# ELMÉLETI INFORMATIKA

## II. rész

# Algoritmus- és kiszámíthatóságelmélet

## Alapfogalmak, aszimptotikus jelölések, keresés rendezett halmazban

7. előadás

# Alapfogalmak

A számítástechnika egyik alapvető fogalma az **algoritmus**.

Matematikatörténészek állítása szerint az algoritmus szó *Abu-Jafar Mohammed ibn Mura al-Kvarîzmi* (780 körül – 850 körül) arab matematikus nevének latinos elferdítéséből származik.

Az **algoritmus** fogalma a tantárgy szempontjából igen fontos, mégsem definiáljuk. Inkább olyan intuitív fogalomnak tekintjük, melynek formalizálására különféle lehetőségek vannak.

Az **algoritmus** olyan matematikai eljárást jelent, amely valamely számítás vagy konstrukció elvégzésére (pl. valamely függvény kiszámítására) szolgál, s amelyet gondolkodás nélkül, gépiesen lehet végrehajtani. Ezért az algoritmus fogalma helyett a **matematikai gép** különböző fogalmait vezetjük majd be.

Minden **matematikai gép** valamilyen *bemenetből* valamilyen *kimenetet* számít ki. A bemenet és kimenet lehet pl. egy rögzített ábécé feletti szó, vagy számok egy sorozata. A gép a számításhoz különböző erőforrásokat (pl. *futási idő*, *elfoglalt tárterület*, *kommunikáció*) vesz igénybe, és a számítás bonyolultságát azzal mérjük, hogy az egyes erőforrásokból mennyit használ fel.

Példák számítási modellekre:

- véges automata,
- Turing-gép,
- RAM (**R**andom **A**ccess **M**achine – Közvetlen elérésű gép),
- logikai hálózat.

**ALGORITMUS:** *Egy meghatározott cél elérésére irányuló, egymástól elkülönített, mechanikusan elvégezhető műveletek sorozata, amelyek segítségével bizonyos kiindulási állapotból véges számú közbenső állapoton keresztül egy előírt feltételeknek eleget tevő végállapotba jutunk*.

Az algoritmusokat tehát valamilyen feladat megoldásának céljából tervezzük, majd valamilyen programozási nyelv segítségével kódoljuk, hogy számítógépen végrehajthassuk.

## Az algoritmus helyessége

- Az algoritmust **helyes**nek nevezzük, ha minden konkrét bemenetre helyes kimenetet ad és megáll.
- Az algoritmust **helytelen**nek nevezzük, ha valamilyen konkrét bemenetre nem várt kimenetet ad, megáll egy közbülső lépésnél, vagy egyáltalán nem áll meg.

## Az algoritmus tulajdonságai

- **Elvégezhető** – az ember is képes eljutni az eredményhez, papírral és ceruzával a kezében követve az algoritmusban leírt műveleteket.
- **Diszkrét** – véges sok elemi lépésből áll.
- **Meghatározott** – mindig tudjuk, hogy a következő lépésben mi fog történni.
- **Véges** – az algoritmus véges számú lépésben vezet eredményre.
- **Univerzális** – az algoritmusnak működnie kell tetszőleges, a kezdeti feltételeknek eleget tevő bemeneti érték esetén.

## Algoritmus leírási módok

- **természetes nyelv** – mindenki számára érthető, ám pontatlan,
- **folyamatábra**,
- **pszeudonyelv** – az algoritmus vázát adó elemi utasítások összessége. Nem konkrét programnyelv, de programnyelvszerű jelölésrendszere van,
- **metanyelv** – sajátos szókészlettel, szimbólumokkal és nyelvtannal rendelkező nyelv (pl. BNF – Backus-Naur forma),
- **magas szintű programnyelvek** – Algol, FORTRAN, BASIC, Pascal, C, C++, Java, stb.

**Megjegyzés:** A továbbiakban az algoritmusokat ún. **pszeudokód**ban adjuk meg. Ez a kód közel áll a programkódhoz, ám nem tartalmazza a változók deklarálását, valamint minden olyan hasonló szerkezetet, amelyek az algoritmus megértését nem befolyásolják.

Az algoritmusok elemzése – a helyesség bizonyítása mellett – a végrehajtáshoz szükséges erőforrások mennyiségének meghatározását is jelenti. Gyakran nem tudjuk, vagy nem akarjuk az igényelt erőforrás mennyiségét pontosan megadni. Ilyenkor megelégszünk az igény **nagyságrend**jének megadásával.

Ilyenkor a kiszámított képletnek csak a főtagját vesszük figyelembe, mivel az alacsonyabb rendű tagok nagy $n$ esetén viszonylag jelentéktelenek. Szintén figyelmen kívül hagyjuk a főtag állandó együtthatóját.

A nagyságrend megadásának jól bevált eszközei az $\Omega$, $O$, $\Theta$, $o$ és $\omega$ jelölések.

## 7.1 definíció: (aszimptotikusan éles felső korlát)

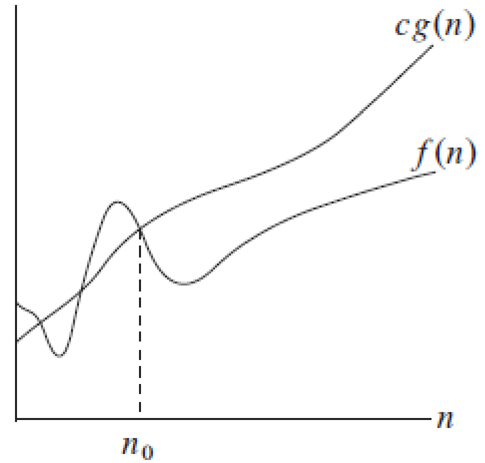
Legyen adott két függvény, $f\colon \mathbb{N} \longrightarrow \mathbb{Z}$, $g\colon \mathbb{N} \longrightarrow \mathbb{Z}$
Ha létezik $c > 0$ konstans, valamint $n_0$ küszöbindex, hogy minden $n \geq n_0$ esetén teljesül az

$$|f(n)| \leq c\,|g(n)|$$

egyenlőtlenség, akkor azt mondjuk, hogy az $f(n)$ függvénynek **aszimptotikusan éles felső korlátja** a $g(n)$ függvény.
Jelölése: $f(n) = O\big(g(n)\big)$

### Megjegyzés:

Ha $f(n) = O\big(g(n)\big)$, akkor a $g(n)$ függvény legalább olyan gyorsan nő, mint az $f(n)$ függvény.



**7.2 definíció:** (aszimptotikusan nem éles felső korlát)

Legyen adott két függvény, $f\colon \mathbb{N} \longrightarrow \mathbb{Z}$, $g\colon \mathbb{N} \longrightarrow \mathbb{Z}$

Ha minden $c > 0$ konstanshoz létezik $n_0$ küszöbindex, hogy minden $n \geq n_0$ esetén teljesül az

$$|f(n)| < c\,|g(n)|$$

egyenlőtlenség, akkor azt mondjuk, hogy az $f(n)$ függvénynek **aszimptotikusan nem éles felső korlátja** a $g(n)$ függvény.

Jelölése: $f(n) = o\big(g(n)\big)$

**Megjegyzés:**

Ha $f(n) = o\big(g(n)\big)$, akkor a $g(n)$ függvény gyorsabban nő, mint az $f(n)$ függvény.

## **7.3 definíció:** (aszimptotikusan éles alsó korlát)

Legyen adott két függvény, $f\colon \mathbb{N} \longrightarrow \mathbb{Z}$, $g\colon \mathbb{N} \longrightarrow \mathbb{Z}$

Ha létezik $c > 0$ konstans, valamint $n_0$ küszöbindex, hogy minden $n \geq n_0$ esetén teljesül az

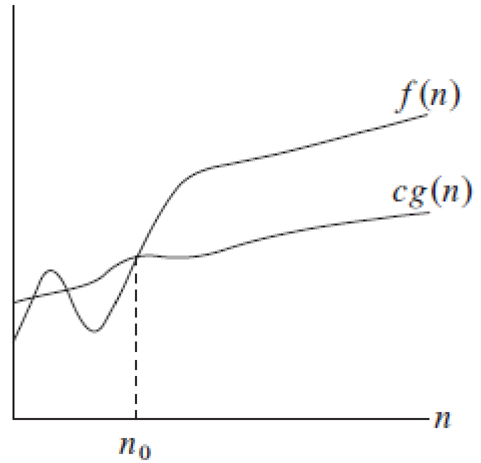
$$c\,|g(n)| \leq |f(n)|$$

egyenlőtlenség, akkor azt mondjuk, hogy az $f(n)$ függvénynek **aszimptotikusan éles alsó korlátja** a $g(n)$ függvény.

Jelölése: $f(n) = \Omega\big(g(n)\big)$

### Megjegyzés:

Ha $f(n) = \Omega\big(g(n)\big)$, akkor a $g(n)$ függvény legalább olyan lassan nő, mint az $f(n)$ függvény.



**7.4 definíció:** (aszimptotikusan nem éles alsó korlát)

Legyen adott két függvény, $f\colon \mathbb{N} \longrightarrow \mathbb{Z}$, $g\colon \mathbb{N} \longrightarrow \mathbb{Z}$
Ha minden $c > 0$ konstanshoz létezik $n_0$ küszöbindex, hogy minden $n \geq n_0$ esetén teljesül az

$$c\,|g(n)| < |f(n)|$$

egyenlőtlenség, akkor azt mondjuk, hogy az $f(n)$ függvénynek **aszimptotikusan nem éles alsó korlátja** a $g(n)$ függvény.
Jelölése: $f(n) = \omega\big(g(n)\big)$

**Megjegyzés:**

Ha $f(n) = \omega\big(g(n)\big)$, akkor a $g(n)$ függvény lassabban nő, mint az $f(n)$ függvény.

## **7.5 definíció:** (aszimptotikusan éles korlát)

Legyen adott két függvény, $f\colon \mathbb{N} \longrightarrow \mathbb{Z}$, $g\colon \mathbb{N} \longrightarrow \mathbb{Z}$

Ha léteznek $c_1 > 0$ és $c_2 > 0$ konstansok, valamint $n_0$ küszöbindex, hogy minden $n \geq n_0$ esetén teljesül az

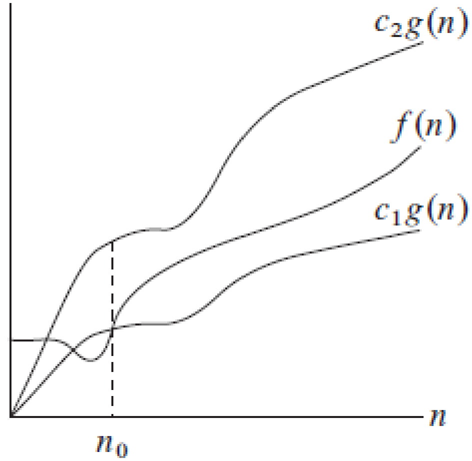
$$c_1\,|g(n)| \leq |f(n)| \leq c_2\,|g(n)|$$

egyenlőtlenség, akkor azt mondjuk, hogy az $f(n)$ függvénynek **aszimptotikusan éles korlátja** a $g(n)$ függvény.

Jelölése: $f(n) = \Theta\big(g(n)\big)$

### **Megjegyzés:**

Ha $f(n) = \Theta\big(g(n)\big)$, akkor a $g(n)$ függvény ugyanolyan gyorsan nő, mint az $f(n)$ függvény.



**7.1 tétel:** Tetszőleges $f\colon \mathbb{N} \longrightarrow \mathbb{Z}$ és $g\colon \mathbb{N} \longrightarrow \mathbb{Z}$ függvényekre érvényes, hogy

$$f(n) = \Theta\big(g(n)\big) \quad \Leftrightarrow \quad f(n) = O\big(g(n)\big) \ \wedge \ f(n) = \Omega\big(g(n)\big)$$

Az $O$, $o$, $\Theta$, $\Omega$, $\omega$ aszimptotikus jelölések a függvények közötti bináris relációként is felfoghatók, így a relációkra vonatkozó ismert definíciók értelmezhetők rájuk.

Ekkor beláthatók a következő állítások:

1) Az $O$, $o$, $\Theta$, $\Omega$, $\omega$ relációk **tranzitív** relációk.
   Pl. tetszőleges $f(n)$, $g(n)$ és $h(n)$ függvényekre:
   ha $f(n) = O\big(g(n)\big) \ \wedge \ g(n) = O\big(h(n)\big)$, akkor $f(n) = O\big(h(n)\big)$

2) Az $O$, $\Theta$, $\Omega$ relációk **reflexív** relációk.
   Pl. tetszőleges $f(n)$ függvényre: $f(n) = O\big(f(n)\big)$

3) A $\Theta$ reláció **szimmetrikus** reláció.
   Pl. tetszőleges $f(n)$ és $g(n)$ függvényekre:
   $f(n) = \Theta\big(g(n)\big)$ akkor és csakis akkor, amikor $g(n) = \Theta\big(f(n)\big)$

**7.1 példa:** Igazoljuk, hogy teljesül: $n^2 - 10n = O(n^2)$

A **7.1 definíció** szerint ha $n^2 - 10n = O(n^2)$, akkor létezik $c > 0$ konstans, valamint $n_0$ küszöbindex, hogy minden $n \geq n_0$ esetén teljesül a következő egyenlőtlenség:

$$|n^2 - 10n| \leq c\,|n^2|$$

A feladatunk találni egy ilyen $c > 0$ konstanst és $n_0$ küszöbindexet.

Ha $n \geq 10$, akkor az egyenlőtlenségben minden abszolút értékben lévő kifejezés nemnegatív értéket fog felvenni, s így az abszolút érték jelölései elhagyhatók. Kapjuk tehát a következő egyenlőtlenséget:

$$n^2 - 10n \leq cn^2$$

$$1 - \frac{10}{n} \leq c$$

Innen már látható, hogy a $c$ konstansnak bármely 1-nél nagyobb szám választható. Legyen pl. $c = 2$, $n_0 = 10$.

**7.2 példa:** Igazoljuk, hogy nem teljesül: $6n^3 - 10n + 3 = O(n^2)$

A **7.1 definíció** szerint ha $6n^3 - 10n + 3 = O(n^2)$, akkor létezik $c > 0$ konstans, valamint $n_0$ küszöbindex, hogy minden $n \geq n_0$ esetén teljesül a következő egyenlőtlenség:

$$|6n^3 - 10n + 3| \leq c\,|n^2|$$

A feladatunk találni egy ilyen $c > 0$ konstanst és $n_0$ küszöbindexet.

Ha $n \geq 2$, akkor az egyenlőtlenségben minden abszolút értékben lévő kifejezés nemnegatív értéket fog felvenni, s így az abszolút érték jelölései elhagyhatók. Kapjuk tehát a következő egyenlőtlenséget:

$$6n^3 - 10n + 3 \leq cn^2$$

$$6n - \frac{10}{n} + \frac{3}{n^2} \leq c$$

Innen már látható, hogy a $c$ konstansnak nem tudunk választani értéket, mert az egyenlőtlenség jobb oldala azt kellően nagy $n$ érték esetén meg fogja haladni. Ezzel beláttuk, hogy $6n^3 - 10n + 3 \neq O(n^2)$.

**7.3 példa:** Igazoljuk, hogy teljesül: $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

A **7.5 definíció** szerint ha $\frac{1}{2}n^2 - 3n = \Theta(n^2)$, akkor léteznek $c_1 > 0$ és $c_2 > 0$ konstansok, valamint $n_0$ küszöbindex, hogy minden $n \geq n_0$ esetén teljesül a következő egyenlőtlenség:

$$c_1\,|n^2| \leq \left|\frac{1}{2}n^2 - 3n\right| \leq c_2\,|n^2|$$

A feladatunk találni egy ilyen $c_1$, $c_2$ konstansokat és $n_0$ küszöbindexet.

Ha $n \geq 7$, akkor az egyenlőtlenségben minden abszolút értékben lévő kifejezés nemnegatív értéket fog felvenni, s így az abszolút érték jelölései elhagyhatók. Kapjuk tehát a következő egyenlőtlenséget:

$$c_1\,n^2 \leq \frac{1}{2}n^2 - 3n \leq c_2\,n^2$$

$$c_1 \leq \frac{1}{2} - \frac{3}{n} \leq c_2$$

$$c_1 \le \frac{1}{2} - \frac{3}{n} \le c_2$$

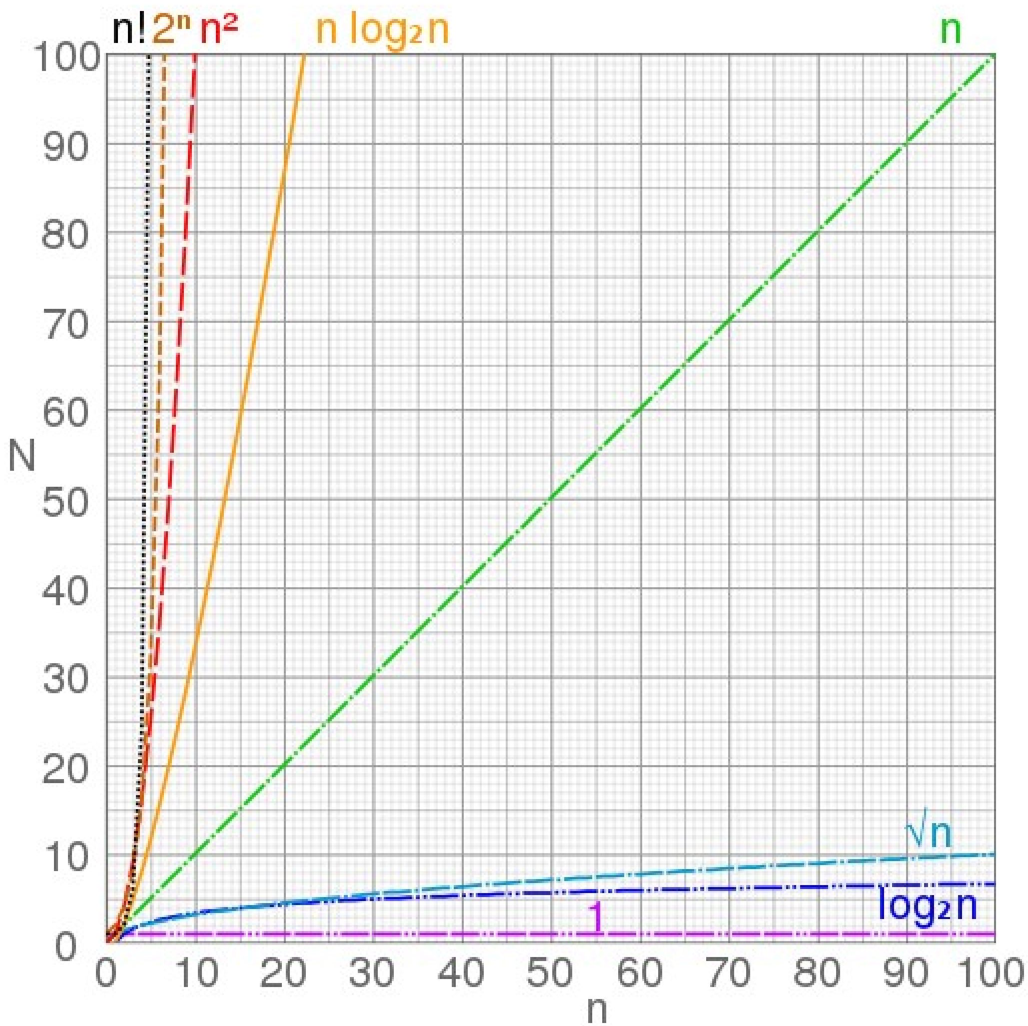
Innen már látható, hogy a $c_1 \le \frac{1}{2} - \frac{3}{n}$ egyenlőtlenség mindig teljesülni fog, ha $c_1 \le \frac{1}{14}$, illetve az $\frac{1}{2} - \frac{3}{n} \le c_2$ egyenlőtlenség mindig teljesülni fog, ha $c_2 \ge \frac{1}{2}$.

Legyen pl. $c_1 = \frac{1}{14}$, $c_2 = \frac{1}{2}$ és $n_0 = 7$.

# Bonyolultsági osztályok

| | | |
|---|---|---|
| $O(1)$ | **konstans** | egyszerű értékadás, írás olvasás veremből |
| $O(n)$ | **lineáris** | lineáris keresés rendezett tömbben,<br>tömb végigolvasása,<br>tömb MIN vagy MAX elemének megkeresése,<br>$n$ elem átlagának meghatározása,<br>$n!$ kiszámítása,<br>Fib($n$) kiszámítása,<br>Counting Sort, |
| $O(n^2)$ | **kvadratikus** | Bubble Sort (buborékrendezés)<br>Insertion Sort (beszúró rendezés)<br>Binary Insertion Sort (bináris beszúró rendezés) |
| $O(\log n)$ | **logaritmikus** | bináris keresés rendezett tömbben<br>elem beszúrása vagy törlése bináris fából |
| $O(n \log n)$ | **linearitmikus** | Quick Sort (gyorsrendezés)<br>Merge Sort (összefésülő rendezés) |
| $O(c^n)$ | **exponenciális** | Hanoi torony probléma<br>$n$ elem összes permutációjának előállítása |

# Bonyolultsági osztályok



# Lineáris keresés

Legyen adott az $S = \{s_1 < s_2 < \cdots < s_{n-1} < s_n\}$ rendezett halmaz és az $x$ elem. Feladatunk annak eldöntése, hogy az $x$ elem benne van-e az $S$ halmazban, s amennyiben igen, akkor az $x$ elem $S$ halmazbeli pozíciója is érdekel minket.

**Lineáris keresést** alkalmazva, első lépésben az $x$ elemet az $S$ halmaz $s_1$ legkisebb elmével hasonlítjuk össze.

Az összehasonlítás eredménye háromféle lehet:

- ha $x = s_1$, akkor a keresés sikeres és a válasz IGEN (1. hely),
- ha $x < s_1$, akkor a keresés sikertelen és a válasz NEM (NIL),
- ha $x > s_1$, akkor az $x$ elemet az $s_2$ elemmel hasonlítjuk össze.

Az eljárást addig folytatjuk, amíg választ nem kapunk a keresőkérdésre, vagy az $S$ halmaz végére nem érünk. Ha $x > s_n$, akkor a válasz NEM (NIL).

# Lineáris keresés (pszeudokód)

**Input:** $S = [s_1, s_2, \ldots, s_n], x$

**Output:** $i$ index, ha $S[i] = x$, vagy NIL, ha $x \notin S$.

---

LINEÁRIS_KERESÉS ($S$,$x$)

**for** $i = 1$ to $length(S)$ **do**
    **if** $x < S[i]$ **then** **return** NIL
    **if** $S[i] = x$ **then** **return** $i$
**return** NIL

**7.4 példa:** Lineáris keresést alkalmazva döntsük el, hogy az $x = 12$ elem benne van-e az $S = [10, 11, 12, 16, 34, 37, 54, 65]$ tömbben!

$x = 12$

| $i$ | 1 | 2 | ↓<br>3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| $S$ | **10** | **11** | **12** | **16** | **34** | **37** | **54** | **65** |

**Output:** 3

# A lineáris keresés bonyolultsága

A **legkedvezőtlenebb eset**ben az $x$ elemet az $S$ tömb minden elemével össze kell hasonlítani, ami $n$ darab összehasonlítást jelent. A módszer bonyolultsága ekkor $O(n)$ nagyságrendű.

**Átlagos eset**ben abból a feltevésből indulunk ki, hogy az $x$ elem egyenlő eséllyel lehet a

$$(-\infty, s_1],\ (s_1, s_2],\ (s_2, s_3],\ \ldots,\ (s_{n-1}, s_n],\ (s_n, \infty)$$

intervallumok bármelyikében.

Ha $x \in (-\infty, s_1]$, akkor a keresés során egy összehasonlítást végzünk (az $x$ elemet az $s_1$ elemmel), és választ kapunk.

Ha $x \in (s_1, s_2]$, akkor a keresés során két összehasonlítást végzünk (az $x$ elemet az $s_1$ és $s_2$ elemekkel), és választ kapunk.

Ha $x \in (s_{n-1}, s_n]$, akkor a kereséskor $n$ összehasonlítást végzünk (az $x$ elemet az $s_1$, $s_2$, …, $s_n$ elemekkel), és választ kapunk.

Ha $x \in (s_n, \infty)$, akkor a kereséskor szintén $n$ összehasonlítást végzünk (az $x$ elemet az $s_1$, $s_2$, …, $s_n$ elemekkel), és választ kapunk.

A keresés $T(n)$ bonyolultságának meghatározásához kiszámítjuk az így kapott $n+1$ darab szám átlagát.

$$T(n) = \frac{1+2+\cdots+n+n}{n+1} = \frac{1+2+\cdots+n}{n+1} + \frac{n}{n+1} =$$

$$= \frac{\frac{n(n+1)}{2}}{n+1} + \frac{n}{n+1} = \frac{n(n+1)}{2(n+1)} + \frac{n}{n+1} = \frac{n}{2} + \frac{n}{n+1}$$

Az $\frac{n}{n+1}$ tört értéke az $n$ növekedésével az 1-hez fog közelíteni. Ezért a lineáris keresés átlagos költsége nagyjából $\frac{n}{2}+1$ összehasonlítás, ami $O(n)$ bonyolultságot jelent.

Selye János Egyetem
GIK, Informatika Tanszék

# Bináris keresés

Legyen adott az $S = \{s_1 < s_2 < \cdots < s_{n-1} < s_n\}$ rendezett halmaz és az $x$ elem. Feladatunk annak eldöntése, hogy az $x$ elem benne van-e az $S$ halmazban, s amennyiben igen, akkor az $x$ elem $S$ halmazbeli pozíciója is érdekel minket.

**Bináris keresés**t alkalmazva, első lépésben az $x$ elemet az $S$ halmaz $s_i$ középső elmével hasonlítjuk össze. Ha az $S$ halmaz elemszáma páros ($n = 2k$), akkor $i = k$; ha páratlan ($n = 2k + 1$), akkor $i = k + 1$.

Az összehasonlítás eredménye háromféle lehet:

- ha $x = s_i$, akkor a keresés sikeres és a válasz IGEN ($i$. hely),
- ha $x < s_i$, akkor az $x$ elemet már csak az $\{s_1, s_2, \ldots, s_{i-1}\}$ részhalmazban kell keresni,
- ha $x > s_i$, akkor az $x$ elemet már csak az $\{s_{i+1}, s_{i+2}, \ldots, s_n\}$ részhalmazban kell keresni.

Ennél a módszernél tehát egyetlen összehasonlítással vagy megtaláljuk az $x$ elem helyét az $S$ rendezett halmazban, vagy pedig visszavezetjük a kérdést egy sokkal kisebb elemszámú $S_1$ rendezett halmazban való keresésre.

Az eljárást addig ismételjük, amíg választ nem kapunk a keresőkérdésre.

# Bináris keresés (pszeudokód)

**Input:** $S = [s_1, s_2, \ldots, s_n]$, $x$

**Output:** $i$ index, ha $S[i] = x$, vagy NIL, ha $x \notin S$.

---

BINÁRIS_KERESÉS ($S$,$x$)

$low \leftarrow 1$<br>
$high \leftarrow length(S)$<br>
**while** $low \leq high$ **do**<br>
    $i \leftarrow (low + high)$ div 2<br>
    **if** $x = S[i]$ **then return** $i$<br>
        **else if** $x > S[i]$ **then** $low \leftarrow i + 1$<br>
                **else** $high \leftarrow i - 1$<br>
**return** NIL

**7.5 példa:** Bináris keresést alkalmazva döntsük el, hogy az $x = 34$ elem benne van-e az $S = [10, 11, 12, 16, 34, 37, 54, 65]$ tömbben!

$x = 34$

| | | | | | ↓ | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | *high* | | | |
| | | | | | *low* | | | |
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $S$ | 10 | 11 | 12 | 16 | **34** | 37 | 54 | 65 |

**Output:** 5

**7.6 példa:** Bináris keresést alkalmazva döntsük el, hogy az $x = 33$ elem benne van-e az $S = [10, 11, 12, 16, 34, 37, 54, 65]$ tömbben!

$x = 33$

| | | | | *high* | *low* | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | ↓ | | | |
| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $S$ | 10 | 11 | 12 | 16 | **34** | 37 | 54 | 65 |

**Output:** NIL

# A bináris keresés bonyolultsága

**Bináris keresés**t alkalmazva az első összehasonlítás után vagy azonnal megtaláljuk az $x$ elem helyét az $S$ rendezett halmazban, vagy pedig visszavezetjük a kérdést egy kisebb elemszámú $S_2$ rendezett halmazban való keresésre. Az így kapott $S_2$ halmaz számosságára érvényes, hogy:

$$|S_2| \leq \frac{|S|}{2} = \frac{n}{2}$$

Az eljárást addig ismételjük, amíg a felezés lehetséges. Az így kapott egyre zsugorodó részhalmazok legyenek: $S = S_1, S_2, \ldots, S_j$.

Az $S_j$ részhalmazról feltételezzük, hogy nem üres halmaz és a középső elemével történő összehasonlítás már megválaszolja az $x \in S?$ keresőkérdést.

A bináris keresés $T(n)$ bonyolultságát ekkor az elvégzett összehasonlítások száma adja meg, azaz $T(n) = j$.

Az $|S_{i+1}| \leq \frac{|S_i|}{2}$ egyenlőtlenségek összefűzésével kapjuk, hogy

$$|S_j| \leq \frac{|S_{j-1}|}{2} \leq \frac{|S_{j-2}|}{2^2} \leq \frac{|S_{j-3}|}{2^3} \leq \cdots \leq \frac{|S_2|}{2^{j-2}} \leq \frac{|S_1|}{2^{j-1}} = \frac{n}{2^{j-1}}$$

Mivel az $S_j$ halmazról feltételeztük, hogy nem üres halmaz, azaz $|S_j| \geq 1$, kapjuk a következő egyenlőtlenséget:

$$1 \leq \frac{n}{2^{j-1}}$$

$$2^{j-1} \leq n$$

$$\log_2 2^{j-1} \leq \log_2 n$$

$$(j-1)\log_2 2 \leq \log_2 n$$

$$j-1 \leq \log_2 n$$

$$j \leq \log_2 n + 1$$

Azt kaptuk tehát, hogy $T(n) \leq \log_2 n + 1$, ami azt jelenti, hogy a bináris keresés bonyolultsága $O(\log n)$.