1 namų darbas (2 užd.)

Uždavinys 1 (0.2 balo). (a) Pasinaudodami aritmetinės progresijos $a_k, a_{k+1}, \dots, a_l$ (kur $a_i = a_k + (i-k)\cdot d$, $i = k+1, \dots, l$) sumos formule

$$\sum_{j=k}^{l} a_j = \frac{a_k + a_l}{2}(l-k+1),$$

geometrinės progresijos $b_1, b_2, b_3, \dots, b_k$ (kur $b_i = b_1 \cdot q^{i-1}$, $i = 2, \dots, k$, ir $q \neq 1$) sumos formule

$$\sum_{i=1}^{k} b_i = b_1 \frac{1-q^k}{1-q}$$

bei nesunkiai matematine indukcija įrodoma formule

$$1^2 + 2^2 + 3^2 + \cdots + n^2 = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

apskaičiuokite baigtinę sumą $f(n) = \sum_{k=u(n)}^{v(n)} g(k)$.

(b) Raskite $f(n)$ asimptotiką, t.y. konstantas a ir b tokias, kad $f(n) \sim an^b$, kai $n \to \infty$. Jei $f(n)$ auga eksponentiškai, tada raskite konstantas a ir b tokias, kad $f(n) \sim ab^n$.

Nurodymas. $f(n) \sim g(n)$ ("f yra asimptotiškai lygi g"), jei

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1.$$

Pavyzdys 1. Reikia rasti $f(n) = 1^2 + 3^2 + 5^2 + \cdots + n^2$, kur $n = 2k+1$ yra nelyginis skaičius. Turime:

$$\begin{aligned}
f(n) &= \sum_{k=0}^{(n-1)/2} (2k+1)^2 = \sum_{k=0}^{(n-1)/2} (4k^2+4k+1) = 4\sum_{k=0}^{(n-1)/2} k^2 + 4\sum_{k=0}^{(n-1)/2} k + \sum_{k=0}^{(n-1)/2} 1 \\
&= 4\frac{\frac{n-1}{2}(\frac{n-1}{2}+1)n}{6} + 4\frac{n-1}{4}\left(\frac{n-1}{2}+1\right) + \left(\frac{n-1}{2}+1\right) \\
&= \frac{(n-1)(n+1)n}{6} + \frac{(n-1)(n+1)}{2} + \frac{n+1}{2} \\
&= \frac{n^3 - n + 3n^2 - 3 + 3n + 3}{6} = \frac{n^3 + 3n^2 + 2n}{6} = \frac{n(n+1)(n+2)}{6}.
\end{aligned}$$

Kai $n = 2k+1 \to \infty$, gauname $f(n) \sim \frac{n^3}{6}$, t.y. $a = \frac{1}{6}$, $b = 3$.

Uždavinys 2 (0.3 balo). Duotas programos fragmentas su parametru n.

(a) Raskite tikslų žingsnių skaičių $L(n)$, laikant, kad bet kurios operacijos (priskyrimo, aritmetinės, palyginimo ir kt.) svoris yra 1. Žingsnių skaičius skaičiuojamas "blogiausiu atveju", t.y. maksimalus galimas "blogiausiems duomenims".

(b) Raskite $L(n)$ asimptotiką, t.y. konstantas a ir b tokias, kad $L(n) \sim an^b$, kai $n \to \infty$.

(c) Duota nedidelė konstanta c. Nurodykite duomenis, kuriems programa atliks lygiai $L(c)$ žingsnių ir išvardinkite tuos žingsnius.

(d) Raskite programos vykdymo laiko $T(n)$ eilę, t.y. konstantą d tokią, kad $T(n) = \Theta(n^d)$, kai $n \to \infty$. Skaičiuojant laiką $T(n)$, laikome, kad skirtingų operacijų (pvz. priskyrimo, aritmetinės, palyginimo) laikas yra skirtingas. Paprastumo dėlei mes nekreipsime dėmesio į pačias operacijas, o laikysime, kad vienkartinis programos kodo eilutės i įvykdymas reikalauja c_i laiko.

Nurodymai

1. $f(n) = \mathrm{O}(g(n))$ (arba $f(n) \preccurlyeq g(n)$) (sakome, kad "f asimptotiškai yra ne aukštesnės eilės dydis kaip g"), jei $\exists N \in \mathbb{N}$ ir $\exists c > 0$: $f(n) \leqslant cg(n)\ \forall n \geq N$;

2. $f(n) = \Theta(g(n))$ (arba $f(n) \asymp g(n)$) (sakome, kad "f ir g asimptotiškai yra tokios pat eilės dydžiai"), jei $f(n) = \mathrm{O}(g(n))$ ir $g(n) = \mathrm{O}(f(n))$.

3. Skaičiuojant žingsnius laikome, kad priskyrimo ir aritmetinė operacija yra 1 žingsnis, t.y. komanda $a := 1$ yra 1 žingsnis, komanda $a := b + c$ irgi yra vienas žingsnis, bet komanda $a := b + c - d$ yra 2 žingsniai. Komanda $A[i + j] := b + c$ taip pat reikalauja 2 žingsnių: (1) apskaičiuojame indekso reikšmę $k = i + j$, (2) masyvo elementui $A[k]$ priskiriame reikšmę $b + c$.

4. Ciklo **for** ilgio k "palaikymas", t.y. komanda **for** $i := 1$ **to** k **do**, reikalauja $2(k+1)$ žingsnių, nes kiekvieną kartą yra vykdoma sudėtis $i := i + 1$ ir palyginimas $i \leq k$?. Baigiant ciklą bus atlikta sudėtis $i := k + 1$ bei palyginimas $k + 1 \leq k$?, po kurių ciklo kūnas jau nebus vykdomas.

Pavyzdys 2. Duota sveikųjų skaičių masyvas $A[1 : n]$, konstanta $c = 2$ ir rūšiavimo programos INSERTION_SORT fragmentas

for $j := 2$ **to** n **do**
    $key := A[j]$
    $i := j - 1$
    while $i > 0$ **and** $A[i] > key$ **do**
        $A[i + 1] := A[i]$
        $i := i - 1$
    $A[i + 1] := key$

(a) Pirmiausia įvertinsime, kiek žingsnių atitiks kiekvieną programos eilutę:

for $j := 2$ **to** n **do**	$2n$
$\quad key := A[j]$	$n-1$
$\quad i := j-1$	$n-1$
$\quad$ **while** $i > 0$ **and** $A[i] > key$ **do**	$3\sum_{j=2}^{n} t_j + n - 1$
$\qquad A[i+1] := A[i]$	$2\sum_{j=2}^{n} t_j$
$\qquad i := i-1$	$\sum_{j=2}^{n} t_j$
$\quad A[i+1] := key$	$2(n-1)$

kur t_j yra ciklo **while** kūno vykdymo kartų skaičius, priklausantis nuo j. Ciklo sąlygos **while** $i > 0$ **and** $A[i] > key$ patikrinimas reikalauja 3 žingsnių (2 palyginimo operacijos ir loginė **and** operacija). Tačiau tikrinant ją paskutinį kartą (kai $i = 0$) laikome, kad po pirmo žingsnio (tikrinimo $0 < 0$?) nustatoma, kad sąlyga yra nepatenkinta ir išeinama iš ciklo. Taip darome todėl, kad priešingu atveju $A[0]$ reikšmė būtų neapibrėžta.

Nesunku pastebėti, kad INSERTION_SORT programai blogiausias atvejis yra tada, kai pradiniai duomenys yra išdėstyti "atvirkščia", t.y. mažėjančia tvarka. Tada kiekvienam i pradedant $i = j-1$ ir baigiant $i = 1$ turėsime $A[i] > key$, taigi blogiausiu atveju ciklo **while** kūnas bus vykdomas $t_j = j-1$ kartą.

Apskaičiuosime bendrą žingsnių skaičių:

$$
\begin{aligned}
L(n) &= 2n + 5(n-1) + 6\sum_{j=2}^{n} t_j = 7n - 5 + 6\sum_{j=2}^{n}(j-1) = 7n - 5 + \frac{6n(n-1)}{2} \\
&= 3n^2 + 4n - 5.
\end{aligned}
$$

(b) Kai $n \to \infty$, gauname, kad $L(n) \sim 3n^2$ (t.y. $a = 3$, $b = 2$), nes

$$
\lim_{n\to\infty} \frac{3n^2 + 4n - 5}{3n^2} = \lim_{n\to\infty}\left(1 + \frac{4}{3n} - \frac{5}{3n^2}\right) = 1.
$$

(c) Parinksime "blogiausius" duomenis, kai $n = 2$. Pakanka paimti bet kuriuos du skaičius, kad būtų $A[2] > A[1]$. Tarkime, kad $A = [7, 3]$. Kadangi $L(2) = 3 \cdot 4 + 8 - 5 = 15$, tai bus atlikta 15 žingsnių:

$j := 2$
$2 \leq 2$? YES
$key := 3$
$i := 1$
$1 > 0$? YES
$7 > 3$? YES
YES **and** YES (=YES)
$1 + 1 = 2$
$A[2] := 7$
$i := 0$
$0 > 0$? NO

$0 + 1 = 1$
$A[1] := 3$
$j := 3$
$3 \leq 2$? NO

(d) Skaičiuojant programos vykdymo laiką kiekvienai programos eilutei jos vykdymo kartų skaičių dauginsime iš konstantos c_i, atitinkančios tos eilutės vienkartinio vykdymo laiką. Gauname tokį rezultatą:

for $j := 2$ **to** n **do**	$c_1 n$
$key := A[j]$	$c_2(n-1)$
$i := j - 1$	$c_3(n-1)$
while $i > 0$ **and** $A[i] > key$ **do**	$c_4 \sum_{j=2}^{n}(t_j + 1)$
$A[i+1] := A[i]$	$c_5 \sum_{j=2}^{n} t_j$
$i := i - 1$	$c_6 \sum_{j=2}^{n} t_j$
$A[i+1] := key$	$c_7(n-1)$

Blogiausiu atveju vėl turime $t_j = j - 1$, taigi $\sum_{j=2}^{n} t_j = \sum_{j=2}^{n}(j-1) = \frac{n(n-1)}{2}$. Apskaičiuojame $T(n)$:

$$\begin{aligned} T(n) &= c_1 n + (c_2 + c_3 + c_4 + c_7)(n-1) + (c_4 + c_5 + c_6)\left(\frac{n^2}{2} - \frac{n}{2}\right) \\ &= \frac{c_4 + c_5 + c_6}{2} n^2 + \left(c_1 + c_2 + c_3 + c_4 + c_7 - \frac{c_4}{2} - \frac{c_5}{2} - \frac{c_6}{2}\right) n - (c_2 + c_3 + c_4 + c_7) \\ &= An^2 + Bn - C. \end{aligned}$$

Taigi, $T(n) = \Theta(n^2)$, nes galime rasti konstantą $D > 0$ bei natūralųjį skaičių N_1 tokius, kad $T(n) \leq Dn^2$ kiekvienam $n > N_1$ ir galime rasti konstantą $E > 0$ bei natūralųjį skaičių N_2 tokius, kad $n^2 \leq T(n)$ kiekvienam $n > N_2$. Vadinasi, programos vykdymo laiko $T(n)$ augimo eilė $d = 2$ (žr. uždavinio sąlygą (d)). Tokį algoritmą dar vadina "kvadratinio sudėtingumo" algoritmu.