódok:

- 1xx: információk (pl. protokollváltás történt)
- 2xx: siker
- 3xx: átirányítás
- 4xx: kliens-oldali hiba

- **5xx:** szerver-oldali hiba

**Tartalomegyeztetés:** amely lehet

1. **proaktív:** előnyös, ha nehéz leírni egy felhasználói ágensnek a rendelkezésre álló reprezentációk közüli választás algoritmusát
2. **reaktív:** előnyös, ha a válasz általánosan használt dimenziók (pl. típus, nyelv, kódolás) mentén változik

**Sütik:**

- **összetevői:** név-érték párok és kapcsolódó metaadatok
  - ezeket egy eredet szerver egy válasz **Set-Cookie** fejlécmezőjében küldi a felhasználói ágensnek
- **felhasználás:** munkamenet kezelés, felhasználói követés stb.
- **attribútumai:**
  - **Expires:** a süti lejáratának dátumát és idejét adja meg
  - **Max-Age:** azt adja meg, hogy hány másodperc múlva jár le a süti
  - további attribútumok: **Domain**, **Secure**, **HttpOnly**
- **perzisztens sütik:** azok a sütik, amik rendelkeznek **Expires** és **Max-Age** attribútumokkal

---

## A web jelölőnyelvei

**HTML** (*HyperText Markup Language*): A web **elsődleges leíró nyelve** – szemantikus<sup>22</sup> leíró nyelv.

Legelterjedtebb verzió a **HTML5**.

**HTML szintaxis:**

- **Kötelező** a dokumentumtípus-deklaráció, a szabályos megjelenéshez. (XML-ben ez nem kötelező)

---

<sup>22</sup>az elemeknek **jelentésük van**

- **Elemek tulajdonságai:**

- az elemeknek, attribútumoknak és attribútumértéknek **meghatározott jelentésük** (szemantikájuk) van
- minden egyes elemnek van egy **tartalommodellje** (az elem szükséges tartalmának egy leírása)
- minden elemet nyitó és záró címke határol
  - \* bizonyos elemek nyitó és záró címkéi elhagyhatóak (pl. a `li` esetében)
- az elem- és attribútumnevek kisbetű-nagybetű érzéketlenek
- **osztályozásuk:** flow, heading, sectioning, metadata stb.
- **speciális elemek:** pl. üres (*void*) elemek (csak nyitó címkéjük van – pl. `<image>`)

- **Attribútumok tulajdonságai:**

- **idézőjel nélküli attribútumérték:** akkor lehetséges, hogyha nem üres és nem tartalmaz egyetlen *whitespace*-t sem
- **speciális attribútumok:** pl. logikai attribútumok

- Nincs HTML DTD.

---

**XML** (*Extensible Markup Language*): általános jelölőnyelv<sup>23</sup>.

**Komponensei:** elemek, névtokenek, literálok, hivatkozások (karakterhivatkozások, egyedhivatkozások), megjegyzések, feldolgozási utasítások, CDATA-szakaszok stb.

**Dokumentumtípus-deklaráció** (DTD): az elemek definiálásához használatos.

**Érvényesség:** egy XML dokumentum érvényes akkor, ha tartozik hozzá dokumentumtípus-deklaráció és a dokumentum eleget tesz a DTD által kifejezett megszorításoknak.

---

<sup>23</sup>szöveg annotálására szolgáló számítógépes nyelvek



## Az XML és HTML összehasonlítása:

- XML

- Nincs előre definiált címkekészlet
- Célja az **adatok leírása**
- **Adatcsere formátumként** használják

- HTML

- Előre definiált címkekészlet van
- Célja az **információ megjelenítés**
- Egy **prezentációs nyelv**
- Tekinthető az **XML egy speciális alkalmazásának** (*XHTML*)

---

**Stíluslap nyelvek – CSS** (*Cascading Style Sheet*): strukturál (HTML, XML) dokumentumok megjelenítésének leírására szolgáló stíluslap nyelv.

**Szintek:** a CSS-nek nincsenek verziói, hanem szintjei vannak:

- **Level 1:** elavultnak tekintett
- **Level 2:** A CSS 2.1 specifikáció definiálja, javítása jelenleg fejlesztés alatt áll
- **Level 3:** jelenleg fejlesztés alatt áll, moduláris felépítésű
- **Nem létezik CSS Level 4**

**Dobozmodell:** tartalom + padding + border + margin

**Szintaktikai elemek:**

- **Vezérlősorozatok:** Unicode karakterek megadásához, általános alakjuk \hhhhh
- **Megjegyzések:** /\* ... \*/ határolók között
- **Deklarációs blokk:** { ... }

- **Szabályhalmazok:** kiválasztók egy sorozata
- **At-szabályok:** a stíluslap feldolgozását vezérlő **speciális szabályok**, pl. @charset

**Tulajdonságok:** a CSS által definiált **paraméterek**, melyek révén a dokumentumok megjelenítése vezérelhető, ezeknek neve és értéke van.

**Kiválasztók:**

- **Típus kiválasztó:** egy CSS minősített név, amely tipikusan egy **azonosító**, pl.  
p { ... }
- **Általános kiválasztó:** \* { ... } alakú, és minden elem illeszkedik rá
- **Attribútum kiválasztók:** pl. [att] vagy [att=érték]
- **Osztály kiválasztó:** pl. class~érték
- **ID-kiválasztó:** #azonosító formájú
- **Pseudo- osztály és elem kiválasztók:** pl. :azonosító és ::before formájúak

**Kombinátorok:** leszármazott- (*whitespace*), gyermek- (">") és szomszéd testvér kombinátor ("+")

**JSON** (*JavaScript Object Notation*): könnyűsúlyú **szöveges nyelvfüggetlen adatcsere** formátum, amely **strukturált adatok** ábrázolására szolgál.

- az **ECMAScript** programozási nyelvből származik
- a JSON az XML alternatívájaként használható adatszeréhez
- a JSON **adatorientált**
- könnyen olvasható és feldolgozható
- **szintaxisa:** kulcs: érték alakú elemek
- **JSON értékek:** null, true és false, string, number, array, object
- **Karakterkódolása:** UTF-8
- **JSON schema:** JSON-alapú sémanyelv JSON dokumentumok érvényesítéséhez

**16. Számítógép-hálózatok osztályozási szempontjai. Hálózati rétegmodellek. IP technológia címzési rendszere, és vezérlése. Forgalomirányítás elve és az útválasztási kategóriák jellemzése. TCP és UDP mechanizmusok.**

**Számítógép-hálózat:** autonóm gépek **összekapcsolt rendszere** közös alkalmazás céljából.

Alkotó elemei: számítógépek, perifériák, hálózati berendezések, fizikai összeköttetést megvalósító eszközök stb.

**Számítógép-hálózatok osztályozási szempontjai:**

**1. Lefedett fizikai terület mérete szerint:**

- BAN (*Body Area Network*)
- PAN (*Personal Area Network*)
- SOHO (*Small Office/Home Office network*)
- LAN (*Local Area Network*)
- MAN (*Metropolitan Area Network*)
- WAN (*Wide-Area Network*)
- GAN (*Generative Adversarial Network*)

**2. Adatátviteli ráta szerint:** klasszikus vagy nagysebességű hálózatok

**3. Tulajdonjog szerint:** magán vagy nyilvános

**4. Mobilitás szerint:** rögzített vagy mobil

---

**Hálózati rétegmodellek:**

**1. ISO/OSI modell (*Open Systems Interconnection Reference Model*):** melynek rétegei

- **Fizikai réteg:** gondoskodik a hálózati *topológiáról és a berendezések globális kapcsolatairól a hálózatra.*

- **Adatkapcsolati réteg:** feladata az *információ megbízható továbbítása* adatátviteli áramkörön keresztül.
- **Hálózati réteg:** *kapcsolatot és útvonalválasztást biztosít* két gazdarendszer között.
- **Szállítási réteg:** feladata a hibamentes *adatok továbbítása* a forrásgépről a célgépre.
- **Viszony réteg:** feladata a *két végrendszer közötti kapcsolat megszervezése*.
- **Megjelenési (prezentációs) réteg:** ügyel az *információk megfelelő megjelenésére* (képek, számok megfelelő megjelenítéséért felel, karakterkódolástól és egyéb tényezőktől függetlenül).
- **Applikációs (alkalmazási) réteg:** lehetőséget biztosít a többi réteg *szolgáltatásaihoz való hozzáférésre*.

2. **TCP/IP modell:** ugyan olyan mint az *ISO/OSI modell*, azzal a különbséggel, hogy **nincs benne viszony- és prezentációs réteg**.

Egyéb rétegmodellek: hibrid modell, IEEE 802 modell.

---

**IP<sup>24</sup> csomag felépítése:**

- **Header:** a *datagram* kézbesítéséhez szükséges információk helye, mely 4B-os szavakból áll.
- **Payload:** szállítási réteg **adatelemét** foglalja magába, melynek mérete maximum 64KB.

**IP technológia címezése rendszere:** két verzióra bontható:

1. **IPv4:**

- Az IPv4 címek 32 bites bináris számok, amelyeket négy decimális számmá alakítanak át.
- A címeket pontokkal elválasztott négy csoportban írják, ahol minden csoport 0 és 255 közötti értéket vehet fel.

---

<sup>24</sup>IP - Internet Protocol

- **két részre bontható:** hálózati rész és hoszt rész
  - **például:** 192.168.0.1
2. **IPv6:** hasonló az IPv4-hez azzal a különbséggel, hogy 128 bites bináris számokból áll és ':' karakterrel vannak elválasztva a csoportok (8db. csoport).
- Előnyük, hogy **nagyobb címtartományt kínálnak.**

### **Maszk:**

- az IP címzésnél használt **kódolási módszer**
- hossza megegyezik az IP cím méretével
- **szabályai:** 0 bit után csakis 0 bit jöhet, az 1 után pedig bármi jöhet
- **például:** 11111111000000000000000000000000, amely decimálisan 255.0.0.0, prefix alakban pedig /8
  - ahol N=8 bit (hálózat azonosító rész) és H=32-8=24 bit (host azonosító rész)

### **ICMP mechanizmus** (*Internet Control Message Protocol*):

- IP-re épülő protokoll, de funkciója miatt a **hálózati réteghez** soroljuk.
- **Célja:** IP datagramok továbbítása során előforduló hibák jelzése, jelzőüzenetek küldésével.
- **Felépítése:** típus kód, ellenőrző összeg, típus specifikus adat
- **Csoportjai és példák:**
  1. **Hibaüzenetek** (*Error Messages*): pl. Destination Unreachable
  2. **Válaszüzenetek** (*Echo Messages*): pl. Echo Request
  3. **Információs üzenetek** (*Informational Messages*): pl. Address Mask Request

**Felhasználási esetek:** útvonal menti MTU<sup>25</sup> meghatározása, útvonal feltérképezése forrás és cél között stb.

---

**Forgalomirányítás elve:** csomagok továbbítási irányának meghatározásával kapcsolatos **döntések meghozatala**. Az ehhez kapcsolódó információkat a **routing table** tartalmazza.

**Problémák a routing táblákkal:**

1. **Túl kicsi kezdőérték probléma:** ha az optimális út megsérül, akkor nagyobb költségű út nem léphet a helyébe.
2. **Végtelenig számolás**

**Útválasztási kategóriák:**

- **Statikus** (*Static Routing*): a hálózati adminisztrátor manuálisan beállítja az útvonalakat a hálózati eszközökön.
- **Dinamikus** (*Dynamic Routing*): a hálózati eszközök (pl. routerek) közötti útvonalakat **automatikusan tanulják meg** és osztják meg egymással.

A hálózati eszközök dinamikus útválasztási protokollokat használnak, amelyek segítségével rendszeresen cserélnek információkat az elérhető útvonalakról és azok állapotáról. Ilyenek például a

- **OSPF** (Open Shortest Path First)
- **EIGRP** (Enhanced Interior Gateway Routing Protocol)
- **RIP** (Routing Information Protocol)
- **Távoli útválasztás** (*Remote Routing*): az útválasztási döntéseket **távoli helyeken** található **központi útválasztók** vagy **vezérlők** hozzák meg.
- **Helyi útválasztás** (*Local Routing*): az útválasztási döntéseket a **helyi hálózati** eszközök hozzák meg.

---

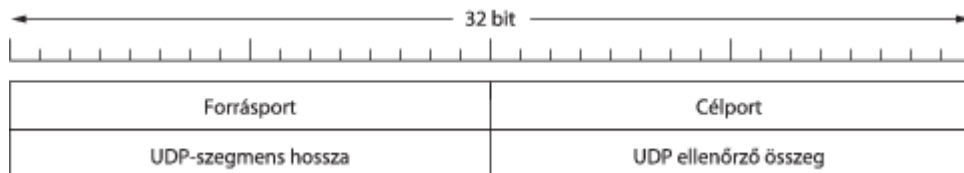
<sup>25</sup>MTU - *Maximum Transmission Unit*

**User Datagram Protocol (UDP): összeköttetés nélküli szállítási réteg** protokoll, amely az internet protokollkészlet része.

- **Előnye:** lehetővé teszi az adatok küldését IP-datagramokban anélkül, hogy előzetes összeköttetést kellene kiépíteni
- **Hátránya:** nem megbízható (mivel nyugta nélküli protokoll – a TCP-vel ellentétben)
- **Felhasználó alkalmazások jellemzői:** adatvesztés tolerancia, átviteli ráta érzékenység.
- **Funkciói:** UDP datagramok küldése és fogadása.
- **Komponensei:**

– **UDP szegmens:** fejrész + rakrész, ahol a

- \* **Fejrész:** forrás- és cél portszám, UDP szegmens hossza, ellenőrző összeg (*checksum*)



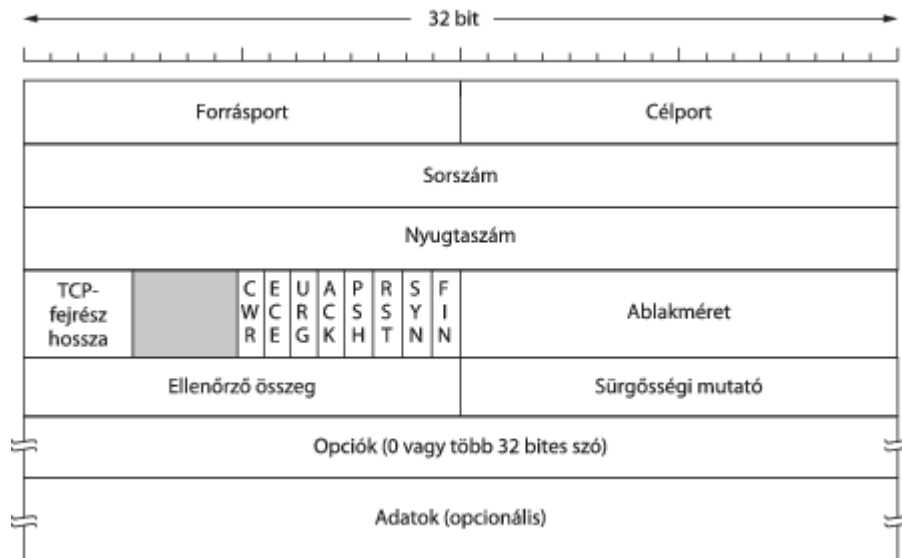
**24. ábra.** UDP szegmens fejrésze

- \* **Rakrész:** tetszőleges tartalom

**Transmission Control Protocol (TCP): összeköttetésen alapuló szállítási réteg** protokoll, amely az internetprotokoll része.

- a szegmensek átvitelére, egy **megbízható és nyugtázott** kapcsolatot biztosít, az összekapcsolt hálózaton keresztül
- dinamikusan alkalmazkodik a hálózat tulajdonságaihoz és az esetleges meghibásodásokhoz

- teljes **duplex kommunikációt** biztosít (vagyis az adatokat mindkét irányban, egyszerre képes továbbítani)
- az adatokat **szegmensek** formájában továbbítja, melyek **szerkezete**:
  1. Forrás- és cél portszám
  2. Szegmens sorszáma
  3. Nyugtaszám
  4. TCP fejrész hossza + kontrol bitek + ablak mérete, ahol
    - **Kontrol bitek**: például
      - \* FIN – adat vége a küldőtől
      - \* RST – a kapcsolat lezárása hiba miatt
      - \* és még sok egyéb mint például: SYN, URG, ECE, ACK, PSH, CWR
    - **Ablakméret**: nyugtázott bájtok száma
  5. Ellenőrző összeg<sup>26</sup> + sűrűsségi mutató
  6. Opciók: 0 vagy több 32 bites szó
  7. Adatok (opcionális)



**25. ábra.** TCP szegmens fejrésze

**Egyéb mechanizmusok:** csúszó ablak mechanizmus és egyebek.

<sup>26</sup>checksum