Adatstruktúrák és Algoritmusok

2. gyakorlat

Ordó szimbolika, növekedési rendek, algoritmus időbonyolultsága

A tanult fogalmak átismétlése

Fogalmak

- Az **absztrakt adat** valamilyen halmaznak az eleme.
- Az **absztrakt adattípus** egy leírás, amely megadja az absztrakt adatok halmazát és a rajtuk elvégezhető műveleteket, nem törődve azok konkrét (gépi) realizálásával.
- **Adatstruktúrá**nak nevezzük az absztrakt adattípus konkrét megjelenési formáját.
- Az **algoritmus** egy meghatározott számítási eljárás a számítási probléma megoldási eszköze.
- A **pszeudokód** az algoritmus lejegyzésének egy módja.

A WHILE DO ciklus

- *egy előltesztelő ciklus,*
- *a ciklusblokk akkor lesz elvégezve, ha a feltétel igaz, és hamisnál lép ki a ciklusból.*

A REPEAT UNTIL ciklus

- *egy hátultesztelő ciklus,*
- *a ciklusblokk mindaddig elvégzésre kerül, amíg a feltétel hamis, azaz akkor enged ki a ciklusból, ha a feltétel igazzá válik.*

Példa

Írjunk algoritmust, amely egy

$$p(x) = p_n x^n + p_{n-1} x^{n-1} + \cdots + p_1 x + p_0$$

n-ed fokú polinom helyettesítési értékét kiszámolja valamilyen $\alpha \in \mathbb{R}$ helyen.
Itt most egy kicsit nagyvonalúskodunk a fokszám fogalmával, ugyanis ha egy polinom konstans, de nem azonosan 0 (azaz $p(x) = p_0$, de $p_0 \neq 0$), akkor azt mondjuk, hogy a polinom fokszáma 0, míg az azonosan 0 polinomra (azaz $p(x) = p_0$ és $p_0 = 0$) esetén azt mondjuk, hogy a fokszáma $-\infty$.
Mi most nem fogunk különbséget tenni, a konstans, de nem azonosan 0, illetve az azonosan 0 polinomok között, mindkétfélére azt mondjuk, hogy a fokszámuk 0.

A másik eltérés a későbbiekhez képest az, hogy kényelmi okokból, kizárólag ebben a példában egy tömb indexelését a 0-tól kezdjük, azonban a későbbiekben az indexelést 1-től fogjuk kezdeni.
Ha P egy tömb, akkor a $\mathrm{Hossz[P]}$ a P tömbben tárolt elemek számát fogja jelölni, azaz ha a fenti polinom együtthatóit tároljuk a P tömbben, akkor

$$P[n] = p_n, \quad P[n-1] = p_{n-1}, \quad \ldots, \quad P[0] = p_0.$$

Könnyű látni, hogyha a $p(x)$ egy n-ed fokú polinom, akkor $\mathrm{Hossz[P]} = n + 1$.

Megoldás

Ismertetjük az úgynevezett **Horner elrendezést**:
Ha a $p(x)$ egy harmadfokú polinom. akkor a helyettesítési értékét egy α helyen az alábbi módon számolhatjuk ki:

	p_3	p_2	p_1	p_0
0	p_3	$p_3\alpha + p_2$	$p_3\alpha^2 + p_2\alpha + p_1$	$p_3\alpha^3 + p_2\alpha^2 + p_1\alpha + p_0$

Motiváció az algoritmushoz:

$$\begin{aligned} p(\alpha) &= p_3\alpha^3 + p_2\alpha^2 + p_1\alpha + p_0 = \\ &= (p_3\alpha^2 + p_2\alpha + p_1)\alpha + p_0 = \\ &= ((p_3\alpha + p_2)\alpha + p_1)\alpha + p_0 = \\ &= (((0 \cdot \alpha + p_3)\alpha + p_2)\alpha + p_1)\alpha + p_0 \end{aligned}$$

Az algoritmus

	$\mathrm{Horner}(P, \alpha, y)$
1.	$y \leftarrow 0$,
2.	FOR $i \leftarrow (\mathrm{Hossz}[P] - 1)$ DOWNTO 0 DO
3.	$\quad y \leftarrow y\alpha + P[i]$,
4.	RETURN(y).

Az algoritmus műveletigénye: $2n + 2$., tehát egy jó kis lineáris idejű algoritmust kaptunk.

Oszd meg és uralkodj elv

Ez a stratégia a kiinduló problémát kisebb méretű független részproblémákra bontja, amelyeket rekurzívan old meg. A kisebb méretű részproblémák megoldásait egyesítve kapja meg az eredeti probléma megoldását. (Pl. Euklideszi algoritmus.)

Idő és tárbonyolultság, növekedési rendek

Jelölések:

Legyen A egy algoritmus. Jelölje

- *D az **algoritmus inputjainak a halmazát**;*
- *$|x|$ az $x \in D$ **input méretét** az, ami az x bitjeinek a száma;*
- *$t_A(x)$ az x **input időigényét** (Pl. a műveletek számát);*
- *$s_A(x)$ az x **input tárigényét**.*

Az algoritmus idő és tárbonyolultsága:

Legyen A egy algoritmus.

- ***Az algoritmus időbonyolultsága:***

$$T_A(n) = \sup\{t_A(x) \mid x \in D, |x| \leqslant n\} \quad (n \in \mathbb{Z}_+).$$

- ***Az algoritmus tárbonyolultsága:***

$$S_A(n) = \sup\{s_A(x) \mid x \in D, |x| \leqslant n\} \quad (n \in \mathbb{Z}_+).$$

A továbbiakban az időbonyolultságot vizsgáljuk, a tárbonyolultságot nem.

Növekedési függvények

A T_A, $S_A : \mathbb{Z}_+ \to \mathbb{Z}_+$ függvények pozitív egész értékű monoton növekvő függvények. Az ilyen függvényeket ***növekedési függvény****eknek nevezzük.*

Példa

Van egy gép, amely 1 s alatt 10^6 *műveletet végez. Könnyű kiszámolni, hogy adott időegység alatt hány műveletet végez el a gép.*

idő	1*mp*	1 *perc*	1 *óra*	1 *nap*	1 *év*	100 *év*
műv. sz.	10^6	$6 \cdot 10^7$	$3.6 \cdot 10^9$	$8.6 \cdot 10^{10}$	$3.1 \cdot 10^{13}$	$3.1 \cdot 10^{15}$

Példa

$T(n)$ ismeretében adott időegység alatt mekkora inputmérettel végez a gép?
Például: $T(n) = n^2$ és időegység 1 nap.
Mo.: $n^2 = T(n) = 8.64 \cdot 10^{10}$, így kapjuk, hogy
$n = \sqrt{8.64 \cdot 10^{10}} = 2.9394 \cdot 10^5$, tehát 1 nap alatt $2.9394 \cdot 10^5$ inputmérettel végez a gép.
Foglaljuk táblázatba, hogy 1 másodperc, 1 perc, 1 óra, 1 nap, 1 év, 100 év alatt mekkora inputmérettel végez a gép.

műv. sz.	10^6	$6 \cdot 10^7$	$3.6 \cdot 10^9$	$8.6 \cdot 10^{10}$	$3.1 \cdot 10^{13}$	$3.1 \cdot 10^{15}$
$T(n)$	1mp	1 perc	1 óra	1 nap	1 év	100 év
$\log(n)$	$10^{3 \cdot 10^5}$	-	-	-	-	-
$\sqrt{n}$	10^{12}	$3.6 \cdot 10^{15}$	$1.2 \cdot 10^{19}$	$7.4 \cdot 10^{21}$	$9.9 \cdot 10^{26}$	$9.9 \cdot 10^{30}$
n	10^6	$6 \cdot 10^7$	$3.6 \cdot 10^9$	$8.6 \cdot 10^{10}$	$3.1 \cdot 10^{13}$	$3.1 \cdot 10^{15}$
$n \log(n)$	62746	$2.8 \cdot 10^6$	$1.3 \cdot 10^8$	$2.7 \cdot 10^9$	$7.9 \cdot 10^{11}$	$6.8 \cdot 10^{13}$
n^2	1000	7746	60000	$2.9 \cdot 10^5$	$5.6 \cdot 10^6$	$5.6 \cdot 10^7$
n^3	100	391	1532	4420	31603	$1.4 \cdot 10^5$
2^n	19	25	31	36	44	51
$n!$	9	11	12	13	16	18

Növekedési rendek, ordó szimbolika

Növekedési rendek, ordó szimbolika

- Legyenek $f(n)$, $g(n)$ növekedési függvények. Például: $f(n)$ egy vizsgált algoritmus időigénye, $g(n)$ valamilyen tipikus növekedési függvény. Szeretnénk megmutatni, hogy az $f(n)$ és $g(n)$ függvények közül melyik nő gyorsabban.
- Az általunk vizsgált ordó szimbolika a futási időket aszimptotikusan (tehát határértékben) vizsgálja, ami talán úgy érthető meg első közelítésben, hogy a kapott eredmények "nagy n-ekre" igazak. Kis n-ekre a futási időbe még nagyon bele tud szólni az n_0 küszöb, ami alatt nem tudjuk, hogy hogyan viselkedik az algoritmus, illetve a c_1 és c_2 konstansok.
- A pszeudokóddal leírt algoritmusok esetén könnyű futási időt számolni, hiszen minden lépéshez odaírjuk az elvégzendő műveletek számát, majd a kapott számokat összeadjuk.

- Rekurzív módon írt algoritmus esetén $T(n)$-re rekurzív összefüggést kapunk. Ha elég ügyesek vagyunk, akkor fel tudjuk oldani a rekurziót, azonban van egy matematikai módszer, a **Mester tétel**, amelynek alkalmazásával mechanikusan számolható, hogy egy adott rekurzív módon definiált növekedési függvény melyik függvényosztályba tartozik. A Mester tétellel és alkalmazásaival a következő gyakorlaton ismerkedünk meg.
- A matematikai definíciók során használni fogunk bizonyos matematikai jeleket (kvantorok), melyek a következők:

$$\forall = \text{ minden}, \qquad \exists = \text{ létezik}.$$

- A következő jelöléseket használjuk:

$$f(n) = \mathcal{O}(g(n)), \qquad f(n) = \Omega(g(n)), \qquad f(n) = \Theta(g(n)).$$
$$f(n) = o(g(n)), \qquad f(n) = \omega(g(n)),$$

- Ezeket az alábbi módon olvassuk ki "$\mathcal{O}$ =nagy ordó", "Ω =nagy omega", "o =kis ordó", "ω=kis omega".
- Az egyenlőségjel használata egy kicsit suta, mivel rögzített $g(n)$ növekedési függvény esetén a $\mathcal{O}(g(n))$, $\Omega(g(n))$, $o(g(n))$, $\omega(g(n))$, $\mathcal{O}(g(n))$, $\Theta(g(n))$ nem függvényeket, hanem függvényosztályokat jelöl. Sajnos még azt sem mondhatjuk, hogy $f(n)$ abba a függvényosztályba tartozik, amelybe $g(n)$ ugyanis

$$g(n) \notin o(g(n)), \qquad \text{és} \qquad g(n) \notin \omega(g(n))$$

Formális definíciók:

$$\boxed{f(n) = \mathcal{O}(g(n))}$$

$$\exists(c > 0)\exists(n_0 \in \mathbb{Z}_+)\forall(n \geqslant n_0) : f(n) \leqslant cg(n)$$

("$f(n)$ lassabban nő, mint a $g(n)$".)

$$\boxed{f(n) = o(g(n))}$$

$$\forall(c > 0)\exists(n_0 \in \mathbb{Z}_+)\forall(n \geqslant n_0) : f(n) \leqslant cg(n)$$

("$f(n)$ szig. lassabban nő, mint a $g(n)$".)

$$\boxed{f(n) = \Omega(g(n))}$$

$$\exists(c > 0)\exists(n_0 \in \mathbb{Z}_+)\forall(n \geqslant n_0) : cg(n) \leqslant f(n)$$

("$f(n)$ gyorsabban nő, mint a $g(n)$".)

$$\boxed{f(n) = \omega(g(n))}$$

$$\forall(c > 0)\exists(n_0 \in \mathbb{Z}_+)\forall(n \geqslant n_0) : cg(n) \leqslant f(n)$$

("$f(n)$ szig. gyorsabban nő, mint a $g(n)$".)

$f(n) = \Theta(g(n))$ akkor és csak akkor, ha

$$\exists(c_1 > 0)\exists(c_2 > 0)\exists(n_0 \in \mathbb{Z}_+)\forall(n \geqslant n_0): \\ f(n) \leqslant c_1 g(n) \text{ és } c_2 g(n) \leqslant f(n).$$

Most megfogalmazzuk a fenti definíciók átfogalmazásait az analízis nyelvén, ehhez azonban szükségük lesz a légyegében teljesülő tulajdonság fogalmára.
Egy sorozatokra vonatkozó **tulajdonság lényegében teljesül**, ha valamely küszöbindextől teljesül.

Átfogalmazás

Legyenek $f(n)$ és $g(n)$ növekedési függvények. Az $\frac{f(n)}{g(n)}$ hányados viselkedését kell figyelni.

1. $f(n) = \mathcal{O}(g(n))$ *akkor és csak akkor, ha* $\frac{f(n)}{g(n)}$ *sorozat lényegében felülről korlátos pozitív felső korláttal.*
2. $f(n) = \Omega(g(n))$ *akkor és csak akkor, ha az* $\frac{f(n)}{g(n)}$ *sorozat lényegében alulról korlátos pozitív alsó korláttal.*
3. $f(n) = \Theta(g(n))$ *akkor és csak akkor, ha az* $\frac{f(n)}{g(n)}$ *sorozat lényegében korlátos pozitív alsó és felső korláttal.*
4. $f(n) = o(g(n))$ *akkor és csak akkor, ha* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$.
5. $f(n) = \omega(g(n))$ *akkor és csak akkor, ha* $\lim_{n\to\infty} \frac{f(n)}{g(n)} = +\infty$.

A leírtakból látszik, hogy a Θ-nak miért nincs kistestvére.

Elégséges feltételek

1. *Ha az* $f(n) = o(g(n))$, *akkor* $f(n) = \mathcal{O}(g(n))$.
2. $f(n) = \omega(g(n))$, *akkor* $f(n) = \Omega(g(n))$.

Talán érdemes megjegyezni, hogyha $f(n) = \Theta(g(n))$ és α, β pozitív valós számok, akkor $\alpha f(n) + \beta = \Theta(g(n))$. Analóg összefüggések fogalmazhatók meg a többi függvényosztállyal kapcsolatban is.

Növekedési rend

Könnyű látni, hogy az $f(n) = \Theta(g(n))$ egy ekvivalenciarelációt határoz meg a növekedési függvények halmazán. A kapott ekvivalenciareláció hatására a növekedési függvények halmaza ekvivalenciaosztályokra esik szét. A kapott ekvivalencia osztályokat **növekedési rend**eknek nevezzük.

Nevezetes növekedési rendek:

Konstans	$f(n) = \Theta(1)$
Lineáris	$f(n) = \Theta(n)$
Kvázilineáris	$f(n) = \Theta(n \log(n))$
Négyzetes	$f(n) = \Theta(n^2)$
Köbös	$f(n) = \Theta(n^3)$
Polinomiális	$f(n) = \Theta(n^k)$ *valamely* $k \in \mathbb{R}_+$ *esetén*
Logaritmikus	$f(n) = \Theta(\log(n))$
Exponenciális	$f(n) = \Theta(a^n)$ *valamely* $a \in \mathbb{R}_+$ $a > 1$ *esetén*

Feladatok

Igazoljuk, hogy

1. $\frac{n^2}{2} - 3n = \Theta(n^2)$,
2. $3 \cdot 2^{n-1} = \Theta(2^n)$,
3. $\log(3n^2 + 2n) = \Theta(\log(n))$.

Megoldás

1. $(\frac{n^2}{2} - 3n = \Theta(n^2))$
Könnyű látni, hogy

$$\frac{f(n)}{g(n)} = \frac{\frac{n^2}{2} - 3n}{n^2} \to \frac{1}{2} > 0.$$

Az alkalmas c_1 és c_2 konstansok konstansokat úgy keressük meg, hogy a határértéktől kicsit megyünk jobbra, kicsit megyünk balra így kapjuk, hogy a $c_1 = 0.4$, $c_2 = 0.6$ választás megfelelő lesz.

A küszöbindexek megkereséséhez egyszerű egyenlőtlenségeket kell megoldanunk.

$$0.4 \leqslant \frac{\frac{n^2}{2} - 3n}{n^2} \iff 0.4n^2 \leqslant \frac{n^2}{2} - 3n \iff$$
$$\iff 3n \leqslant 0.1n^2 \iff 3 \leqslant 0.1n \iff 30 \leqslant n,$$

illetve

$$\frac{\frac{n^2}{2} - 3n}{n^2} \leqslant 0.6 \iff \frac{n^2}{2} - 3n \leqslant 0.6n^2 \iff$$
$$\iff -3n \leqslant 0.1n^2 \iff -3 \leqslant 0.1n,$$

az utolsó egyenlőtlenség azonban minden $n \in \mathbb{Z}_+$ esetén teljesük. Így a $c_1 = 0.4$, $c_2 = 0.6$, $n_0 = 30$ választás megfelelő.

2.$(3 \cdot 2^{n-1} = \Theta(2^n))$
Könnyű látni, hogy

$$\frac{f(n)}{g(n)} = \frac{3 \cdot 2^{n-1}}{2^n} = \frac{3}{2} \to \frac{3}{2} > 0.$$

Így $3 \cdot 2^{n-1} = \Theta(2^n)$. A konstansok megkeresését nem részletezzük.

3. $(\log(3n^2 + 2n) = \Theta(\log(n)))$
Heurisztikus megoldás: $\log(3n^2) = \log(3) + 2\log(n) = \Theta(\log(n))$.
Illetve számolhatunk a L'Hospital szabály segítségével is:

$$\lim_{x\to\infty} \frac{\log(3x^3 + 2x)}{\log(x)} = \lim_{x\to\infty} \frac{\frac{1}{\ln(2)}\ln(3x^3 + 2x)}{\frac{1}{\ln(2)}\ln(x)} =$$

$$= \lim_{x\to\infty} \frac{\frac{1}{3x^2+2x}\cdot(6x+2)}{\frac{1}{x}} = \lim_{x\to\infty} \frac{6x^2 + 2x}{3x^2 + 2x} = 2 > 0,$$

azaz teljesül az állítás.

További feladatok

Igazoljuk, hogy

4. $6n^3 + 2n = \Omega(n^2)$,
5. $2n^2 + 3n = o(n^3)$,
6. $2^{n+1} = \Omega(2^n)$,
7. $2^n = o(3^n)$,
8. $n^k = o(2^n)$,
9. $2^n = o(n!)$.

A bizonyítások triviálisak az elemi analízis megfelelő állításai alapján, azonban a feladatok tanulságosak.

KÖSZÖNÖM A FIGYELMET!