2 | A nagy ordó jelölés

A félév során általában $L(n)$-nel jelöljük azt a függvényt, ami a vizsgált algoritmusunk lépésszámát írja le. Ez a függvény minden pozitív n érték esetén azt adja meg, hogy hány lépést tesz az algoritmus a „legrosszabb” n-méretű bemeneten.

A lépésszám pontos meghatározása gyakran nehéz feladat. Már annak az eldöntése sem mindig egyszerű, hogy melyik bemenet számít „legrosszabbnak”, és egy összetettebb algoritmus esetén a megtett lépések pontos megszámolása is bonyolult lehet.

Ráadásul a lépésszám meghatározásakor inkább arra vagyunk kíváncsiak, hogy mennyire lassul le az algoritmus, ahogy egyre nagyobb méretű bemeneteken kezdjük el azt futtatni. Ehhez pedig nincs feltétlenül szükségünk arra, hogy pontosan fel tudjuk írni az $L(n)$ függvényt.

Például, ha ábrázoljuk a kiválasztásos rendezés algoritmusának korábban kiszámolt lépésszámát, azaz az $L(n) = 1{,}5n^2 + 1{,}5n - 2$ függvényt, akkor azt látjuk, hogy „nagy” n értékek esetén ez „alig különbözik” a $1{,}5n^2$ függvénytől.

Amikor lépésszámokról beszélünk, akkor szinte mindig elég azt meghatározni, hogy a lépésszám n-nel, $n \log n$-nel, n^2-tel, 2^n-nel arányos-e, és nem törődünk azzal, hogy milyen konstans szorzó szerepel az n-es tag előtt. Ennek az az oka, hogy a lépésszámra adott becsléssel azt szeretnénk leírni, hogy milyen tempóban fog az algoritmus futása lassulni. Ebből a szempontból pedig az n és az $5n$ lépésszámú algoritmusok ugyanúgy viselkednek, míg például az n^2-es és az $3n^2$ lépésszámú algoritmusok máshogy; lásd a 2.1. táblázatot.

Algoritmus sorszáma	Lépésszám tetszőleges n-méretű tömbön	Hány lépést tesz az algoritmus egy		Hányszorosára nőtt a lépésszám?
		k-méretű tömbön?	$(2k)$-méretű tömbön?	
1.	n	k	$2k$	kétszeresére
2.	$5n$	$5k$	$5 \cdot (2k) = 10k$	kétszeresére
3.	n^2	k^2	$(2k)^2 = 4k^2$	négyszeresére
4.	$3n^2$	$3k^2$	$3 \cdot (2k)^2 = 12k^2$	négyszeresére
5.	n^3	k^3	$(2k)^3 = 8k^3$	nyolcszorosára
6.	$4n^3$	$4k^3$	$4 \cdot (2k)^3 = 32k^3$	nyolcszorosára

2.1. táblázat. Milyen tempóban lassul egy n, egy $5n$, egy n^2, egy $3n^2$, egy n^3 és egy $4n^3$ lépésszámú algoritmus?

Ezek a meggondolások motiválják az alábbi definíciót.

2.1. definíció.
Azt mondjuk, hogy egy algoritmus $L(n)$ lépésszáma $O\big(f(n)\big)$ (ejtsd: nagy ordó $f(n)$), ha van olyan c pozitív konstans és n_0 pozitív küszöbérték, hogy minden $n \geq n_0$ esetén

$$L(n) \leq c \cdot f(n)$$

teljesül.

Érdemes megfigyelni, hogy a 2.1. definíció a korábbi szempontok mindegyikét magában foglalja: „nagy" méretű bemenetek esetén ad az $L(n)$ lépésszámra egy olyan felső becslést, ami nem foglalkozik a konstans szorzóval.

2.2. példa.
Megmutatjuk, hogy a kiválasztásos rendezést megvalósító algoritmus lépésszáma $O(n^2)$. Nyilván

$$L(n) = 1{,}5n^2 + 1{,}5n - 2 \leq 1{,}5n^2 + 1{,}5n.$$

Mivel ha $n \geq 1$, akkor $n \leq n^2$, ezért

$$L(n) \leq 1{,}5n^2 + 1{,}5n \leq 1{,}5n^2 + 1{,}5n^2 = 3n^2.$$

Vagyis a 2.1. definícióban a $c = 3$ és az $n_0 = 1$ értékeket választva látjuk, hogy $L(n)$ valóban $O(n^2)$.

Nyilván ekkor a kiválasztásos rendezés algoritmusának a lépésszáma $O(n^3)$ is, hiszen ha $n \geq 1$, akkor

$$L(n) \leq 3n^2 \leq 3n^3.$$

Vagyis a 2.1. definícióban a $c = 3$ és az $n_0 = 1$ értékeket választva látjuk, hogy $L(n)$ valóban $O(n^2)$.

Megjegyezzük azt is, hogy a nagy ordó definíciójában nincs valamiféle „legjobb" c és n_0 választás. Például úgy is megmutathattuk volna, hogy a kiválasztásos rendezés lépésszáma $O(n^3)$-ös, ha a $c = 1$ és $n = 3$ értékeket választjuk: ha $n \geq 3$, akkor

$$L(n) \leq 3n^2 \leq n \cdot n^2 = n^3.$$

Ellenben a kiválasztásos rendezés algoritmusának a lépésszáma nem lesz $O(n)$; de ezzel nem foglalkozunk részletesebben ennek a tárgynak a keretein belül.

A nagy ordó tulajdonságai.

Ha egy $O\big(f(n)\big)$-es és egy $O\big(g(n)\big)$-es függvényt összeszorzunk, akkor az eredmény $O\big(f(n)\cdot g(n)\big)$-es lesz. Avagy tömörebben leírva $O\big(f(n)\big)\cdot O\big(g(n)\big) = O\big(f(n)\cdot g(n)\big)$.

Ha egy $O\big(f(n)\big)$-es és egy $O\big(g(n)\big)$-es függvényt összeadunk, akkor az eredmény $O\Big(\max\big(f(n), g(n)\big)\Big)$-es lesz. Avagy tömörebben leírva $O\big(f(n)\big) + O\big(g(n)\big) = O\Big(\max\big(f(n), g(n)\big)\Big)$.

2.3. példa.
Az előző, összeadásra vonatkozó szabály csak konstans sok függvény összeadása esetén működik. Ha az összeadandók darabszáma függ n-től, akkor már nem feltétlenül lesz igaz, hogy a legnagyobb nagyságrendű összeadandó fogja meghatározni az összeg nagyságrendjét. Ahogy már korábban láttuk,

$$(n-1) + (n-2) + (n-3) + \ldots + 1 = \frac{n(n-1)}{2},$$

ami $O(n^2)$-es, de nem $O(n)$-es, holott a legnagyobb nagyságrendű (sőt, mindegyik) összeadandó $O(n)$-es.

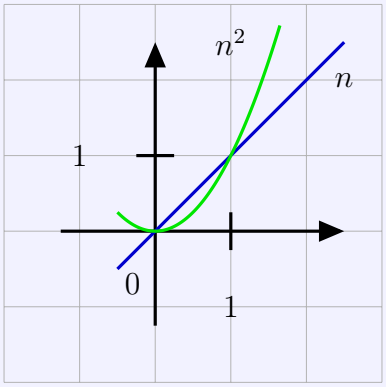
Néhány nevezetes lépésszám nagy ordó szerinti sorrendje.

Az alábbi lépésszámok a hozzájuk tartozó algoritmusok gyorsasága szerint vannak felsorolva, a gyorsabbaktól a lassabbakig.

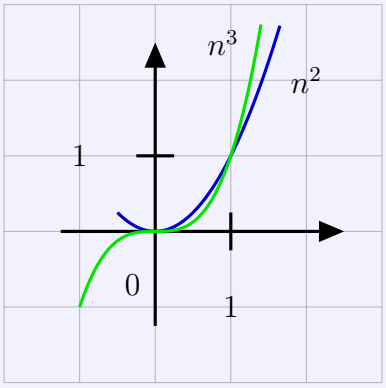
$$1, \quad \log_2 n, \quad \sqrt{n}, \quad n, \quad n^2, \quad n^3, \quad \ldots, \quad 2^n, \quad 3^n, \quad \ldots, \quad n!, \quad n^n$$

2.4. példa (Küszöbindexek és konstans-szorzók meghatározása gyakran használt becslésekhez)**.**

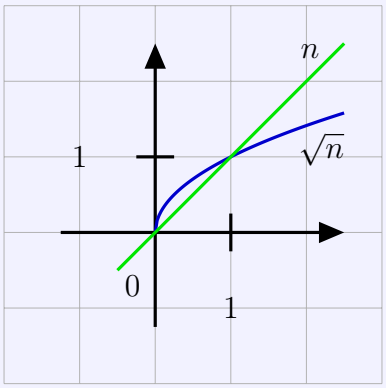
- Ha $n \geq 1$, akkor $n \leq n^2$, azaz a definícióban a $c = 1$ és $n_0 = 1$ értékeket választva látjuk, hogy n valóban $O(n^2)$-es.



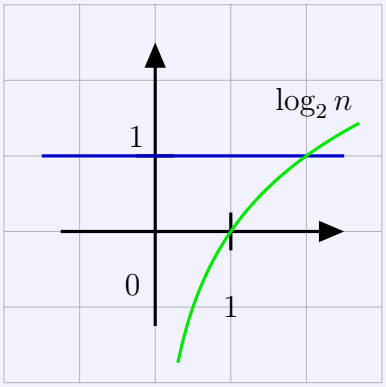
- Ha $n \geq 1$, akkor $n^2 \leq n^3$, azaz a definícióban a $c = 1$ és $n_0 = 1$ értékeket választva látjuk, hogy n^2 valóban $O(n^3)$-ös.



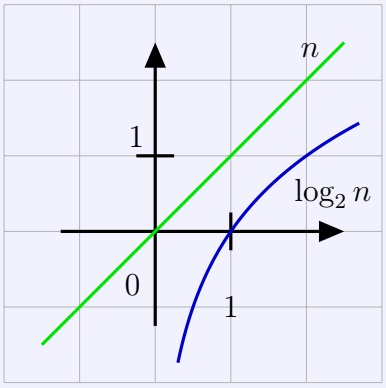
– Ha $n \geq 1$, akkor $\sqrt{n} \leq n$, azaz a definícióban a $c = 1$ és $n_0 = 1$ értékeket választva látjuk, hogy $\sqrt{n}$ valóban $O(n)$-es.



– Ha $n \geq 2$, akkor $1 \leq \log_2 n$, azaz a definícióban a $c = 1$ és $n_0 = 2$ értékeket választva látjuk, hogy 1 valóban $O(\log_2 n)$-es.



– Ha $n > 0$, akkor $\log_2 n \leq n$, azaz a definícióban a $c = 1$ és például az $n_0 = 1$ értékeket választva látjuk, hogy $\log_2 n$ valóban $O(n)$-es.



2.5. feladat.
Határozzuk meg, hogy hol helyezkedik el a fenti sorrendben egy $n \log_2 n$ lépésszámú algoritmus.

Megjegyezzük, hogy nagy ordóban a logaritmus alapját gyakran elhagyjuk, ugyanis a középiskolában tanult logaritmus-azonosságok miatt a különböző (konstans) alapú logaritmusok mindössze egy konstans-szorzóban térnek el egymástól. Például

$$\log_{10} n = \frac{\log_2 n}{\log_2 10} \approx 0{,}3010 \cdot \log_2 n.$$

A kiválasztásos rendezés lépésszámának becslése a nagy ordó segítségével.

A nagy ordó definíciójának ismeretében a kiválasztásos rendezést megvalósító algoritmus lépésszámát egyszerűbben is meg tudjuk becsülni. A külső ciklus magja $O(n)$-szer fut le (egészen pontosan $n-1$-szer fut le, és $n-1 \leq n$). A külső ciklus magjában minden alkalommal $O(1)$ lépést teszünk (nevezetesen 2 darab értékadást és 1 darab cserét hajtunk végre) és $O(n)$-szer lefuttatjuk a belső ciklus magját (egészen pontosan a $(n-1-j)$-szer fut ez le, és $n-1-j \leq n$). A belső ciklus magjában minden alkalommal $O(1)$ lépést teszünk (nevezetesen 1 darab összehasonlítást és legfeljebb 2 darab értékadást végzünk el). A végén még van $O(1)$ lépés (nevezetesen az eredmény-visszaadás). Ez összesen

$$O(n) \cdot \big(O(1) + O(n) \cdot O(1)\big) + O(1) = O(n^2)$$

lépés.